Oracle® Communications Session Border Controller

Configuration Guide

Release S-Cz10.0.0
G20488-01
March 2025

ORACLE

Oracle Communications Session Border Controller Configuration Guide, Release S-Cz10.0.0

G20488-01

Copyright © 2025, 2025, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

About this Guide

My Oracle Support lxxi

Revision History

1 Oracle Communications Session Border Controller Basics

What Is a Realm 1-1
Nested Realms 1-1
What Is a Session Agent 1-1
    SIP session agents 1-2
    H.323 session agents 1-2
Why You Need Session Agents 1-2
How to Use Session Agents 1-2
What is a Session Agent Group 1-2
High Availability 1-3
Network Functions Virtualization Overview 1-4

2 Getting Started

Installation and Start-Up 2-1
    Hardware Installation Process 2-1
    Connecting to Your Oracle Communications Session Border Controller 2-2
        Create a Console Connection 2-2
        SSH Remote Connections 2-3
    System Boot 2-14
Boot Parameters 2-14
    Boot Parameter Definitions 2-16
    Boot Flags 2-17
Setting Up System Basics 2-17
    New System Prompt 2-17
    Set Initial Passwords for Admin and User 2-17
Using the Oracle Communications Session Border Controller Image 2-18

Obtaining a New Image 2-18
Copy an Image to the Oracle Communications Session Border Controller using SFTP 2-19
System Image Filename 2-20
Booting an Image on Your Oracle Communications Session Border Controller 2-20
Boot Parameter Definitions 2-20
Booting from Local Storage 2-21
Setting Up Product-Type, Features, and Functionality 2-23
System Setup 2-23
Setup Product 2-24
Setup Entitlements 2-24
Editing and Viewing Features 2-25
License Keys and Self-Provisioned Entitlements Compatibility 2-25
Concurrent Session License Usage 2-26
Adding and Deleting License Keys 2-28
Add a License Key 2-28
Delete a License Key 2-30
View Installed Features, Entitlements, and Licenses 2-31
Setup Features on an HA Pair 2-32
Configuration Assistant Operations 2-35
The Configuration Assistant Work Flow 2-35
Configuration Assistant ACLI Navigation Commands 2-38
User Accounts 2-38
Named SSH Keys 2-38
Local User Accounts 2-39
Manage Local Accounts 2-39
RADIUS Authentication 2-42
PAP Handshake 2-44
CHAP Handshake 2-44
MS-CHAP-v2 Handshake 2-45
Management Protocol Behavior 2-46
RADIUS Authentication Configuration 2-47
TACACS+ 2-49
TACACS+ Overview 2-50
TACACS+ Authentication 2-51
TACACS+ Authorization 2-53
TACACS+ Accounting 2-58
TACACS+ Configuration 2-59
Customizing Your ACLI Settings 2-66
Disabling the Second Login Prompt 2-67
Disabling the Second Login Prompt Configuration 2-67
Persistent ACLI more Parameter 2-67
Persistent ACLI more Parameter Configuration 2-67

Customize the Log On Banner Message 2-68

3 System Configuration

Virtual SBCs 3-1
CPU Core Configuration 3-2
Host Hypervisor CPU Affinity (Pinning) 3-4
Support for Hyperthreading Datapath CPUs 3-5
Datapath CPU Hyperthreading Considerations 3-7
System Shutdown 3-8
Configuration Overview 3-8
Configure Cores 3-9
Configure Hyperthreading Support 3-10
Session Router Session Capacity Enhancement 3-10
General System Information 3-11
System Identification 3-11
Connection Timeouts 3-11
Cluster Member Graceful Shutdown 3-11
Detailed Description of Graceful Shutdowns with Active SIP Calls or Registrations 3-11
High-level Procedure for Graceful Shutdown 3-13
Configuring General System Information 3-13
System Identification 3-14
Configuring Connection and Debug Logging Timeouts 3-14
Phy-Interfaces 3-15
Before You Configure 3-17
Phy-Interface Configuration 3-18
Interface Utilization: Graceful Call Control, Monitoring, and Fault Management 3-19
Calculation Overview 3-19
Alarms 3-19
Alarm Configuration 3-19
Network Interfaces 3-21
IP Configuration 3-21
VLANs 3-21
Overlapping Networks 3-21
Administrative Applications Over Media Interfaces 3-22
Configurable MTU Size 3-22
Disabling GARP and ND for out-of-subnet Addresses 3-23
Network Interface Configuration 3-23
Special Considerations 3-24
Network Interfaces Configuration 3-24
Scheduled External Backups 3-27
Scheduled External Configuration Backup 3-27

Scheduled External Log and Support-Info Backup 3-28
Configure Scheduled Log and Support Info Backups 3-30
Configure Scheduled Configuration Backups 3-31
IP Identification (ID) Field 3-33
IP Identification Field Configuration 3-33
SNMP 3-34
Syslog and Process Logs 3-34
Overview 3-35
Process Log Messages 3-35
Syslog and Process Logs Configuration 3-35
Syslog Configuration 3-36
Configure the Process Log Server 3-36
Host Routes 3-37
Host Routes Example 3-37
Host Route Configuration 3-38
Setting Holidays in Local Policy 3-38
Holidays Configuration 3-38
Opening TCP Ports 3000 and 3001 3-39
Enable System to Connect to SDM 3-39
DNS on the OCSBC 3-40
DNS Configuration 3-41
Retransmission Logic 3-41
DNS Support for IPv6 3-42
DNS Transaction Timeout 3-42
DNS Transaction Timeout Configuration 3-42
DNS Entry Maximum TTL 3-43
DNS Entry Max TTL Configuration per Network Interface 3-43
DNS-SRV Session Agent Recursion Error Handling 3-44
Interface and Realm Support of DNS Servers 3-45
DNS Re-query over TCP 3-46
DNS Re-query over TCP Config 3-46
Configurable DNS Response Size 3-47
DNS Response Size Configuration 3-47
Disabling Recursive DNS Queries for ENUM 3-47
DNS Server Status via SNMP 3-48
Persistent Protocol Tracing 3-48
About Persistent Protocol Tracing 3-48
About the Logs 3-49
Process Logs 3-49
Communication Logs 3-49
Protocol Trace Logs 3-49
Persistent Protocol Tracing Configuration 3-49

System Access Control 3-50
Adding an ACL for the Management Interface 3-51
Notes on Deleting System ACLs 3-52
System TCP Keepalive Settings 3-52
System TCP Keepalive Configuration 3-52
Configurable TCP Timers 3-54
Configuring TCP Connection Establishment 3-54
Configuring TCP Data Retransmission 3-55
Timer for Idle Connections 3-56
RTP TTL 3-57
Bidirectional Forwarding Detection 3-57
Gateway Health Checking with BFD 3-59
Using BFD To Signal Virtual Address Re-routing 3-61
Configuring a BFD Config 3-63
Configuring BFD Sessions 3-63
Displaying Information on BFD Operation 3-65
RAMdrive Log Cleaner 3-67
Applicable Settings 3-67
Clean-Up Procedure 3-68
Clean-Up Frequency 3-68
RAMdrive Log Cleaner Configuration 3-69
System Resource Monitoring 3-70
Reporting on the Resource Monitoring Function 3-74
Configure the Resource Monitoring Function 3-80
Configurable Alarm Thresholds and Traps 3-82
SNMP Traps 3-83
Alarm Thresholds Configuration 3-84
Alarm Synchronization 3-85
Caveats 3-86
Alarm Synchronization Configuration 3-86
Accounting Configuration 3-86
Stream Control Transfer Protocol Overview 3-87
SCTP Packets 3-87
SCTP Terminology 3-87
SCTP Message Flow 3-88
Congestion Control 3-90
Multi-Streaming 3-90
Delivery Modes 3-91
Multi-Homing 3-91
Multi-Homing and Path Diversity 3-92
Monitoring Failure Detection and Recovery 3-92
Configuring SCTP Support for SIP 3-93

Configuring an SCTP SIP Port 3-93
Configuring the Realm 3-94
Configuring Session Agents 3-95
Setting SCTP Timers and Counters 3-96
Setting the RTO 3-96
Setting the Heartbeat Interval 3-98
Setting the SACK Delay Timer 3-98
Limiting DATA Bursts 3-99
Setting Endpoint Failure Detection 3-100
Setting Path Failure Detection 3-101
Specifying the Delivery Mode 3-102
Example Configurations 3-102
Phy Interface Configuration 3-103
Network Interface Configuration 3-103
SIP Port Configuration 3-104
Realm Configuration 3-104
Session Agent Configuration 3-105
IPv6 Address Configuration 3-105
Access Control 3-106
Host Route 3-106
Local Policy 3-106
Network Interface 3-106
ENUM Server 3-107
Realm Configuration 3-107
Session Agent 3-107
SIP Configuration 3-107
SIP Interface SIP Ports 3-107
Steering Pool 3-108
System Configuration 3-108
Account Server 3-108
IPv6 Support for Management and Telemetry 3-108
IPv6 Default Gateway 3-109
IPv6 Link Local Addresses 3-109
Network Interfaces and IPv6 3-110
IPv6 Reassembly and Fragmentation Support 3-110
Access Control List Support 3-111
Data Entry 3-111
Homogeneous Realms 3-112
Parent-Child Network Interface Mismatch 3-112
Address Prefix-Network Interface Mismatch 3-112
RADIUS Support for IPv6 3-112
Supporting RADIUS VSAs 3-112

NTP Synchronization 3-113
Setting NTP Synchronization 3-113
FQDNs for Time Servers on the SBC 3-114
Resolution Process 3-116
Configuring NTP Using an FQDN - Wancom 3-117
Authenticated NTP 3-119
Monitoring NTP from the ACLI 3-120
View Statistics 3-121
View Status 3-121
HTTP Connection Management 3-121
Configure HTTP Connection Management 3-123
Telephony Fraud Protection 3-124
Telephony Fraud Protection Target Matching Rules 3-126
Telephony Fraud Protection File Activation 3-128
Telephony Fraud Protection Data Types and Formats 3-128
Configure Telephony Fraud Protection 3-129
Refresh the Telephony Fraud Protection File 3-130
Telephony Fraud Protection ACLI Show Commands 3-130
Telephony Fraud Protection verify-config 3-132
Fraud Protection XML Source File Example 3-132

4 Realms and Nested Realms

Overview 4-1
About Realms and Network Interfaces 4-2
About the SIP Home Realm 4-2
About Realms and Other Functions 4-2
Realms 4-2
Before You Configure 4-3
Configure realm-config 4-3
Identity and IP Address Prefix 4-3
Realm Interfaces 4-4
Realm Service Profile 4-4
QoS Measurement 4-5
QoS Marking 4-6
Address Translation Profiles 4-6
Realm Support of DNS Servers 4-6
DoS ACL Configuration 4-7
Enabling RTP-RTCP UDP Checksum Generation 4-7
Hiding Media Updates 4-7
Aggregate Session Constraints Per Realm 4-10
Admission Control Configuration 4-10

Nested Realms 4-10
Configuring Nested Realms 4-12
Parent and Child Realm Configuration 4-13
Required Signaling Service Parameters 4-13
Aggregate Session Constraints Nested Realms 4-13
Impact to Other Session Constraints and Emergency Calls 4-14
Session Constraints Configuration 4-15
Realm-Based Packet Marking 4-15
About TOS DiffServ 4-15
ToS Byte 4-16
DiffServ Byte 4-16
Packet Marking for Media 4-16
Configuring Packet Marking by Media Type 4-17
Packet Marking Configuration 4-17
Applying a Media Policy to a Realm 4-18
Signaling Packet Marking Configuration 4-19
Configuring a Media Policy for Signaling Packet Marking 4-19
Applying a Media Policy to a Realm 4-20
Using Class Profile for Packet Marking 4-20
Class Profile and Class Policy Configuration 4-21
Differentiated Services for DNS and ENUM 4-22
Differentiated Services for DNS and ENUM Configuration 4-23
SIP-SDP DCSP Marking ToS Bit Manipulation 4-24
ToS Bit Manipulation Configuration 4-25
DSCP Marking for MSRP and Media Over TCP 4-26
PAI Header and FQDN Manipulation 4-26
PAI Header Manipulation 4-26
FQDN Hostname Manipulation 4-28
Configure Manipulation Attributes on a Realm 4-28
SDP Alternate Connectivity 4-29
SDP Alternate Connectivity Configuration 4-30
Steering Pools 4-31
Configuration Overview 4-31
Allocation Strategies for Steering Pools 4-32
Steering Pool Allocation Examples 4-34
Monitoring Port Allocation 4-36
Steering Pool Configuration 4-37
Multiple Interface Realms 4-38
Steering Pool Port Allocation 4-41
Network Interface Configuration 4-41
Creating Steering Pools for Multiple Interface Realms 4-42
Media over TCP 4-42

TCP Bearer Conditions 4-43
TCP Port Selection 4-43
SDP Offer Example 4-47
Timers 4-48
TCP Port Configuration 4-48
Restricted Media Latching 4-49
About Latching 4-49
Restricted Latching 4-50
Symmetric Latching 4-50
Relationship to Symmetric Latching 4-50
Example 1 4-51
Example 2 4-51
Restricted Latching using Address and Port 4-51
Call Flows Supporting Restricted Latching to Address and Port 4-53
Restricted Latching Configuration 4-57
Media Release Across SIP Network Interfaces 4-58
Media Release Configuration 4-58
Media Release Behind the Same IP Address 4-59
Additional Media Management Options 4-59
Configuring Media Release Behind the Same IP Address 4-59
Bandwidth CAC for Media Release 4-60
Bandwidth CAC Configuration 4-61
Media Release between Endpoints with the Same IP Address 4-61
Media Release Configuration 4-61
Media Release Behind the Same NAT IP Address 4-62
Media Release Configuration 4-62
Codec Reordering 4-63
Preferred Codec Precedence 4-64
Codec Reordering Configuration 4-64
Setting a Preferred Codec for a Realm 4-65
Setting a Preferred Codec for a Session Agent 4-65
Media Profiles Per Realm 4-66
Call Admission Control and Policing 4-67
Media Profile Configuration 4-67
About Wildcarding 4-67
Multiple Media Profiles 4-69
Use Case 1 4-69
Use Case 2 4-69
Multiple Media Profiles Configuration 4-70
Per-Realm Media Guard Timers 4-70
SIP Disable Media Inactivity Timer for Calls Placed on Hold 4-71
Media Inactivity Timer Configuration 4-71

Media Manager Configuration for Virtual Machines 4-72

5 SIP Signaling Services

About the Oracle Communications Session Border Controller and SIP 5-1
Types of SIP Devices 5-1
Basic Service Models 5-1
About B2BUA 5-2
SIP B2BUA Peering 5-2
B2BUA Hosted IP Services 5-2
SIP B2BUA and L3 L5 NAT 5-3
About SIP Interfaces 5-3
SIP INVITE Message Processing 5-3
Example 5-4
Configuring the Oracle Communications Session Border Controller for SIP Signaling 5-4
Home Realm 5-5
Overview 5-5
SIP NAT Function 5-5
Home Realm's Purpose 5-6
Home Realm Configuration 5-6
SIP Interfaces 5-7
Overview 5-8
About SIP Ports 5-8
Preferred SIP Port 5-8
Proxy Mode 5-8
Redirect Action 5-9
SIP maddr Resolution 5-9
SIP maddr Resolution Configuration 5-10
Trust Mode 5-10
About the Process 5-11
Call Duration Counters 5-12
Configurable Timers and Counters 5-13
Timer to Tear Down Long Duration Calls 5-13
Timer to Tear Down Long Duration Calls Configuration 5-14
Asymmetric Preconditions 5-15
Asymmetric Preconditions Configuration Guidelines 5-16
Transcoder Free Operation for Asymmetric Preconditions 5-18
Dynamic Preconditions 5-24
Dynamic Preconditions Call Flows 5-26
Early Media Gating using Locally Generated 183 5-29
Changing the Media Direction Attribute 5-32
Strength Tag Support 5-32

Enhanced Preconditions Configuration 5-33
Preconditions and Multiple Early Dialogs 5-34
Precondition and MED merge Functional Overview 5-35
Static and Dynamic Preconditions with TrFO 5-38
Preconditions and MED with and without Delay 5-44
Related Preconditions and MED Call Flows 5-46
SIP Interface Configuration 5-54
Configuring SIP Ports 5-59
Asymmetric Preconditions Configuration 5-60
Diversion Info and History-Info Header Mapping 5-61
History-Info and Diversion Header Interworking Operations 5-62
History-Info to Diversion Header Interworking 5-64
Diversion to History-Info Header Interworking 5-65
Mapping the History-Info Cause Parameter 5-66
Anonymization of History and Diversion Information 5-67
Diversion and History-Info Headers Interworking Configuration 5-68
History-Info Cause 380 Parameter Interworking Configuration 5-68
Anonymize History Configuration 5-69
Digest Authentication with SIP 5-69
Challenge-Responses in Requests not in the Dialog 5-71
Configuring Digest Authentication 5-71
Additional Notes 5-73
Digest Authentication and High Availability 5-73
Surrogate Agents and the SBC 5-73
Surrogate Registration 5-74
Registration 5-74
Surrogate Agent Authentication Across Realms 5-76
Routing Calls from an IP-PBX 5-78
Surrogate Agent Refresh on Invalidate 5-86
Surrogate Agents Re-Register after Switchover 5-87
Surrogate Registration Configuration 5-88
Recurse 305 Only Redirect Action 5-94
Redirect Action Process 5-95
Redirect-Action Set to Proxy 5-95
Redirect-Action Set to Recurse 5-96
Redirect-Action Set to Recurse-305-Only 5-97
Redirect Configuration for SIP Interface 5-97
Redirect Configuration for Session Agent 5-98
Embedded Routes in Redirect Responses 5-99
Selecting SDP within Multi-Dialog Call Scenarios 5-100
UPDATE Interworking 5-100
UPDATE Interworking Configuration 5-102

SIP PRACK Interworking 5-102
UAC-Side PRACK Interworking 5-103
UAS-Side PRACK Interworking 5-104
PRACK Interworking Configuration 5-105
After Dialog Establishment (INVITE transaction terminated) 5-106
During dialog establishment (INVITE transaction not yet terminated) 5-106
Global SIP Timers 5-119
Overview 5-119
Timers Configuration 5-120
SIP Timers Discreet Configuration 5-121
Session Timer Support 5-122
Call Flow Example 5-122
SIP Per-User CAC 5-123
Per User CAC Modes 5-124
Per User CAC Sessions 5-124
Per User CAC Bandwidth 5-124
Notes on HA Nodes 5-125
SIP per User CAC Configuration 5-125
SIP Per-Realm CAC 5-126
SIP per Realm CAC Configuration 5-127
Enabling Realm-Based CAC 5-127
Viewing Realm-Based CAC Data 5-127
SIP Options Tag Handling 5-128
Overview 5-128
Configuration Overview 5-128
SIP Option Tag Handling Configuration 5-129
Replaces Header Support 5-130
New SDP Parameters in INVITE with Replaces 5-131
Early Dialog Replacement 5-132
INVITE with Replaces in Early Dialog Server Side 5-132
Replace Header Configuration 5-133
Debugging 5-133
show sipd status 5-133
show sipd errors 5-134
SIP Options 5-134
Overview 5-134
Global SIP Options 5-134
SIP Interface Options 5-141
SIP Session Agent Options 5-143
SIP Realm Options 5-143
SIP Realm Options Configuration 5-143
Configuring Multiple Options 5-144

Adding an Entry 5-144
SIP Security 5-144
Denial of Service Protection 5-145
Levels of DoS Protection 5-145
Configuration Overview 5-146
SIP Unauthorized Endpoint Call Routing 5-146
SIP Unauthorized Endpoint Call Routing Configuration 5-146
SIP NAT Function 5-147
Overview 5-147
NAT Modes 5-148
Adding a maddr Parameter to a URI 5-149
About Headers 5-149
Replacing Headers 5-149
Mapping FQDNs 5-150
SIP NAT Function Cookies 5-150
userinfo 5-150
host 5-151
URL Parameter 5-151
tel URL 5-152
Configuration Overview 5-152
SIP NAT Interface 5-152
SIP NAT Function Policies 5-153
SIP NAT Function Configuration 5-154
SIP Realm Bridging 5-158
About SIP NAT Bridging 5-158
SIP NAT Bridge Configuration Scenarios 5-159
Many to One Configuration 5-159
One-to-One Configuration 5-160
SIP NAT Bridge Configuration 5-161
Creating a Virtual Home Network 5-161
Many-to-One Configuration 5-161
One-to-One Configuration 5-162
Shared Session Agent 5-162
SIP Hosted NAT Traversal (HNT) 5-163
About SIP HNT 5-163
Using HNT with Existing NAT Device 5-163
Registering Endpoints 5-164
Establishing Media Flows 5-164
Prerequisites 5-164
Keeping the NAT Binding Open 5-164
Working with Multiple Domains 5-166
HNT Configuration Overview 5-167

SIP HNT Single Domain Example 5-167
SIP HNT Multiple Domain Example 5-167
HNT Configuration 5-168
Global SIP Configuration 5-170
Endpoint-Initiated Keep-Alives 5-172
Endpoint-Initiated Keep-Alive Negotiation 5-173
Connection Oriented Keep-Alives 5-174
Connectionless Keep-Alives 5-174
Enpoint-Initiated Keep-Alives Configuration 5-176
SD-originated Keep-Alive Negotiation 5-178
SD-Originated Keep-Alive Format 5-179
SD-Initiated Keep-Alives Configuration 5-179
Statistics 5-180
SIP Registration Local Expiration 5-181
SIP Registration Local Expiration Configuration 5-181
Simultaneous TCP Connection and Registration Cache Deletion 5-182
Registration Cache Deletion Configuration 5-182
SBC Incorrectly Appends Cookie in SIP REGISTER Message 5-183
process-implicit-tel-URI Configuration 5-183
SIP HNT Forced Unregistration 5-184
When to Use Forced Unregistration 5-184
Caution for Using Forced Unregistration 5-185
SIP HNT Forced Unregistration Configuration 5-185
Adaptive HNT 5-186
Overview 5-186
Adaptive HNT Example 5-186
Adaptive HNT over TCP 5-187
Adaptive HNT Call Flows 5-188
Synchronize A-HNT Successful Timer to Standby 5-195
Adaptive HNT Configuration 5-195
SIP IP Address Hiding and NATing in XML 5-196
Sample SIP NOTIFY with NATed XML 5-196
SIP Server Redundancy 5-197
Overview 5-198
Configuration Overview 5-198
SIP Server Redundancy Configuration 5-199
Administratively Disabling a SIP Registrar 5-200
Considerations for Implicit Service Route Use 5-200
Manual Trigger Configuration 5-201
Manual Trigger Confirmation 5-201
Surrogate Agent Refresh on Invalidate 5-202
Invalidate Registrations 5-202

Performance Impact 5-203
Media Inactivity Timer Configuration 5-203
Support for Encoded Multipart Message Bodies 5-203
Multipart Message Encoding Support Configuration 5-204
SIP Distributed Media Release 5-205
Overview 5-205
Endpoint Locations 5-205
Location of the Encoded Information 5-206
Example Distributed Media Release 5-206
Overview of SIP DMR Configuration 5-207
SIP DMR Configuration 5-208
Configuring the Realm 5-209
Add-On Conferencing 5-210
Overview 5-210
Caveats 5-210
Add-On Conferencing Scenario 5-210
SIP B2BUA Functionality 5-211
Contact Header Processing 5-211
Target Mapping and Conferences 5-211
Refer-To Header Processing 5-211
Add-on Conferencing Configuration 5-212
SIP REFER Method Call Transfer 5-212
Unsuccessful Transfer Scenarios 5-213
Call Flows 5-214
SIP REFER Method Configuration 5-215
REFER-Initiated Call Transfer 5-216
Supported Scenarios 5-217
Call Flows 5-218
Maintaining RBT During Transfers 5-220
REFER Source Routing 5-226
REFER Source Routing Configuration 5-226
180 & 100 NOTIFY in REFER Call Transfers 5-228
Sample Messages 5-230
180 and 100 NOTIFY Configuration 5-232
SIP REFER Re-Invite for Call Leg SDP Renegotiation 5-232
Scenario 5-232
Alterations to SIP REFER 5-233
More about SIP REFER 5-233
SIP REFER with Replaces 5-234
SIP REFER with Replaces Configuration 5-236
SDP Handling for Compatibility Configuration 5-237
SIP REFER-to-BYE 5-237

SIP hold-refer-reinvite 5-238
Enable hold-refer-reinvite - ACLI 5-238
SIP Roaming 5-239
Overview 5-239
Process Overview 5-239
Using Private IPv4 Addresses 5-239
Example 1 With a NAT Firewall 5-240
Example 2 Without a NAT Firewall 5-240
SIP Roaming Configuration 5-241
SIP REFER Call Transfer UUI Relay 5-242
SIP REFER Call Transfer UUI Relay Configuration 5-245
Embedded Header Support 5-245
Embedded Header Support Configuration 5-246
Dialog Transparency 5-246
Overview 5-246
Dialog Transparency Configuration 5-247
Route Header Removal 5-247
Route Header Removal Configuration 5-248
SIP Via Transparency 5-248
SIP Via Transparency Configuration 5-249
Symmetric Latching 5-250
Symmetric Latching Configuration 5-250
Enabling RTCP Latching 5-251
SIP Pre-emptive Symmetric Media Latching 5-251
Pre-emptive Symmetric Media Latching Configuration 5-252
SIP Number Normalization 5-252
Terminology 5-252
Calls from IP Endpoints 5-253
Calls from IP Peer Network 5-253
SIP Number Normalization Configuration 5-254
Realm 5-254
Session Agent 5-254
SIP Port Mapping 5-255
About SIP Port Mapping 5-255
How SIP Port Mapping Works 5-256
SIP Port Mapping Based on IP Address 5-257
About NAT Table ACL Entries 5-257
Using SIP Port Mapping 5-258
Dynamic Configuration 5-258
Registration Statistics 5-259
SIP Port Mapping Configuration 5-259
SIP Port Mapping for TCP and TLS 5-261

SIP Port Mapping Configuration for TCP TLS 5-262
SIP Configurable Route Recursion 5-263
Example 1 5-263
Example 2 5-264
SIP Route Recursion Configuration 5-265
Configuring a Session Agent for SIP Route Recursion 5-265
Configuring a SIP Interface for SIP Route Recursion 5-265
SIP Event Package Interoperability 5-266
SIP Event Package Interoperability Configuration 5-267
SIP Proxy Subscriptions 5-267
Topology Hiding 5-268
Feature Interaction 5-269
SIP Proxy Subscription Configuration 5-269
SIP REGISTER Forwarding After Call-ID Change 5-270
SIP REGISTER Forwarding Configuration 5-270
SIP Local Response Code Mapping 5-271
SIP Local Response Code Mapping Configuration 5-271
Creating a SIP Response Code Map 5-272
Assigning SIP Response Code Maps to Session Agents 5-273
Assigning SIP Response Code Maps to SIP Interfaces 5-273
Specific Reason Headers in 503 Response 5-274
Specific Reason Headers in 503 Response Configuration 5-274
Session Agent Ping Message Formatting 5-275
Session Agent Ping Message Formatting Configuration 5-275
SIP Session Agent Continuous Ping 5-276
SIP SA Continuous Ping Configuration 5-277
SIP PAI Stripping 5-278
SIP PAI Stripping Configuration 5-280
SIP Statuses to Q.850 Reasons 5-281
SIP-SIP Calls 5-282
Configure Reason and Cause Mapping for SIP-SIP Calls 5-282
Configure the System to Add Reason Headers 5-283
Calls Requiring IWF 5-284
Default Mappings 5-285
SIP Status 5-286
Trunk Group URIs 5-288
Terminology 5-289
Trunk Group URI Parameters 5-289
Originating Trunk Group URI Parameters and Formats 5-289
Terminating Trunk Group URI Parameters and Formats 5-291
Trunk Group Signaling Parameters 5-292
SIP Header and Parameter Manipulation 5-294

Trunk Group Routing 5-294
Trunk Group URIs and SIP Registration Caching 5-294
Trunk Group URI Configuration 5-295
Precedence Used for Trunk Group Configurations 5-295
Configuring SIP Manipulations 5-296
Setting the Trunk Group URI Mode for Routing 5-296
Configuring a Session Agent for Trunk Group URIs 5-297
Configuring a Session Agent Group for Trunk Group URIs 5-298
Setting a Trunk Group Context in a Realm 5-298
Using this Feature with a SIP Interface 5-299
Example 1 Adding Originating Trunk Group Parameters in IPTEL Format 5-299
Example 2 Adding Originating Trunk Group Parameters in Custom Format 5-300
Example 3 Removing IPTEL Trunk Group Names 5-300
Example 4 Removing Custom Trunk Group Names 5-301
Emergency Session Handling 5-301
Emergency Session Handling Configuration Procedures 5-302
Emergency Session Handling Configuration 5-302
Setting Policy Priority 5-303
Fraud Prevention 5-304
Fraud Prevention Configuration 5-304
Early Media Support 5-304
P-Early-Media SIP Header Support 5-305
P-Early-Media SIP Header 5-305
P-Early-Media-Header Usage 5-306
Functional Design 5-307
P-Early-Media Trusted to Trusted 5-308
P-Early-Media Untrusted to Trusted 5-310
P-Early-Media Trusted to Untrusted 5-311
Bypassing Early Media Gating 5-312
P-Early-Media ACLI Configuration 5-317
SIP Early Media Suppression 5-317
Example 5-318
Early Media Suppression Support 5-319
Call Signaling 5-319
Suppression Duration 5-320
About the Early Media Suppression Rule 5-320
Selective Early Media Suppression 5-320
SDP-Response Early Media Suppression 5-325
SIP-Based Addressing 5-325
SDP-Based Addressing 5-325
Configuring SDP-Response Early Media Suppression 5-327
Early Media Support for Multiple Early Dialog Scenarios 5-330

Merge Function within Early Dialog Support 5-331
Many-to-Many Support within Early Dialog Support 5-335
PEM Header Support for Early Dialogs 5-337
Configuring Multiple Early Dialog Support 5-338
Selecting SDP within Multi-Dialog Call Scenarios 5-339
SDP Compliance Enforcement 5-340
SDP Compliance Enforcement Configuration 5-340
SIP Duplicate SDP Suppression 5-341
SIP Duplicate SDP Suppression Configuration 5-341
SIP SDP Address Correlation 5-342
SIP SDP Address Correlation Configuration Address Checking 5-342
SIP SDP Address Correlation Configuration Mismatch Status Code 5-343
SIP SDP Address Correlation Configuration Enforcement Profile 5-343
SDP Insertion for (Re)INVITEs 5-344
SDP Insertion for SIP INVITES 5-344
SDP Insertion for SIP ReINVITEs 5-345
SDP Insertion Configuration 5-346
Configuring SDP Insertion for SIP INVITEs 5-346
Configuring SDP Insertion for SIP ReINVITEs 5-347
SDP Version Change without SDP Body Change 5-347
SDP Version Change Configuration 5-348
Enhanced SIP Port Mapping 5-348
Anonymous Requests 5-349
Anonymous SIP Requests Configuration 5-349
SIP Registration Via Proxy 5-349
Considerations for Reg-Via-Key and Port Mapping 5-350
Request Routing 5-350
SIP Registration Via Proxy Configuration 5-350
Dynamic Transport Protocol Change 5-351
Dynamic Transport Protocol Change Configuration 5-351
SIP Privacy Extensions 5-352
Privacy Types Supported 5-352
user 5-352
header 5-353
id 5-353
Examples 5-353
Calls from Untrusted Source to Trusted Target 5-353
Calls from Trusted to Untrusted 5-353
Calls from Trusted to Trusted 5-354
Configuring SIP Privacy Extensions 5-354
Trust Mode 5-354
Disabling the PPI to PAI Change 5-355

SIP Registration Cache Limiting 5-356
About Registration Cache Additions Modifications and Removals 5-356
Registration Cache Alarm Threshold 5-357
Notes on Surrogate Registration 5-357
Monitoring Information 5-357
SIP Registration Cache Limiting Configuration 5-357
SIP Registration Overload Protection 5-358
SIP Registration Overload Protection Configuration 5-359
SIP Registration Overload Protection for IMS-AKA 5-360
SIP Instance ID in Registration Cache 5-361
SIP Instance ID and the Registration Cache 5-362
SIP Instance ID Configuration 5-362
SIP Request Method Throttling 5-363
About Counters and Statistics 5-363
SIP Request Method Throttling Configuration 5-364
Rate Constraints for SIP Interfaces 5-364
Applying Session and Rate Constraints to a SIP Interface 5-365
Configuring Rate Constraints for Session Agents 5-366
Suppressing Re-INVITEs for Call Hold/Resume Dialogs 5-367
Call Flows for INVITE Suppression 5-369
Interaction Between Session Timers and re-INVITE Suppression 5-373
SIP Delayed Media Update 5-375
Delayed Media Update Disabled 5-375
Delayed Media Update Enabled 5-376
SIP Delayed Media Update Configuration 5-376
Expedited Call Leg Release for Preempted Hairpin Calls 5-377
Accounting Considerations 5-377
Supporting Released Flows that are Subsequently Hairpinned back to the SBC 5-378
SIPconnect 5-378
Modifications to Registration Caching Behavior 5-379
Configuring SIP Connect Support 5-379
Required Configuration 5-379
Suggested Additional Configuration 5-380
SIP Connect Configuration 5-380
SIP Registration Event Package Support 5-381
Updating Expiration Values 5-382
Contact Cache Linger Configuration 5-382
SIP Event Package for Registrations 5-383
Applicable Standards 5-383
Call Flow 5-383
Notification Bodies 5-386
SIP Event Package for Registrations Configuration 5-386

SIP Transport Selection 5-387
SIP Transport Selection Configuration 5-387
uaCSTA NAT Support 5-388
Overview 5-388
SIP Packet Cable Multi-Media 5-389
Details 5-390
Core-Side SDP Insertion Configuration 5-391
SIP Method-Transaction Statistic Enhancements 5-392
SIP Method Tracking Enhancements Configuration 5-392
National Security and Emergency Preparedness for SIP 5-393
Matching by NMC and by RPH 5-393
Call Treatment 5-395
Generating Egress RPH 5-395
Media Treatment 5-396
DSCP Marking for NSEP Traffic 5-396
Configure Realm-Specific DSCP Marking for NSEP Traffic 5-397
Configure a Media Policy for NSEP Traffic 5-398
Enable NSEP and Apply the RPH Profile to the NMC 5-399
Specify a Media Policy for a Realm's NSEP Traffic 5-400
Reserving Session Capacity for NSEP 5-401
Reject Non-Emergency Traffic using Emergency DSCP 5-403
Reporting on NSEP Traffic Statistics 5-404
RPH Configuration 5-405
Setting Up and Applying RPH Policy 5-405
Setting Up and Applying RPH Profile 5-406
Enabling NSEP for an NMC Rule 5-407
Global SIP Configuration Settings Enabling NSEP 5-408
Global SIP Configuration Settings Enabling CAC and Congestion Control 5-408
Global SIP Configuration Settings Enabling ARPH Insertion 5-409
Setting Up NSEP for Session Agents 5-410
Configure NSEP Resource Reservation 5-411
E-CSCF Emergency Setting Precedence for NMC 5-412
E-CSCF Emergency Configuration 5-412
SIP TCP Connection Reuse 5-413
SIP TCP Connection Reuse Configuration 5-414
SIP TCP Keepalive 5-414
SIP TCP Keepalive Configuration for Session Agents 5-415
SIP TCP Keepalive Configuration for SIP Interfaces 5-415
SIP Enforcement Profile and Allowed Methods 5-416
SIP Enforcement Profile Configuration 5-416
Setting Up and Enforcement Profile 5-416
Applying an Enforcement Profile 5-417

Local Policy Session Agent Matching for SIP 5-419
Local Policy Session Agent Matching Configuration 5-422
About Wildcarding 5-423
Monitoring 5-423
Enforcement Profile Configuration with subscribe-event 5-423
Setting Up Subscribe Dialog Limits 5-423
Applying an Enforcement Profile to a Realm 5-425
STUN Server 5-425
About STUN Messaging 5-425
STUN Server Functions on the Oracle Communications Session Border Controller 5-427
RFC 3489 Procedures 5-427
rfc3489bis Procedures 5-428
Monitoring 5-428
STUN Server Configuration 5-428
SIP GRUU 5-429
Contact Header URI Replacement 5-429
Record-Route Addition 5-430
GRUU URI Parameter Name 5-430
SIP GRUU Configuration 5-431
SIP SIP-I Interworking 5-431
SIP-SIPI IWF Operational Overview 5-432
SIP-SIPI Interworking Call Flows 5-434
Base SIP-SIPI IWF Functionality 5-436
SIP-SIPI IWF using HMR 5-437
SIP-SIPI IWF Using Session Translation 5-437
Additional Controls on CDR Population 5-439
SIP-SIPI IWF using Native Processing 5-440
IAM Interworking Support 5-441
BYE and REL Interworking Support 5-445
200OK (of BYE) and RLC Interworking Support 5-445
Interworking 183 Message Content 5-445
Interworking Call Flows with History-Info Information 5-448
Interworking IAM with Redirection Parameters 5-448
Interworking ACM or CPG with History-Info Headers 5-450
Interworking ANM or CON with History-Info Headers 5-452
Interworking Call Flows with Diversion Information 5-453
Interworking IAM with Diversion Information 5-453
Interworking ACM or CPG with Diversion Information 5-454
Interworking ANM or CON with Diversion Information 5-455
Interworking User-Configured IAM Parameters into SIP INVITE Headers 5-456
Additional ISUP IAM Generation Functions 5-458
Interworking SIP Response to ISUP Cause Values 5-460

In-Band Announcement Indication 5-460
Supplementary Service for COLP and COLR 5-461
Interworking for Call Hold and Resume Cases 5-461
ISUP Version Interworking 5-462
Configuring SIP-SIPI Interworking 5-463
Configuring SIP SIPI IWF using HMR 5-463
Configuring SIP-SIPI IWF using Session Translation 5-463
Configuring translation rules 5-463
Applying translation-rules to session-translations 5-464
Applying session-translations to realms or session-agents 5-465
Configuring Native SIP-SIPI Interworking 5-465
Configuring the SIP-ISUP Profile 5-466
Configuring ISUP Parameter Extraction 5-469
Apply Your Profiles to a Session Agent 5-469
Configuring SIP-ISUP Reason Header IWF 5-470
SIP Session Timer Feature 5-470
How the Session Timer Feature Works 5-471
SIP Session Timer Configuration 5-473
DTMF Conversion Processing 5-474
LMSD Offerless INVITE handling 5-476
RFC 4028 Session Timers 5-477
Ingress Call Leg 5-478
Setting 200 OK’s Session-Expire value 5-478
Refresher 5-478
Egress Call Leg 5-479
Outbound INVITE Message 5-479
UAS Initial Response 5-480
Session Refreshes 5-480
Oracle Communications Session Border Controller as Refresher 5-480
Oracle Communications Session Border Controller as Refresh Responder 5-481
Timer Expiration 5-481
Interaction with SIP Features 5-482
Examples 5-483
ACLI Configuration 5-486
Verify Config Validation 5-488
show sipd status 5-488
305 Response to Registrations on Secondary Interfaces 5-488
ACLI Instructions and Examples 5-489
notify sipd offload-users 5-489
show registration 5-490
show registration sipd 5-490
show sipd endpoint-ip 5-491

SNMP Configuration 5-491
SNMP 5-492

6 H.323 Signaling Services

Peering Environment for H.323 6-1
Overview 6-2
Signaling Modes of Operation 6-2
Back-to-Back Gateway Signaling 6-2
Back-to-Back Gatekeeper Proxy and Gateway 6-3
Interworking Gatekeeper-Gateway 6-4
Realm Bridging with Static and Dynamic Routing 6-5
Before You Configure 6-6
Global H.323 Settings 6-6
Global H.232 Settings Configuration 6-6
Accessing Global H.323 Parameters 6-6
Global H.323 Settings 6-7
H.323 Interfaces 6-8
H.232 Interfaces Configuration 6-8
Identity and State 6-9
Realm and Interface Associations 6-9
H.323 Signaling Interface Settings 6-9
H. 323 System Resource Allocation 6-10
H.323 Service Modes 6-10
H.232 Service Modes Configuration 6-11
Configuring Gateway Only Settings 6-11
Gatekeeper Proxy Settings 6-12
H.323 Features 6-12
Fast Start Slow Start Translations 6-13
Fast Start to Slow Start Translation 6-13
Slow Start to Fast Start Translation 6-13
Slow Start Fast Start Prerequisites 6-14
Media Profile Configuration 6-15
Fast Start/Slow Start Configurations 6-17
H.235 Encryption 6-18
RFC 2833 DTMF Interworking 6-19
About RFC 2833 6-19
About H.245 UII 6-20
About 2833 to H.245 UII Interworking 6-20
About DTMF Transfer 6-20
Preferred and Transparent 2833 6-21
Preferred 2883 Support 6-21

Transparent 2833 Support 6-22
Basic RFC 2833 Negotiation Support 6-23
H.323 to H.323 Negotiation 6-23
Signal and Alpha Type Support 6-24
H.323 Endpoints 6-24
Translating H.245 UII to 2833 for H.323 Calls 6-25
RFC 2833 Mode Configuration 6-25
RFC 2833 Payload Configuration 6-26
RFC 2833 SA Configuration 6-26
H.323 Registration Proxy 6-27
H.235 Authentication Transparency 6-28
Unique CSA Per Registered Gateway 6-28
Virtual Call Signaling Address 6-28
Virtual RAS Address 6-29
RAS Message Proxy 6-29
About Setting Port Ranges 6-29
H.323 Registration Proxy Configuration 6-30
H.323 Registration Caching 6-31
Caveats for Registration Caching 6-32
Configuration Requirements 6-32
H.323 Registration Caching Configuration 6-32
Configuring the Gatekeeper Interface for Registration Caching 6-34
ACLI Registration Caching Configuration Example 6-35
H.245 Stage 6-36
Dynamic H.245 Stage Support 6-36
Dynamic H.245 Stage for Incoming Calls 6-36
Dynamic H.245 Stage for Outgoing Calls 6-37
H.245 Stage Configuration 6-38
H.323 HNT 6-38
Caveats 6-39
H.323 HNT Configuration 6-40
H.323 Party Number-E.164 Support 6-40
Signaling Only Operation 6-41
H.245 6-41
H.225 6-42
Maintenance Proxy Function 6-42
Maintenance Proxy Configuration 6-43
Applying TCP Keepalive to the H.323 Interface 6-43
Automatic Gatekeeper Discovery 6-44
Automatic Gatekeeper Configuration 6-44
H.323 Alternate Routing 6-45
Without Alternate Routing Enabled 6-45

With Alternate Routing Enabled 6-45
H.323 Alternate Routing Configuration 6-46
H.323 LRQ Alternate Routing 6-47
Caveats 6-48
H.323 LRQ RAS Retransmission Configuration 6-49
H.323 LRJ Limit Configuration 6-49
H.323 CAC Release Mechanism 6-50
H.323 CAC Release Configuration 6-50
H.323 Per-Realm CAC 6-51
Caveats 6-52
H.323 Per-Realm CAC Configuration 6-52
H.323 Bearer-Independent Setup 6-53
H.323 BIS Disabled 6-53
H.323 BIS Enabled 6-53
H.323 BIS Global Configuration 6-54
H.323 BIS Specific Configuration 6-54
TOS Marking for H.323 Signaling 6-55
H.323 Codec Fallback 6-55
Codec Fallback Disabled 6-55
Codec Fallback Enabled 6-56
H.323 Codec Fallback Configuration 6-57
H.323 TCS Media Sample Size Preservation 6-58
Media Sample Size Configuration 6-59
H.323-TCS H.245 Support for H.264 and G722.1 6-60
H.323-TCS Generic Video Configuration 6-60
H.323-TCS Generic Audio Configuration 6-61
International Peering with IWF and H.323 Calls 6-62
Default OLC Behavior Changed in Upgrade 6-63
Options 6-64
Global H.323 Options 6-64
H.323 Interface Options 6-65
H.323 Stack Monitoring 6-66
H.323 Stack Monitoring Configuration 6-67
H.323 Automatic Features 6-67
Alias Mapping 6-68
Call Hold and Transfer 6-68
Call Hold and Transfer Basic Call 6-68
Call Hold and Transfer Music on Hold 6-70
Call Hold and Transfer 6-70
Media Release for SS-FS Calls 6-73
Dependencies 6-74
Hold-and-Resume Procedure 6-75

H.323 and IWF Call Forwarding 6-75
Previous Behavior 6-75
New Behavior 6-75
H.323 Sample Call Flow 6-76
H.323 NOTIFY Support 6-77
Caveats 6-77
H.323 H.239 Support for Video+Content 6-77
Multiple Media Streams with the Same Payload 6-78
Support for Generic Capabilities 6-78
Support for H.239 Generic Messages 6-79
Support for Miscellaneous Indication 6-80
Video Conferencing Support for Polycom Terminals 6-80
ACLI Signaling Mode Configuration Examples 6-81
Configuration Fields and Values for B2BGW Signaling 6-81
Back-to-Back Gatekeeper Proxy and Gateway 6-84
Interworking Gatekeeper-Gateway 6-86
Additional Information 6-88
About Payload Types 6-88
Payload Types for Standard Audio and Visual Encodings 6-89
About RAS Message Treatment 6-90

7 IWF Services

Access Network Application 7-1
Networking Peering Application 7-2
SIP and H.323 7-2
SIP H.323 Negotiation H.323 Fast Start 7-2
SIP to Fast Start H.323 7-3
H.323 Fast Start to SIP 7-3
SIP H.323 Negotiation H.323 Slow Start 7-4
H.323 SIP to Slow Start 7-4
H.323 Slow Start to SIP 7-5
Status and Codec Mapping 7-6
IWF Termination from H.323 7-6
IWF Termination During H.323 RAS 7-7
IWF RAS Registration Failure Code Mapping 7-7
IWF Termination from SIP 7-9
Q.850 Cause to H.323 Release Complete Reason 7-9
Codec Mapping 7-10
IWF Service Enhancements 7-10
SIP Redirect—H.323 LRQ Management 7-11
Redirect—LRQ Management Sample 1 7-11

Redirect—LRQ Management Sample 2 7-12
Redirect—LRQ Management Sample 3 7-12
SIP INFO and DTMF UII Management 7-13
Mid-Session Media Change 7-13
Enhanced Support for FAX Calls 7-14
Removing the T.38 Codec from an H.245 TCS 7-14
Early Media 7-14
Display Name Mapping 7-15
IWF Ringback Support 7-15
Sample 1 In-band Ringback without Progress Message 7-16
Sample 2 In-band Ringback with Progress Message 7-17
Sample 3 In-band Ringback without Alerting Message 7-18
Sample 4 Out-of-band Ringback without Progress Message 7-19
Sample Flow 5 Out-of-band Ringback with Progress Message 7-19
H.323 Endpoint-Originated Call Hold and Transfer 7-20
Basic Call 7-21
Hold 7-22
Music On Hold 7-23
Transfer 7-24
Conference 7-25
IWF Call Forwarding 7-26
New Behavior 7-27
H.323 Sample Call Flow 7-27
Media Release for H.323 SS-FS Calls for IWF 7-28
H.323 7-28
Hold-and-Resume Procedure 7-29
Additional IWF Steps 7-30
Dependencies 7-31
Before You Configure 7-31
H.323 Configuration 7-31
SIP Configuration 7-31
The Role of Local Policy 7-32
Local Policy in an IWF Session Initiated with H.323 7-32
Local Policy in an IWF Session Initiated with SIP 7-33
SIP-H.323 interworking with Dynamic Payload Types 7-33
Configuring Interworking 7-36
Configure IWF 7-36
Topology Hiding for IWF with an Internal Home-Realm 7-37
IWF Topology Hiding Configuration 7-38
DTMF Support 7-38
DTMF Configuration 7-39
Applying the Media Profile 7-40

RFC 2833 DTMF Interworking 7-41
About RFC 2833 7-42
About H.245 UII 7-42
About RFC 2833 to H.245 UII Interworking 7-42
About DTMF Transfer 7-42
Preferred and Transparent 2833 7-43
Preferred 2883 Support 7-44
Transparent 2833 Support 7-44
Payload Type Handling 7-45
Basic RFC 2833 Negotiation Support 7-46
H.323 to H.323 Negotiation 7-46
Signal and Alpha Type Support 7-47
H.323 to SIP Calls 7-47
SIP Endpoints 7-47
H.323 Non-2833 Interworking with SIP 7-47
How H.323 to SIP Calls Work 7-48
SIP INFO—RFC 2833 Conversion 7-49
IPv6 SIP INFO to RFC 2833 Telephone Event Interworking 7-49
RFC 2833 Interworking Configuration 7-49
RFC 2833 Mode for H.323 Stacks 7-49
RFC 2833 Payload for H.323 7-50
Configuring the SIP Interface 7-50
Configuring Session Agents 7-51
Enabling Payload Type Handling 7-52
DTMF Transparency for IWF 7-54
DTMF Transparency Configuration 7-54
RFC 2833 Packet Sequencing 7-55
RFC 2833 Packet Sequencing Configuration 7-55
SIP Tel URI Support 7-55
SIP Interface Configuration 7-56
Graceful DTMF Conversion Call Processing 7-57
IWF Inband Tone Option 7-58
IWF Inband Tone Configuration 7-59
RFC 3326 Support 7-59
Default Mappings 7-61
RFC 3326 Support Configuration 7-64
IWF Privacy Caller Privacy on Unsecure Networks 7-65
About the Presentation Indicator 7-66
H.323 to SIP IWF Call 7-66
Example 1 SETUP Sent from h323d to Remote H.323 Endpoints 7-66
Example 2 INVITE from h323d to sipd 7-67
SIP to H.323 7-67

Example INVITE from SIP End Point to sipd 7-68
IWF Privacy Caller Privacy on Secure Connections 7-69
H.323 to SIP IWF 7-69
Calls with Presentation Allowed 7-70
H.323 to SIP 7-70
Sample SETUP sent from h323d to Remote H323 Endpoints 7-70
SIP to H.323 7-71
Example 1 INVITE from sip EP to sipd 7-71
Example INVITE from sipd to h323d 7-71
IWF Privacy Extensions for Asserted Identity in Untrusted Networks 7-72
IWF Call Originating in H.323 7-73
Sample H.323 Setup from a Remote Endpoint 7-73
Sample SIP INVITE from the SBC to a SIP Endpoint 7-74
Before You Configure 7-74
P-Preferred-Identity Configuration 7-75
IWF Privacy for Business Trunking 7-75
A Call Originating in H.323 7-76
Sample SETUP Message from an H.323 Endpoint 7-76
Sample INVITE from the Oracle Communications Session Border Controller to the SIP Endpoint 7-77
A Call Originating in SIP 7-78
Sample INVITE from a SIP Endpoint to the Oracle Communications Session Border Controller 7-78
Sample SETUP from the Oracle Communications Session Border Controller to the H.323 Endpoint 7-78
allowCPN Configuration 7-79
Trunk Group Documentation 7-81
IWF COLP COLR Support 7-81
SIP to H.323 Calls 7-81
H.323 to SIP Calls 7-82
IWF COLP COLR Configuration 7-82
Options for Calls that Require the IWF 7-83
Global Configuration for H.323 7-83
Individual Configuration for H.323 7-84
Configuring H.323 SA Options 7-84
H.323 SA Options 7-85
Suppress SIP Reliable Response Support for IWF 7-85
suppress100rel Configuration 7-86
IWF Codec Negotiation H.323 Slow Start to SIP 7-86
IWF Codec Negotiation Configuration 7-87
IWF H.245 Signaling Support for G.726 7-87
H.245 and G.726 Configuration 7-88
Media Profile for H.245 and G.726 Configuration 7-88

Media Profile Configuration for Generic Audio Support 7-89
Flow Control Mapping for Interworking Function (IWF) Video 7-89
Customized G.729 Support 7-91
About Dynamic Payload Mapping 7-92
Customized G.729 Configuration 7-92
SIP-H.323 IWF Support for H.264 and H.263+ 7-93
H.264 in H.323 (H.241) 7-93
Capabilities 7-95
H.264 Media Packetization 7-95
H.264 in SIP 7-95
H.264 Packetization Mode 7-96
H.264 IWF Conversions 7-96
IWF Unsupported Parameters 7-97
H.263+ in H.323 7-97
H.263+ in SIP 7-98
H.263+ IWF Conversions 7-98
IWF Unsupported Parameters 7-99
SIP-H.323 IWF in Video Conferencing Applications 7-100
International Peering with IWF and H.323 Calls 7-100
International Peering Configuration 7-100
IWF Codec Renegotiation for Audio Sessions 7-101
Codec Request Change from the SIP Side 7-102
Codec Request Change from the H.323 Side 7-102
Exceptional Cases 7-102
IWF Codec Renogotiation Configuration 7-102

8 Application Layer Gateway Services

DNS ALG 8-1
Overview 8-1
Configuring DNS ALG Service 8-2
Before You Configure 8-2
DNS ALG Service Name Configuration 8-3
Identity Realm and Interface Addresses 8-3
DNS Server Attributes 8-4
DNS Transaction Timeout 8-6
DNS Transaction Timeout Configuration 8-6
DNS Server Operation States 8-6
DNS Entry Maximum TTL Configuration for DNS ALG 8-7
DNS ALG Message Throttling 8-7
Bursty Traffic Throttling 8-8
Sustained Traffic Throttling 8-8

Maximum Latency 8-8
DNS ALG Constraints Configuration 8-8
Applying Traffic Constraints 8-9
ACLI DNS ALG Statistics 8-10
Other Show commands 8-10
SNMP DNS ALG Statistics and Traps 8-10

9 Session Routing and Load Balancing

Routing Overview 9-1
Session Agents Session Groups and Local Policy 9-1
About Session Agents 9-2
SIP Session Agents 9-3
Session Agent Status Based on SIP Response 9-3
SIP Pings from Multiple Realms to Global SAs 9-4
SIP Session Agent Continuous Ping 9-7
SIP SA Continuous Ping Configuration 9-8
Ingress Session Agent Identification 9-9
About Master Agent and Associated-Agents 9-10
Session-Agent Identification Example 9-10
Legacy Configuration 9-12
Override Alphanumeric Ordering of Session Agents with same IP address 9-14
Override Alphanumeric Ordering of Session Agents with Same IP Configuration 9-14
H.323 Session Agents 9-15
Overlapping H.323 Session Agent IP Address and Port 9-15
Managing Session Agent Traffic 9-15
Session Agent Groups 9-17
Request URI Construction as Forwarded to SAG-member Session Agent 9-18
SIP Session Agent Group Recursion 9-19
Session Agent Group Naming Support 9-19
About Local Policy 9-20
Routing Calls by Matching Digits 9-20
SIP and H.323 Interworking 9-21
Route Preference 9-21
DTMF-Style URI Routing 9-22
SIP Routing 9-22
Limiting Route Selection Options for SIP 9-22
About Loose Routing 9-22
About the Ingress Realm 9-23
About the Egress Realm 9-23
Ping Message Egress Realm Precedence 9-24
Normal Request Egress Realm Precedence 9-24

Session Agent Egress Realm Configuration 9-24
About SIP Redirect 9-25
Proxy Redirect 9-25
Tunnel Redirect 9-25
SIP Method Matching and To Header Use for Local Policies 9-25
SIP Methods for Local Policies 9-25
Routing Using the TO Header 9-26
H.323 Routing 9-27
Egress Stack Selection 9-27
Static Stack Selection 9-28
Policy-Based Stack Selection 9-28
Registration Caching 9-29
Gatekeeper Provided Routes 9-29
Back-to-Back Gateway 9-29
Back-to-Back Gatekeeper and Gateway 9-30
Interworking Gatekeeper Gateway 9-31
Load Balancing 9-31
Parallel Call Forking 9-32
Forking Operation 9-36
Parallel Forking Call Flows 9-39
Configuring Routing 9-44
Configuration Prerequisite 9-44
Configuration Order 9-44
Routing Configuration 9-44
Configuring Session Agents 9-45
Session Agent Group Configuration 9-55
SAG Matching for LRT and ENUM 9-56
Configuring Local Policy 9-57
Local Policy Matching for Parent Realms 9-62
SIP Session Agent DNS-SRV Load Balancing 9-63
Session Agent DNS-SRV Load Balancing Configuration 9-64
Answer to Seizure Ratio-Based Routing 9-64
ASR Constraints Configuration 9-65
SIP Recursion Policy 9-67
SIP Recursion Policy Configuration 9-68
ENUM Lookup 9-69
How ENUM Works 9-69
Translating the Telephone Number 9-69
About NAPTR Records 9-70
About the Oracle Communications Session Border Controller ENUM Functionality 9-70
Configurable Lookup Length 9-70
UDP Datagram Support for DNS NAPTR Responses 9-71

Custom ENUM Service Type Support 9-71
ENUM Failover and Query Distribution 9-71
ENUM Query Distribution 9-71
Failover to New enum-config 9-72
ENUM Server Operation States 9-72
Server Availability Monitoring 9-72
ENUM Server IP Address and Port 9-72
Unapplicable SNMP Traps and Objects 9-73
IPv6 ENUM SNMP Traps and Objects 9-73
Caching ENUM Responses 9-75
Source URI Information in ENUM Requests 9-75
Operation Modes 9-75
Stateless Proxy Mode 9-75
Transaction Stateful Proxy 9-76
Session Stateful Proxy 9-76
B2BUA 9-76
Example ENUM Stateless Proxy 9-77
ENUM Configuration 9-77
Example 9-80
Configuring the Local Policy Attribute 9-80
Local Policy Example 9-81
CNAM Subtype Support for ENUM Queries 9-82
CNAM Unavailable Response 9-82
SIP Profile Inheritance 9-82
CNAM Subtype Support Configuration 9-83
Using the Local Route Table (LRT) for Routing 9-83
Local Route Table (LRT) Performance 9-84
Multi-Tiered LRT Route Selection 9-85
Local Routing Configuration 9-88
Configure Local Routing 9-89
Applying the Local Routing Configuration 9-89
Multiple Contact Handling in Redirect Action for LRT 9-90
Local Route Table Support for H.323 and IWF 9-91
IWF Considerations 9-91
ENUM LRT Responses 9-91
LRT Entry Matching 9-92
LRT Entry Matching Configuration 9-92
LRT String Lookup 9-92
LRT String Lookup Configuration 9-93
Directed Egress Realm from LRT ENUM 9-93
Directed Egress Realm Configuration 9-94
SIP Embedded Route Header 9-94

SIP Embedded Route Header Configuration 9-95
LRT Lookup Key Creation 9-95
Arbitrary LRT Lookup Key 9-96
Hidden Headers for HMR and LRT lookup 9-96
Compound Key LRT Lookup 9-96
Retargeting LRT ENUM-based Requests 9-96
Re-targeting LRT ENUM-based Requests Configuration 9-97
Recursive ENUM Queries 9-98
Recursive ENUM Queries Configuration 9-98
Multistage Local Policy Routing 9-99
Routing Stages 9-99
Multi-stage Routing Source Realm 9-99
Network Applications 9-99
Multistage Routing Conceptual Example 9-100
Multistage Routing Example 2 9-100
Customizing Lookup Keys 9-104
Multistage Routing Lookup Termination 9-104
Global Local Policy Termination 9-104
Multistage Local Policy Routing Configuration 9-105
Maintenance and Troubleshooting 9-106
Traps 9-106
Routing-based RN and CIC 9-106
Routing-based RN Configuration 9-107
Codec Policies for SIP 9-108
Relationship to Media Profiles 9-109
Manipulation Modes 9-109
In-Realm Codec Manipulation 9-110
Codec Policy Configuration 9-111
Creating a Codec Policy 9-111
Applying a Codec Policy to a Realm 9-112
Applying a Codec Policy to a Session Agent 9-112
In-Realm Codec Manipulations 9-113
QoS Based Routing 9-113
Management 9-114
QoS Contraints Configuration 9-114
Configuring QoS Constraints 9-114
Applying QoS Constraint to a Realm 9-115

10 Session Translation

Session Translation Headers 10-2
Session Translation Configuration 10-4

Configure Translation Rules 10-4
Configure Session Translation 10-5
Apply a Session Translation 10-6
Apply a Session Translation to a Session Agent 10-6
Apply a Session Translation to a Realm 10-7
Message Types 10-8

11 Admission Control and QoS

About Call Admission Control 11-1
Bandwidth-Based Admission Control 11-1
Multi-Level Bandwidth Policy Nesting 11-2
Session Capacity- and Rate-based Admission Control 11-3
Constraints for Proxy Mode 11-3
CAC Policing and Marking for non-Audio non-Video Media 11-4
Bandwidth CAC Fallback Based on ICMP Failure 11-4
Bandwidth CAC Fallback Based on ICMP Failure Configuration 11-5
Bandwidth CAC for Aggregate Emergency Sessions 11-5
Bandwidth CAC for Aggregate Emergency Sessions Configuration 11-6
Admission Control for Session Agents 11-7
Session Agents Admission Control Configuration 11-7
Realm Bandwidth Configuration 11-9
SIP Admission Control Configuration 11-10
Aggregate Session Constraints for SIP 11-11
Aggregate Session Constraints Configuration 11-12
Applying Session Constraints in a SIP Interfaces 11-14
Configuring CAC Policing and Marking for non-Audio non-Video Media 11-14
Support for the AS Bandwidth Modifier 11-15
Media Profile Configuration 11-15
AS Modifier and Headroom Configuration 11-16
Offerless Bandwidth CAC for SIP 11-17
Offerless Bandwidth CAC for SIP Configuration 11-17
Shared CAC for SIP Forked Calls 11-18
Bandwidth Sharing Scenarios 11-18
Bandwidth Sharing Configuration 11-19
Configuring a SIP Profile 11-19
Applying a SIP Profile 11-20
RADIUS Accounting Support 11-20
Monitoring 11-20
Conditional Bandwidth CAC for Media Release 11-21
About Conditional Bandwidth CAC for Media Release 11-21
Details and Conditions 11-21

INVITEs UPDATEs Initially Received By Oracle Communications Session Border Controller 11-21
INVITEs UPDATEs Received by Second SBC 11-22
Conditional Admission with Per-user CAC 11-23
Conditional Bandwidth CAC Configuration 11-23
SIP Profile Configuration 11-23
Applying a SIP Profile 11-24
Configuring Require Header Option Tag 11-25
CAC Utilization Statistics via SNMP 11-25
CAC utilization threshold trap on a session agent configuration 11-28
Configuring the CAC Utilization Thresholds - realm 11-29
About QoS Reporting 11-29
Overview 11-30
QoS Statistics 11-30
Incremental QoS Updates 11-31
RADIUS Support 11-32
Configuring QoS 11-33
QoS Configuration 11-34
Network Management Controls 11-34
Matching a Call to a Control Rule 11-35
Call Handling Determination 11-35
Treatment Methods 11-36
Priority Call Exemption from Policy Server Approval 11-37
Enhanced Call Gapping 11-37
About the Call Gapping Algorithm 11-37
Network Management Control Configuration 11-38
Configuring an Individual Control Rule 11-38
Enabling Enhanced Call Gapping 11-40
Applying a Network Management Control Rule to a Realm 11-41
3147 - Emergency Call Fallback 11-42
Emergency Call Fallback Configuration 11-42
Accounting Configuration for QoS 11-42
QoS Accounting Configuration 11-42
Account Configuration 11-43
Account Server 11-46
Allowlists for Managing Incoming SIP Headers and Parameters 11-47
What is an Allowlist? 11-48
Configure Allowlists for SIP Header and URI Parameter Management 11-48
Configuration Exception 11-51
Verify Allowlist Configuration 11-51
How Allowlists Work 11-51
Allowlist Learning 11-52

Allowlist Learning Configuration 11-53
Rejected Messages Monitoring 11-53

12 Static Flows

About Static Flows 12-1
IPv6 / IPv4 Translations 12-2
About Network Address Translation ALG 12-2
NAPT 12-2
TFTP 12-3
Configuring Static Flows 12-4
Basic Static Flow Configuration Overview 12-4
Static Flow Configuration 12-4
Example Configuration: Bidirectional Static Flows 12-8

13 High Availability Nodes

Overview 13-1
Establishing Active and Standby Roles 13-2
Health Score 13-2
Switchovers 13-3
Automatic Switchovers 13-3
Manual Switchovers 13-3
State Transitions 13-3
State Transition Sequences 13-4
HA Features 13-4
Multiple Rear Interfaces 13-5
Configuration Checkpointing 13-5
Gateway Link Failure Detection and Polling 13-5
Georedundant High Availability (HA) 13-6
Before Configuring a High Availability (HA) Pair 13-7
HA Node Connections 13-8
Virtual MAC Addresses 13-10
Virtual MAC Address Configuration 13-10
Virtual MAC Addresses for VNFs 13-12
HA Node Connections 13-12
HA Node Connection Configuration 13-13
Rear Interfaces 13-13
Media Interface Virtual MAC Addresses 13-14
Alarms and Traps for Wancom1 and 2 Interfaces 13-15
HA Node Parameters 13-17
HA Node Parameter Configuration 13-18

HA Node Peer Configuration 13-21
HA Node Health And State Configuration 13-21
Synchronizing Configurations 13-22
Synchronize HA Peers 13-22
RTP Timestamp Synchronization 13-23
Using Configuration Checkpointing 13-23
HA Configuration Checkpointing 13-24
Manually Checking Configuration Synchronization 13-26
Media Interface Link Detection and Gateway Polling 13-27
Media Interface Link Detection and Gateway Polling Configuration 13-27
Media Interface Link Detection and Gateway Polling Configuration 2 13-28
Signaling Checkpointing 13-29
SIP Signaling Checkpointing 13-29
Signaling Checkpointing Configuration 13-30
Media State Checkpointing 13-31
Media State Checkpointing Configuration 13-31
Limiting the Rate of Gratuitous ARP (GARP) 13-32
HA Media Interface Keepalive 13-32
Impact to Boot-Up Behavior 13-33
HA Media Interface Keepalive Configuration 13-33
Critical Memory Switchover 13-34
RTC Notes 13-35
HA 13-36
Protocol-Specific Parameters and RTC 13-36

14 Security

Security Overview 14-1
Denial of Service Protection 14-2
Levels of DoS Protection 14-3
About the Process 14-4
Trusted Path 14-5
Address Resolution Protocol Flow 14-5
Untrusted Path 14-5
IP Fragment Packet Flow 14-6
Fragment Packet Loss Prevention 14-6
Static and Dynamic ACL Entry Limits 14-6
Dynamic Deny for HNT 14-7
Host and Media Path Protection Process 14-8
SBC Access Control 14-8
Access Control for Hosts 14-8
Access Control Endpoint Classification Capacity and DoS 14-9

Media Access Control 14-9
Host Path Traffic Management 14-9
Traffic Promotion 14-9
Malicious Source Blocking 14-9
Blocking Actions 14-10
Protecting Against Session Agent Overloads 14-10
ARP Flood Protection Enhancements 14-10
Dynamic Demotion for NAT Devices 14-10
DoS Counter Notifications 14-11
DDoS Alarms and SNMP Traps 14-13
DoS Protection at the Session Level 14-13
Session Based DoS Mitigation Examples 14-18
Session Based DOS Mitigation Configuration 14-20
Configuring DoS Security 14-21
Configuration Overview 14-21
Changing the Default Oracle Communications Session Border Controller Behavior 14-22
Example 1 Limiting Access to a Specific Address Prefix Range 14-22
Example 2 Classifying the Packets as Trusted 14-22
Example 3 Installing Only Static ACLs 14-23
Access Control List Configuration 14-23
Packet Loss Alarms for Access Control Lists 14-26
Packet Loss Trap for Access Control Lists 14-27
Host Access Policing 14-28
Configuring ARP Flood Protection 14-29
Access Control for a Realm 14-30
Configuring Overload Protection for Session Agents 14-32
DDoS Protection from Devices Behind a NAT 14-33
Restricting the Number of Endpoints behind a NAT 14-34
Counting Invalid Messages from Endpoints behind a NAT 14-34
DDoS Protection Configuration realm-config 14-34
DDoS Protection Configuration access-control 14-35
SNMP Trap support 14-35
Syslog Support 14-36
Debugging 14-36
DoS Threshold Configuration 14-36
Configure DOS Protection at the Session Level 14-37
Media Policing 14-38
Policing Methods 14-39
Session Media Flow Policing 14-39
Configuration Notes 14-39
Session Media Flow Policing 14-39
Static Flow Policing 14-40

Media Policing Configuration for RTP Flows 14-40
Media Policing Configuration for RTCP Flows 14-41
RTP Payload Type Mapping 14-41
ITU-T to IANA Codec Mapping 14-42
SDP Anonymization 14-42
SDP Anonymization Configuration 14-43
Unique SDP Session ID 14-43
Unique SDP Session ID Configuration 14-43
TCP Synchronize Attack Prevention 14-44
About SYN 14-44
Server Vulnerability 14-44
Configuring TCP SYN Attack Prevention 14-45
Transport Layer Security 14-45
The SBC and TLS 14-45
Supported Encryption 14-46
Diffie-Hellman Key Size 14-46
Suite B and Cipher List Support 14-46
TLS Ciphers 14-46
Minimum Advertised SSL/TLS Version 14-47
Minimum Advertised TLS Version Configuration 14-47
Signaling Support 14-47
Endpoint Authentication 14-47
Keeping Pinholes Open at the Endpoint 14-48
Key Usage Control 14-48
Key Usage List 14-49
Extended Key Usage List 14-49
4096-bit RSA Key Support 14-50
Reusing a TLS Connection 14-50
TLS Configuration Process 14-50
Certificate Configuration Process 14-50
Configure a TLS Profile 14-58
Notifications for Certificate Expiration 14-60
Configuring Notifications for Certificate Expiration 14-61
Denial of Service for TLS 14-62
DoS for TLS Configuration 14-63
TLS Session Caching 14-65
TLS Session Caching Configuration 14-66
TLS Endpoint Certificate Data Caching 14-67
Inserting Customized SIP Headers in an Outgoing INVITE 14-67
Validating the Request-URI Based on Certificate Information 14-69
TLS Endpoint Certificate Data Caching Configuration 14-71
Untrusted Connection Timeout for TCP and TLS 14-71

Caveats 14-72
Untrusted Connection Timeout Configuration for TCP and TLS 14-72
Securing Communications with TLS 14-72
Online Certificate Status Protocol 14-73
Caveats 14-73
Online Certificate Status Protocol Configuration 14-73
Enable Certificate Status Checking 14-75
Unreachable OCSP Responder 14-76
Unreachable OCSP Responder Configuration 14-76
OCSP Status Monitoring 14-77
OCSP Access via FQDN 14-77
OCSP Access Configuration via IP Address 14-78
OCSP Access Configuration via FQDN 14-78
Direct and Delegated Trust Models 14-79
Direct Trust Model Configuration 14-79
Delegated Trust Model Configuration 14-79
IPv6-IPv4 Internetworking 14-81
SRTP IPv4 IPv6 Internetworking 14-82
Supported Topologies 14-83
Configuration 14-84
Key Exchange Protocols 14-84
IKEv1 Protocol 14-84
IKEv1 Configuration 14-85
IKEv1 Global Configuration 14-85
IKEv1 Interface Configuration 14-88
IKEv1 Security Association Configuration 14-89
IPsec Security Policy Configuration 14-93
IKEv2 Protocol 14-93
IKEv2 Support 14-93
IKEv2 Interface Configuration 14-105
Secure Real-Time Transport Protocol 14-161
Protocol Overview 14-161
Licensing and Hardware Requirements 14-163
Operational Modes 14-163
Single-Ended SRTP Termination 14-163
Back-to-Back SRTP Termination 14-164
SRTP Pass-Thru 14-164
RFC 5939 Support 14-165
SDES Configuration 14-171
SDES Profile Configuration 14-171
Media Security Policy Configuration 14-172
Assign the Media Security Policy to a Realm 14-174

RFC 5939 Configuration 14-174
ACLI Example Configurations 14-174
Single-Ended SRTP Termination Configuration 14-175
Back-to-Back SRTP Termination Configuration 14-176
SRTP Pass-Thru Configuration 14-177
Security Policy 14-179
Modified ALCI Configuration Elements 14-180
Increase SSRC changes allowed in a SRTP stream 14-181
Secure Real-Time Protocol (SRTP) for Software 14-182
Protocol Overview 14-182
Operational Modes 14-184
Single-Ended SRTP Termination 14-184
Back-to-Back SRTP Termination 14-185
SRTP Pass-Thru 14-185
ACLI Instructions 14-185
Configure an SDES Profile 14-186
Media Security Policy Configuration 14-188
Assign the Media Security Policy to a Realm 14-189
ACLI Example Configurations 14-189
Single-Ended SRTP Termination Configuration 14-189
Back-to-Back SRTP Termination Configuration 14-191
SRTP Pass-Thru Configuration 14-192
Security Policy 14-194
SRTP Re-keying 14-195
SRTP Re-keying Configuration 14-196
Modified ALCI Configuration Elements 14-197
ARIA Cipher Support 14-198
Call Flow 14-198
ARIA Support Configuration 14-199
DTLS-SRTP 14-199
DTLS-SRTP Overview 14-200
SBC Support for DTLS-SRTP 14-200
Example DTLS-SRTP Flows 14-203
DTLS-SRTP Configuration 14-210
Reporting on DTLS-SRTP Statistics 14-212
Secure and Non-Secure Flows in the Same Realm 14-213
Mode Settings in the Media Security Policy 14-213
For Incoming Flows 14-213
For Outgoing Flows 14-214
Security Associations for RTP and RTCP 14-218
Security Configuration for RTP and RTCP 14-219
Supporting UAs with Different SRTP Capabilities 14-220

Receiving Offer SDP 14-220
Receiving Answer SDP 14-221
SDES Profile Configuration 14-221
Refining Interoperability 14-222
HMU Support for RTP to SRTP Interworking 14-222
Multi-system Selective SRTP Pass-through 14-224
Constraints 14-224
Operational Overview 14-225
Call Flows 14-225
Call Setup 14-226
Music on Hold 14-227
Call Transfer 14-228
Early Media 14-229
Multi-system Selective SRTP Pass-through with Media Release 14-229
Multi-system Selective SRTP Pass-through Configuration 14-229
Statistics 14-231
SRTP Re-keying 14-231
SRTP Re-keying Configuration 14-232
IPSec Support 14-233
Supported Protocols 14-233
AH vs. ESP 14-233
Tunnel Mode vs. Transport Mode 14-233
Cryptographic Algorithms 14-234
IPSec Implementation 14-234
Outbound Packet Processing 14-234
Security Policy 14-235
Fine-grained policy Selection 14-235
Security Associations 14-236
Secure Connection Details 14-236
Inbound Packet Processing 14-237
IP Header Inspection 14-237
SA Matching 14-237
Inbound Full Policy Lookup 14-238
HA Considerations 14-238
Packet Size Considerations 14-238
IPSec Application Example 14-238
IPSec Configuration 14-240
Configuring an IPSec Security Policy 14-240
Defining Outbound Fine-Grained SA Matching Criteria 14-241
Configuring an IPSec SA 14-242
Defining Criteria for Matching Traffic Selectors per SA 14-242
Defining Endpoints for IPSec Tunnel Mode 14-244

Real-Time IPSec Process Control 14-244
Key Generation 14-244
IDS Reporting 14-245
Basic Endpoint Demotion Behavior 14-245
Endpoint Demotion Reporting 14-245
SNMP Reporting 14-245
HDR Reporting 14-245
Endpoint Demotion SNMP Traps 14-246
Trusted to Untrusted Reporting 14-246
SNMP Reporting 14-246
Endpoint Demotion Trusted-to-Untrusted SNMP Trap 14-247
Endpoint Demotion Syslog Message 14-247
Event Log Notification Demotion from Trusted to Untrusted 14-247
Endpoint Demotion Configuration 14-248
Endpoint Demotion due to CAC overage 14-248
CAC Attributes used for Endpoint Demotion 14-249
Authentication Failures used for Endpoint Demotion 14-249
Endpoint Demotion Configuration on CAC Failures 14-249
IDS Phase 2 (Advanced Reporting) 14-250
Rejected SIP Calls 14-250
Rejected Calls Counter 14-250
Syslog Reporting of Rejected Calls 14-251
TCA Reporting of Denied Entries 14-252
Syslog Reporting of Denied Entries 14-253
CPU Load Limiting 14-253
Denied Endpoints 14-254
Maintenance and Troubleshooting 14-254
show sipd acls 14-254

15 R226 Security Recommendation Compliance

Bootparam Security 15-1
R226 and SIPREC License Management 15-2
SFTP Access Restrictions 15-2
SHA-2 Authentication-Password Hashing 15-2

16 Lawful Intercept

Recommendations 16-2
Interoperability Using X1 X2 X3 16-2

17 External Policy Servers

Diameter-based External Policy Servers 17-1
- Diameter Connection 17-1
- HA Support 17-2
- FQDN Support 17-2
- IPv6 Support 17-3
- Diameter Heartbeat 17-3
- Diameter Failures 17-3
- Application IDs and Modes 17-4
- Asynchronous SIP-Diameter Communication 17-5
  - SIP-Diameter Communication Configuration 17-5
- Serialized Diameter Messaging 17-6
- Flow-Description AVP for Media Release 17-6
- Load Balancing Rx Servers 17-7
  - Configuring Policy Groups and Policy Agents 17-8
  - Assigning a Policy Group to a Policy Server 17-9

IPv6 Support 17-10
- IPv6 Addresses in UTF-8 Format 17-10
- Framed-IPv6-Prefix AVP 17-10
- Bandwidth Allocation Compensation for IPv6 17-10
- Bandwidth Requests while Transcoding 17-11

Diameter: RACF 17-11
- Implementation Features 17-11
- Bandwidth Negotiation 17-13
- Session Lifetime 17-13
- Opening for RTCP Flows 17-14

RACF-only AVPs 17-14
- Diameter AAR Query Post SDP Exchange 17-14
- The Proxy Bit 17-14
- Experimental-Result-Code AVP: RACF 17-15
- Transport-Class AVP 17-15
  - Overriding Transport- Class AVP Value 17-16
  - Transport-class AVP Configuration 17-16
- Service-Info-Status AVP 17-17
- Rx-Request-Type AVP 17-18
- SIP-Forking-Indication AVP (523) 17-18

RACF and CLF AVPs 17-19
- Frame-IP-Address AVP 17-19
- 1637 - Diameter Destination Realm AVP 17-19
- Legacy Destination-Realm AVP Behavior 17-19
- Origin-Host AVP 17-20

Wildcard Transport Protocol 17-20
New Configurations and Upgrading 17-21
Configuring Diameter-based RACF 17-21
Diameter Support Realm Configuration 17-21
External Bandwidth Manager Configuration 17-22
Media Profile Configuration 17-24
Additional Diameter Compliance for the Rx Interface 17-25
Configuring the Rx Interface for SCTP 17-26
CAC Debugging 17-27
2127 - Configurable Subscription ID Types 17-28
Subscriber Information AVP 17-28
Subscription-ID AVP 17-28
Subscription ID AVP in AA-Request Message Configuration 17-30
Subscription ID AVPs in AA-Request Message Configuration 17-30
Specific Action AVP support 17-31
ACLI Examples - specific-action-subscription 17-32
Diameter STR Timeouts 17-32
Diameter STR Timeouts Configuration 17-32
Gq Interface Features 17-33
Rx Interface Features 17-33
Non-Priority Call Handling 17-33
Priority Call Handling 17-34
Rx bearer plane event 17-34
AA-Request (AAR) Command 17-35
Re-Auth-Request (RAR) Command Handling 17-35
2836 - Early Media Suppression for Rx 17-35
Media-Component-Number and Flow-Number AVP Values 17-36
Media-Component-Number AVP 17-36
Flow Examples 17-37
Media-Component AVP 3GPP Compliance Configuration 17-39
Rx Interface Reason Header Usage 17-39
Specific-Action AVP 17-39
Abort-Action AVP 17-40
Message Flows 17-40
RAR Loss-of-Bearer After Session Establishment 17-40
Network Provided Location Information 17-40
NPLI Implementation 17-41
Handling User-Endpoint-Provided Location Information 17-42
Selective INVITE Holding for NPLI 17-42
Optimizing Location Information Handling 17-44
NPLI Emergency Call Processing 17-49
Handling NPLI for Unregistered Emergency Calls 17-49

NPLI Support for 5G NR 17-54
Key AVP Support 17-59
3GPP-User-Location-Info AVP 17-59
IP-CAN-Type AVP 17-60
MSISDN AVP 17-60
RAT-Type AVP 17-61
NPLI Required-Access-Info AVP 17-61
NPLI Specific-Action AVP 17-61
Key Header Support 17-61
Extended PANI SIP Header 17-61
P-Subscription-MSISDN SIP Header 17-63
P-Access-Network-Info SIP Header 17-63
NPLI Configuration 17-64
Enabling NPLI Notifications 17-64
Configuring Default PANI Contents 17-64
Configuring the NPLI Profile 17-65
User-Endpoint-Provided Location Information Configuration 17-66
NPLI Emergency Call Configuration 17-67
Caveats 17-67
Rf Interface Features 17-68
Node-Functionality AVP Support 17-68
Node Functionality AVP Configuration 17-68
Node Functionality Per Realm Configuration 17-69
AAR Message Optimization 17-69
AAR Message Configuration 17-71
AAR Optimization with supplementary-service Configuration 17-72
AF-Application-Identifier AVP Generation 17-72
AVP Generation on Initial INVITE from UE 17-73
AVP Generation on response to Initial INVITE from UE 17-73
AVP generation on reINVITE from UE 17-74
Examples 17-74
AVP Generation on Initial INVITE from core 17-76
AVP Generation on response to initial INVITE from core 17-76
AVP generation on reINVITE from core 17-76
Diameter: CLF 17-78
CLF Behavior 17-79
P-Access-Network-Info Header Handling 17-80
P-CSCF PANI Enhancements 17-80
CLF Re-registration 17-81
CLF Failures 17-81
CLF Emergency Call Handling 17-81
CLF Diameter e2 Error Handling 17-82

Result-Code AVP 17-82
Experimental-Result-Code AVP 17-83
Diameter CLF e2 Error Bypass 17-83
CLF-only AVPs 17-83
Globally-Unique-Address AVP 17-83
e2 Interface 17-83
CLF e2 Interface User-Name AVP Support 17-84
HA Functionality 17-85
Configuring Diameter-based CLF 17-85
SIP Interface Configuration 17-85
SIP Interface Configuration for CLF Support 17-86
External Policy Server Configuration 17-86
CLF Debugging 17-89
E2 CLF Configurable Timeout 17-89
Activating a Configuration with the E2 CLF Timeout Defined 17-90
E2 CLF Timeout Configuration 17-90
Diameter Policy Server High Availability 17-90
External Policy Server HA Cluster 17-90
Standby Server Prioritization 17-91
Server States 17-91
HA Cluster Refresh (TCP Transport) 17-91
DNS Failure 17-92
Policy Server Failover 17-92
External Policy Server High Availability Configuration 17-92
Redundancy for Rx Servers over SCTP 17-93
Diameter Multi-tiered Policy Server Support 17-94
Diameter Multi-tiered Policy Server Support 17-95
Diameter Message Manipulations 17-95
Manipulation Rule 17-96
Naming Diameter Manipulations 17-96
Message Based Testing 17-96
AVP Search Value 17-97
Reserved Keywords 17-97
Actions on Found Match Value 17-97
none 17-98
add 17-98
delete 17-98
replace 17-98
store 17-98
diameter-manip 17-98
find-replace-all 17-98
group-manip 17-99

AVP Header Manipulation 17-100
AVP Flag Manipulation 17-100
vendor-id Manipulation 17-101
Multi-instance AVP Manipulation 17-102
ACLI Instructions 17-103
Diameter Manipulation 17-103
Manipulation Rule 17-103
AVP Header Manipulation 17-104
Applying the Manipulation 17-104
Diameter Manipulation Example - Supported Features AVP 17-105
COPS-based External Policy Servers 17-106
COPS Connection 17-107
COPS Failures 17-107
Failure Detection 17-107
Failure Recovery 17-107
COPS PS Connection Down 17-108
HA Support 17-108
Application Types 17-108
COPS: RACF 17-108
Implementation Features 17-109
Bandwidth Negotiation 17-110
COPS pkt-mm-3 Policy Control 17-110
Relationship to the TCP Connection 17-111
COPS Gate-Set Timeout 17-111
When a Gate-Set Times Out 17-111
COPS Decision Gate-Set Message Rejected 17-112
About the Gate-Spec Mask 17-114
COPS Debugging 17-114
COPS-based RACF Configuration 17-114
Realm Configuration 17-115
External Bandwidth Manager Configuration 17-115
Media Profile Configuration 17-117
COPS: CLF 17-117
CLF Behavior 17-118
P-Access-Network-Info Header Handling 17-118
CLF Re-registration 17-119
CLF Failures 17-119
CLF Emergency Call Handling 17-119
HA Functionality 17-120
CLF Debugging 17-120
COPS-based CLF Configuration 17-121
SIP Interface Configuration 17-121

Application Type / Interface Matrix Reference 17-123
Bandwidth Management Applications 17-123
Emergency Location Services 17-123

18 IMS Support

Oracle Communications Session Border Controller Access Border Functions 18-1
Oracle Communications Session Border Controller Interconnect Border Functions 18-2
IMS Access Border Functions 18-2
P-CSCF Functions 18-2
A-BGF Functions 18-3
Resource and Admission Control (RACS) Functions 18-3
IMS Interconnect Border Functions 18-4
Interworking Function (IWF) 18-4
Interconnect Border Control Function (I-BCF) 18-4
Interconnect-Border Gateway Function (I-BGF) 18-4
IMS Path and Service Route Header Support 18-5
Path Header 18-5
Service Route Header 18-5
Summary 18-6
IMS Path and Service Route Headers Configuration 18-7
Network Provided Location Information During Registration 18-7
Network Provided Location Information upon Register Configuration 18-13
Adding NPLI to Interim CDRs 18-14
Network Provided Location Information for Short Message Service 18-16
NPLI for Short Message Service Examples 18-17
NPLI for Short Message Configuration 18-21
Wildcard Public User Identity (PUI) 18-22
Path Header 18-22
P-Profile-Key Header 18-22
Registration Event Package 18-23
Message Flows 18-23
HA Configurations 18-25
IMS Support for Private Header Extensions for 3GPP 18-26
P-Associated-URI Header 18-26
P-Asserted-Identity Header 18-26
P-Asserted-Identity Header Handling 18-26
P-Asserted-Identity Header Configuration 18-27
P-Called-Party-ID Header 18-28
IMS Charging Headers 18-28
P-Charging-Vector 18-28
P-Charging-Vector Header Example 18-29

P-Charging-Function-Address 18-29
P-Charging-Function-Address Header Example 18-31
RADIUS Accounting of Charging Headers 18-31
P-Charging-Vector Processing for SIP Interfaces Configuration 18-31
Charging Correlation 18-34
Charging Correlation Configuration 18-34
P-Early-Media SIP Header Support 18-35
P-Visited-Network-ID Header 18-35
P-Visited-Network-ID Header Handling for SIP Interfaces Configuration 18-36
Second P-Asserted-Identity Header for Emergency Calls 18-37
Two Incoming P-Asserted-Identity Headers 18-38
Temporary Public User Identities and Multi-SIM Scenarios 18-38
Old Behavior 18-39
New Behavior 18-40
Configuring SIP Interface with reg-via-key and reg-via-match 18-41
Surrogate Registration 18-41
Integrating with IMS 18-41
Registration 18-43
Routing Calls from the IMS Core 18-44
SIP 18-44
H.323 18-44
Routing Calls from an IP-PBX 18-44
Configure Surrogate Agents 18-47
Example 18-49
SIP Surrogate Registration Enhancements 18-49
Without Enhancements 18-50
With Enhancements 18-50
Configuring the Retry Mechanism 18-50
Configuring the Count Start 18-51
SIP-IMS Surrogate Registration Proxy Authorization Header for Non-Register Requests 18-51
SIP-IMS Surrogate Registration Proxy Authorization Header Configuration 18-52
SIP Surrogate Agent Registration Re-initialization 18-53
Surrogate Agent Reregistration to SAG Member Handling 18-54
IMS Implicit Service Route 18-55
IMS Implicit Service Route Configuration 18-56
IMS Service Route Configuration 18-56
Notes About Upgrading 18-57
IMS Charging Vector Mode Adaptation 18-58
IMS Charging Vector Mode Configuration 18-58
IMS P-CSCF Endpoint Identification Using Address and Port 18-58
IMS P-CSCF Endpoint Identification Configuration 18-58
IMS-AKA 18-59

Requirements 18-59
The refreshRegForward Option 18-60
Monitoring 18-60
DDoS for IMS-AKA 18-61
ACLI Instructions and Examples 18-62
Setting Up an IMS-AKA Profile 18-62
Setting Up an IPSec Profile for IMS-AKA Use 18-63
Enabling IMS-AKA Support for a SIP Interface 18-64
Applying an IMS-AKA Profile to a SIP Port 18-64
IPSec IMS-AKA 18-65
Sample IMS-AKA Configuration 18-65
Sample Security Policy Configuration 18-65
Sec-Agree 18-67
TLS Session Setup During Registration 18-68
SEC-agree Configuration 18-71
IMS AKA over TCP 18-71
IMS-AKA Secure Call Registration over TCP 18-71
sip-ports 18-72
ims-aka-profile 18-72
IMS-AKA Call Establishment over TCP 18-73
SIP SUBSCRIBE and NOTIFY over TCP IMS-AKA 18-74
IMS-AKA Change Client Port 18-74
Protected Ports 18-75
IMS-AKA Change Client Port Configuration 18-76
Sample IMS-AKA Configuration 18-77
SIP IMS P-CSCF P-Asserted Identity in Responses 18-77
Important Notes 18-78
SIP IMS P-CSCF P-Asserted Identity in Responses Configuration 18-78
SIP IMS P-CSCF S-CSCF Target Caching and Invalidation 18-78
Acting as a B2BUA Front End to Third-Party P-CSCF 18-79
NAPTR 18-79
DNS SRV 18-80
DNS 18-80
Acting as a P-CSCF 18-80
NAPTR 18-80
DNS SRV 18-81
DNS 18-81
S-CSCF Configuration 18-81
E-CSCF Support 18-82
Service URN Support 18-82
E-CSCF Configuration Architecture 18-82
CLF Connectivity 18-83

NMC Emergency Call Control 18-83
Local Policy 18-83
Emergency LRT 18-83
CLF Response Failure 18-84
E-CSCF Configuration 18-84
Maintenance and Troubleshooting 18-86
2774 - Provisioning of SIP Signaling Flow Information 18-86
Initial Registration 18-87
Register Refresh 18-87
De-Registration 18-88
Failure Response to Re-Register 18-88
Provisioning SIP Signaling Flows Configuration 18-88
Troubleshooting 18-89
show ext-band-mgr 18-89
Subscription for Notification of Signaling Path Status 18-90
Subscription for Notification of Signaling Path Status Configuration 18-90
RTP and RTCP Bandwidth Calculation and Reporting 18-91
Max-Requested-Bandwidth-UL & Max-Requested-Bandwidth-DL AVPs 18-91
Optional AVP Creation 18-92
RR-Bandwidth & RS-Bandwidth AVPs 18-92
Flow-status AVP 18-92
2629 - IR.92 Compliance via SIP 380 Response 18-93
380 Response Format 18-93
380 Response Example 18-94
IR.92 Compliance Configuration 18-94
IR.92 Multiple Emergency Numbers 18-95
IR.92 Multiple Emergency Numbers Configuration 18-95
Emergency Calls from Unregistered Users 18-95
Emergency Calls from Unregistered Users Configuration 18-96
IR.94 Support 18-97
IR.94 Loss Of Voice Bearer 18-98
IR.94 Loss Of Voice Bearer Configuration 18-99
Dynamic Sessions Agents for Home-Remote S-CSCF Liveliness 18-99
Discovery 18-99
Creation 18-99
Property Inheritance 18-100
Deletion 18-101
How to Wildcard a Session Agent 18-101
Enabling the Global SIP Configuration for Dynamic Session Agents 18-102
SRVCC Handover Support in Alerting Phase 18-102
SIP Feature Capabilities 18-104
SIP Feature Capabilities Configuration 18-105

ATCF INVITE ICSI Matching 18-105
ATCF INVITE ICSI Matching Configuration 18-106
Enhanced eSRVCC Call Continuity 18-106
Handsets and Session Continuity 18-107
Anchors for Signaling and Media 18-107
Architectural View 18-108
IMS Registration Details 18-109
SIP Register Request UE to ATCF 18-110
SIP Register Request ATCF to S-CSCF 18-111
SIP 200 OK from S-CSCF 18-111
Originating Sessions for SRVCC with ATCF 18-112
SIP INVITE for SRVCC Using the ATCF 18-112
Terminating Sessions for SRVCC with ATCF 18-113
SIP INVITE from UE2 ATCF 18-114
TS 24.237 Proposed Changes 18-115
Accounting 18-118
External Bandwidth Management 18-118
ATCF Configuration 18-118
ATCF INVITE ICSI Matching 18-119
ATCF INVITE ICSI Matching Configuration 18-119
SRVCC PS-CS Access Transfer 18-120
MSC Server-Assisted Mid-Call Feature Supported by SCC AS 18-121
Failure and Cancellation 18-123
Confirmed Dialogs 18-124
Early Dialogs 18-124
SRVCC Handover Support in the Pre-Alerting Phase 18-125
SRVCC Handover Support in Alerting Phase 18-129
SIP Feature Capabilities 18-131
SIP Feature Capabilities Configuration 18-132
Reporting SRVCC Statistics 18-132
S8HR Roaming Compatibility 18-133
VPLMN-ID Management Support 18-136
Sec-Agree for S8HR Roaming Support 18-138
Emergency Service Support 18-139
Supporting Emergency Registration 18-140
Emergency Registrations Based on Roaming Status 18-142
Emergency INVITEs 18-145
Creating the PAI from Rx IMSI information for Emergency Calls 18-146
Use of the AF-Requested-Data AVP to Obtain EPC Identity for Emergency Calls 18-150
EPC-Level ID Processing for Registered Users 18-153
EPC-Level ID Processing for Unregistered Users 18-154
Configuring AF-Requested EPC Identities 18-155

Configuring an S8HR Profile 18-156
Applying an S8HR Profile to a SIP Interface 18-158
Configuring a PLMN INFO Change Subscription 18-158
LIR LIA Lookup to Home Subscriber Server from I-BCF 18-159
Home Subscriber Server 18-159
P-Acme-Serving Parameter Creation 18-159
HSS Watchdog Keepalives 18-159
Local Policy 18-160
Compliance for the Vendor-Specific-Application-Id 18-160
Vendor-Specific-Application-Id AVP 18-161
LIR LIA Transaction 18-161
LIR Format 18-162
LIA Format 18-163
Home Subscriber Server Configuration 18-163
Local Policy Configuration 18-164
Statistics 18-165
Emergency Access Transfer Function 18-165
Enabling EATF Capability 18-166
Monitoring SRVCC Sessions 18-166
Multimedia Priority Service for VoLTE Access 18-167
MPSIdentifier AVP (AVP 528) 18-173
Reservation-Priority AVP (468) 18-173
DRMP AVP (301) 18-174
Enabling MPS Services 18-174
Applying Network Management Control Rules to a Realm 18-174
Restoration Procedures 18-175
3GPP IM CN Subsystem XML Body 18-175
Enabling Restoration Procedures 18-176

19 SBC Processing Language (SPL)

Oracle SPL Plug-ins 19-1
Supported SPL Engines 19-1
SPL Parameter Configuration 19-2
Upload an SPL Plug-in 19-2
Add an SPL Plug-in to the Configuration 19-2
Executing SPL Files 19-3
Synchronize SPL Plug-in Files Across an HA Pair 19-3
Maintenance and Troubleshooting 19-4
show spl 19-4
show running-config spl-config 19-4
show spl-options 19-5

show directory code spl 19-5
SPL Signature State 19-5
SPL Log Types 19-5
Delete SPL Plugin Files 19-5
Service Provider SPLs 19-6
Universal Call Identifier SPL 19-6
UCID-App-ID 19-6
GUCID-Node-ID 19-7
GUID-Node-ID 19-7
GenUUID-App-ID 19-7
convert-to 19-7
Example SPL Options 19-8
Sample Metadata 19-8
Configuring Universal Call Identifier Options 19-8
Inserting SIP Headers into SIPREC Metadata 19-9
Sample Metadata 19-9
Configure SIP Headers for SIPREC Metadata 19-11
Configure LRE for SIPREC Metadata 19-11
SBC Deployment Behind a NAT Device 19-12
Configure the SBC Behind a NAT Device Option 19-15
SIP Trunking SPLs for Specific Service Providers 19-16
Surrogate Registration 19-16
Excluding the Random Contact Validation from the Surrogate Registration SPL Feature 19-16
Configure Surrogate Registration 19-17
Configure Surrogate Registration for KDDI Environments 19-19
NTT Message Converter 19-19
Emergency Location Identification Number (ELIN) Gateway Support 19-21
How the Emergency Location Identification Number (ELIN) SPL Works 19-21
PSAP Callback Options 19-22
Configure the ELIN Gateway Options 19-27
Comfort Noise Generation SPL 19-28
Configure the Comfort Noise Generation SPL 19-30
Example Configuration 19-31
High Availability (HA) Support 19-31
Licensing Information 19-31
Request Validation SPL 19-32
Validating the Request-URIs in INVITEs and re-INVITEs 19-33
Validating the Request-URIs in PRACKs, CANCELs, ACKs and BYEs 19-34
Validating the Via Header in INVITEs and re-INVITEs 19-34
Suppression of Extraneous 18x Messages 19-35

Call Flows for 18x Suppression 19-36

20 Transcoding

Transcoding Resources 20-1
Hardware-based Transcoding Resources 20-1
Transcodable Codecs 20-2
Transcodable Codec Details 20-2
T.38 FAX Support 20-3
Software-based transcoding 20-3
Software-based transcoding alarms and traps 20-4
Adaptive Jitter Buffers for Transcoding Flows on vSBCs 20-4
PCIe Transcoding Accelerator Cards 20-6
Transcoding Processing Overview 20-7
Unoffered Codec Reordering 20-7
Non-transcoded Call 20-8
Transcoded Call 20-8
Post Processing 20-8
Voice Transcoding 20-8
Voice Scenario 1 20-8
Voice Scenario 2 20-11
Voice Scenario 3 20-13
RFC 2833 Transcoding 20-14
RFC 2833 Scenario 1 20-15
RFC 2833 Scenario 2 20-17
FAX Transcoding 20-18
Defining G711FB 20-19
FAX Scenario 1 20-19
FAX Scenario 2 20-20
FAX Scenario 3 20-21
Transrating 20-23
Transrating Scenario 1 20-23
Reactive Transcoding 20-25
Reactive Transcoding Examples 20-27
Reactive Transcoding Mode Configuration 20-29
Transcoding Configuration 20-30
Codec Policy Configuration 20-30
Terms Used in Codec Policies 20-30
Ingress Policy 20-30
Egress Policy 20-31
Post Processing 20-32
allow-codecs 20-32

order-codecs 20-33
Add on Egress 20-33
Packetization Time 20-33
Name a Codec Policy 20-34
Order Codecs 20-34
Allow, Remove, and Add Codecs for Transcoding 20-35
Configure a Codec Policy 20-36
Configure Transrating 20-37
Apply a Codec Policy to a Realm 20-37
Secure DTMF Cancellation 20-38
Enable Secure DTMF Cancellation 20-39
Default Media Profiles 20-39
Preferred Default Payload Type 20-39
Redefining Codec Packetization Time 20-40
mptime Support for Packet Cable 20-40
Media Profile Configuration 20-41
Media Type Subnames 20-41
SDP Parameter Matching 20-41
Using Subnames with Codec Policies 20-41
Subname Syntax With the Wildcard Character 20-42
Wildcard Character in add-codecs-on-egress Limitation 20-43
Configure Media Type and Subname 20-43
Configure a Codec Policy with a Media Type with a Subname 20-44
Updating the b=AS line for Transcoded Calls 20-44
Codec and Conditional Codec Policies for SIP 20-45
Codecs in Relationship to Media Profiles 20-46
Manipulation Modes in Codec Policies 20-46
In-Realm Codec Manipulation 20-48
Conditional Codec Policies 20-48
Conditional Codec Lists 20-49
Conditional Codec Operators 20-50
Codec Policies Instructions and Examples 20-51
Create a Codec Policy 20-51
Apply a Codec Policy to a Realm 20-52
Apply a Codec Policy to a Session Agent 20-52
In-Realm Codec Manipulations 20-53
Pooled Transcoding 20-53
Supported Codecs for Pooled Transcoding 20-55
Hardware and Software Requirements 20-55
Implementation Details 20-55
Scenario 1 INVITE with SDP 20-56
Scenario 2 INVITE without SDP 20-57

Re-INVITES and Updates with SDP 20-58
RFC 2833 Considerations 20-58
eSRVCC Support 20-59
Configuration Requirements and Verification 20-60
A-SBC Configuration Requirements 20-60
T-SBC Requirements 20-61
Configuration Verification 20-61
Configure Pooled Transcoding 20-61
Monitor Dialogs Between the A-SBC and the T-SBC 20-62
Per-Method Statistics 20-63
Pooled Transcoding and MS Teams 20-63
Configure Pooled Transcoding for MS Teams 20-64
Notes on the DIAMETER Rx Interface 20-65
Accounting and Transcoding 20-66
EVRC Family of Codecs 20-66
EVRC-A Codec for Transcoding 20-66
EVRC0 Supported Options 20-67
EVRC Supported Options 20-67
EVRC1 Supported Options 20-67
Default settings for EVRC encoding 20-68
EVRC Configuration 20-68
EVRC-B Codec for Transcoding 20-68
EVRCB0 Supported Options 20-68
EVRCB Supported Options 20-69
EVRCB1 Supported Options 20-69
Default fixed settings for EVRCB encoding 20-69
EVRCB Configuration 20-69
Opus Codec Transcoding Support 20-70
SILK Codec Transcoding Support 20-72
Comfort Noise Transcoding 20-73
System Behavior Without Comfort Noise Transcoding 20-74
System Behavior With Comfort Noise Transcoding 20-75
T.140 to Baudot Relay 20-78
AMR-NB and AMR-WB Specifications 20-81
AMR AMR-WB Payload Type Mapping 20-82
AMR AMR-WB octet-align Parameter 20-82
AMR AMR-WB mode-set Parameter 20-83
Other AMR AMR-WB Parameters 20-83
Examples and Explanations 20-83
rtpmap Attribute 20-84
fmtp Attribute 20-84
Basic Scenarios 20-85

Advanced Scenarios 20-89
ACLI Configuration 20-94
EVS Codec Transcoding Support 20-94
EVS Configuration Detail 20-99
AMR-WB to EVS AMR-WB IO Transparent Call Example 20-100
EVS and SRVCC 20-101
Asymmetric Dynamic Payload Types Enablement 20-102
Configurations for non-Standard PT Cases 20-103
Configure Transcoding for Asymmetric Dynamic Payload Types 20-108
Asymmetric Payload Type Support for RFC2833 Interworking 20-109
Separate Clock Rates for Audio and Telephone Events 20-111
Call Flows for Differing Tel-event and Codec Clock Rates 20-112
Call Flows for Differing Egress Clock Rates 20-115
Supporting Different Codec and Telephone-Event Rates in the SDP 20-121
Simultaneous Payload Type Mapping for Audio and DTMF 20-121
ACLI Configuration 20-122
DTMF Indication over HD Audio Codecs 20-122
FAX Detection 20-123
Supporting FAX to UAs that Do Not Support Multiple SDP M-Lines 20-127
FAX Detection and Redirect 20-129
Call Flows for Fax with Redirect 20-133
Configure Fax Detect and Redirect 20-137
Configure reINVITE and Single M Line Behavior for FAX Calls 20-138
Maintenance and Troubleshooting 20-139
show mbcd errors 20-139
show xcode api-stats 20-140
show xcode dbginfo 20-140
Active and Period Statistics for EVRC and other Codecs 20-141
show sipd codecs 20-141
Session Based Statistics 20-141
Flow Based Statistics 20-141
Single audio stream example 20-142
Multiple audio stream example 20-142
Transcoded audio stream example 20-142
show xcode load 20-143
show xcode session-all 20-144
show xcode session-byid 20-144
show xcode session-byattr 20-147
show xcode session-byipp 20-147
show xcode xlist 20-148
Logs 20-148
Alarms 20-148

Transcoding Capacity Traps 20-150
SRTP and Transcoding 20-152
Generating RTCP 20-152
RTCP Generation Platform Support 20-153
Configuring RTCP Generation 20-154
Obtaining System Information about RTCP Generation 20-155
Forced RTCP Receiver Report Generation 20-155
Generate an RTCP Receiver Report 20-156
SNMP 20-156
Acme Packet Codec and Transcoding MIB (ap-codec.mib) 20-157
Acme Packet System Management MIB (ap-smgmt.mib) 20-160
Acme Packet 6300 NIU Hotswap Guidelines 20-161
Network Interface Unit Removal/Replacement -- Standalone Node 20-161
NIU Removal/Replacement -- High Availability Deployment 20-162
Minimum TCM3 Versions on the Acme Packet 3950/4900 20-162

21 DTMF Interworking

DTMF Indication 21-1
RFC 2833 telephone-event 21-1
SIP INFO Messages 21-2
DTMF Transfer Processing Overview 21-2
Capability Negotiation 21-2
SDP Manipulated by Codec Policy 21-3
telephone-event Modification by Codec Policy 21-3
SDP Manipulated by RFC 2833 Mode 21-4
Transparent RFC 2833 Support 21-4
Preferred RFC 2883 Support 21-5
RFC 2833 Payload Type Mapping 21-6
Translation Evaluation 21-6
RFC 2833 Sent by Offerer 21-7
RFC 2833 to RFC 2833 21-7
RFC 2833 to DTMF Audio Tones 21-7
RFC 2833 to SIP INFO 21-8
DTMF Audio Tones Sent by Offerer 21-8
DTMF Audio to DTMF Audio 21-8
DTMF Audio to RFC 2833 21-9
DTMF Audio to SIP 21-9
SIP INFO Sent By Offerer 21-10
SIP INFO to RFC 2833 21-10
SIP INFO to DTMF Audio 21-10
SIP INFO to SIP INFO 21-11

Dual Mode 21-11
P-Dual-Info Header 21-12
Example 1 21-12
Example 2 21-12
DTMF Transfer for Spiral Calls 21-13
P-Dual-Info Header 21-13
DTMF Transfer Hardware Processing 21-14
DTMF Transfer Configuration 21-14
RFC 2833 Session Agent Configuration 21-14
ACLI Configuration and Instructions 21-15
SIP Interface 21-15
Session Agent 21-15
Codec Policy 21-16
Translate Non2833 Event Behavior 21-17
P-dual-info Header Appearance 21-17
RFC 2833 Customization 21-18
RTP Timestamp 21-18
RFC 2833 telephone-event duration intervals 21-18
RFC 2833 End Packets 21-19
ACLI Instructions and Examples 21-19
RFC2833 and KPML Interworking 21-20
Interworking RFC 2833 and KPML for Hairpin Sessions 21-24
KPML-2833 Interworking on a SIP Interface Configuration 21-27
KPML-2833 Interworking on a Session Agent Configuration 21-27

22 RCS Services

Message Session Relay Protocol 22-1
MSRP Platform Support 22-1
MSRP IP Address Family Support 22-1
MSRP Operational Description 22-2
Secure MSRP Session Negotiation 22-5
MSRP Session Setup 22-6
Initiating MSRP Sessions 22-6
Connection Negotiation 22-6
Specifying the Connection Delay Timer 22-9
Multiple MSRP Connections 22-10
Accepting Connections 22-10
Making Connections 22-10
MSRP Session Termination 22-10
Network Address Translation 22-12
Double Steering Pool Port Allocation 22-12

Certificate Fingerprint 22-13
MSRP B2BUA Support for NG911 22-14
MSRP and Middlebox Traversal Using the CEMA Extension and Session-ID 22-15
MSRP Media Types Filtering 22-19
MSRP Message Size Limiting 22-19
MSRP and High Availability 22-20
MSRP Configuration 22-20
msrp-config Configuration 22-20
Configure tcp-media-profile 22-23
Assign a tcp-media-profile to a Realm 22-25
tls-profile Configuration 22-26
MSRP Statistics 22-26
MSRP Management Monitoring 22-27
Extended MSRP Statistics 22-28
MSRP Session Limitations Based on Entitlements 22-31
RCSe TLS/TCP Re-Use Connections 22-34
RCSE TLS/TCP Re-Use Connections Configuration 22-35

23 Local Media Playback

Local Media Playback Operation 23-2
Supported Local Media Triggers 23-3
Media Files 23-4
Media Setup and Playback 23-4
The P-Acme-Playback Header 23-5
Supported Playback Scenarios 23-6
Playback on 180-no-sdp 23-7
Playback on 180-force 23-8
Playback on REFER 23-9
Playback Header 23-10
Playback on 183 Session Progress 23-11
Media Spirals 23-12
Transcoding Free Operation for Local Media Playback 23-13
RBT TrFO Flows 23-15
RBT TrFO Reporting 23-18
Considerations for HA Nodes 23-18
RTC Support 23-18
Alarms 23-18
Monitoring 23-19
RBT TrFO Configuration 23-19
Configuring Local Media Playback with Transcoding Resources 23-20

24 Advanced Media Termination Support

Advanced Media Termination Operations in the Network 24-2
Additional STUN Candidate for RTCP 24-3
Advanced Media Termination Configuration Process 24-6
- Configure ICE Profile 24-6
- Configure Advanced Media Termination in Realm Config 24-7

Advanced Media Termination Troubleshooting 24-8

25 STIR/SHAKEN Client

HTTP Client Operating Modes 25-1
- Operating Under the ATIS Mode 25-2
- STIR/SHAKEN Client Operation 25-2
- Operating Under the 3GPP Mode 25-8
  - Global 3GPP Behaviors 25-9
  - 3GPP Behaviors Related to Configuration 25-11
  - 3GPP Behaviors Related to Errors 25-14
  - Extract verifyResults Content for DIV Passports for 3GPP 25-16

Number Authentication Mechanism Standards Compliance Features 25-17
- Changing the STI-VS Trigger 25-17
- Including STI-VS Failure Reasons in INVITEs 25-18
- CDR Compliance for Number Authentication Mechanism 25-21
  - CDR Behavior Changes 25-21
  - Additional CDR Data 25-24

STIR/SHAKEN Servers 25-27
- Server Names as FQDNs 25-27
- Load Balancing STIR/SHAKEN Servers 25-29
- STI Server Heartbeat 25-29
  - Example STI Server Heartbeat Requests 25-32
- Per Call Availability of STI Servers 25-34
- Supporting HA with STIR SHAKEN over TCP 25-37
- Configuring the STIR/SHAKEN Client 25-38
  - Configure the Authentication Profile 25-39
  - Configure the HTTP Client 25-39
  - Configure the STI Config 25-40
  - Configure the STI Servers 25-42
  - Configure STI Server Groups 25-44
  - Configure STIR/SHAKEN REST Client Functionality on a Session Agent 25-44

<Enter Concept Title Here> 25-45
- Call Rejection Enhancement 25-46
- Changing the Precedence for Handling orig and verstat Values 25-49

Creating a Reason Header During Verification 25-51
Enhanced Reason Header Configuration 25-53
Including CALEA in Authentication Requests 25-56
Rejecting Calls During Verification 25-57
Creating Detailed Call Rejection Information 25-60
Mapping SIP to HTTP Headers and Parameters 25-65
Mapping SIP to HTTP Headers 25-66
Mapping SIP to HTTP Parameters 25-69
Parameter Mapping Configuration Examples 25-72
Bypass STI-AS Signature and STI-VS Verification Requests 25-75
Remove the verstat from Messages to the Caller 25-75
Configure STIR/SHAKEN Functions 25-76
Configure Header and Parameter Mapping 25-76
Configure the sti-reason-header-config 25-77
Configure the sti-response-treatment-config 25-78
Additional Administrative Features 25-80
Accepting Non-Standard verstat 25-80
Media Policy for the HTTP Client Element 25-81
TLS Session Key Logging 25-81
Managing Curl within HTTP Server Interactions 25-81
Logging Compliance for Number Authentication Mechanism 25-82
Commands for Testing STIR Deployments 25-83
Configure test-stir 25-86
<Enter Concept Title Here> 25-87
STIR/SHAKEN Client Statistics 25-87
STIR/SHAKEN Statistics in the ACLI 25-89
STIR/SHAKEN Statistics in SNMP 25-91
STIR/SHAKEN Statistics in HDR 25-94
STIR/SHAKEN Information in CDRs 25-95
STIR/SHAKEN Client Alarms, Traps and Logs 25-98

A RTC Support

B SIP Compatibility Options

LMSD SIP Call Progress Tone Interworking B-1
LMSD Interworking Configuration B-1
Ensuring Telephone Event Negotiation within Delayed Media and Conflicting Configurations B-2
Accommodating m=line Omissions During Call Setup B-2
SIP re-INVITE Suppression B-3
SIP re-INVITE Suppression Configuration B-3

About this Guide

The ACLI Configuration Guide provides information about:

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

Documentation Set

The following table describes the documentation set for this release:

Document Name	Document Description
Acme Packet 3900 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 3900.
Acme Packet 4600 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 4600.
Acme Packet 4900 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 3950 and Acme Packet 4900.
Acme Packet 6350 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 6350.
Release Notes	Contains information about the current documentation set release, including new features and management changes.
Known Issues & Caveats	Contains known issues and caveats
Configuration Guide	Contains information about the administration and software configuration of the Service Provider Session Border Controller (SBC).
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 SBC logs, performance announcements, system management, inventory management, upgrades, working with configurations, and managing backups and archives.
MIB 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.

Document Name	Document Description
Accounting Guide	Contains information about the SBC's accounting support, including details about RADIUS and Diameter accounting.
HDR Guide	Contains information about the SBC's Historical Data Recording (HDR) feature. This guide includes HDR configuration and system-wide statistical information.
Admin Security Guide	Contains information about the SBC'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 SBC family of products.
Platform Preparation and Installation 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.
HMR Guide	Contains information about configuring and using Header Manipulation Rules to manage service traffic.
REST API	Contains information about the supported REST APIs and how to use the REST API interface.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

My Oracle Support

My Oracle Support (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support (CAS) can assist you with My Oracle Support registration.

Call the CAS main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. When calling, make the selections in the sequence shown below on the Support telephone menu:

1. Select `2` for New Service Request.
2. Select `3` for Hardware, Networking, and Solaris Operating System Support.
3. Select one of the following options:
   - For technical issues such as creating a new Service Request (SR), select `1`.
   - For non-technical issues such as registration or assistance with My Oracle Support, select `2`.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

Emergency Response

In the event of a critical service situation, emergency response is offered by the Customer Access Support (CAS) main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. The emergency response provides immediate coverage, automatic escalation, and other features to ensure that the critical situation is resolved as rapidly as possible.

A critical situation is defined as a problem with the installed equipment that severely affects service, traffic, or maintenance capabilities, and requires immediate corrective action. Critical situations affect service and/or system operation resulting in one or several of these situations:

- A total system failure that results in loss of all transaction processing capability
- Significant reduction in system capacity or traffic handling capability
- Loss of the system's ability to perform automatic system reconfiguration
- Inability to restart a processor or the system
- Corruption of system databases that requires service affecting corrective actions
- Loss of access for maintenance or recovery operations
- Loss of the system ability to provide any required critical or major trouble notification

Any other problem severely affecting service, capacity/traffic, billing, and maintenance capabilities may be defined as critical by prior discussion and agreement with Oracle.

Locate Product Documentation on the Oracle Help Center Site

Oracle Communications customer documentation is available on the web at the Oracle Help Center (OHC) site, http://docs.oracle.com. You do not have to register to access these documents. Viewing these files requires Adobe Acrobat Reader, which can be downloaded at http://www.adobe.com.

1. Access the Oracle Help Center site at http://docs.oracle.com.
2. Click **Industries**.
3. Under the Oracle Communications sub-header, click the **Oracle Communications documentation** link.
   The Communications Documentation page appears. Most products covered by these documentation sets appear under the headings "Network Session Delivery and Control Infrastructure" or "Platforms."
4. Click on your Product and then Release Number.
   A list of the entire documentation set for the selected product and release appears.
5. To download a file to your location, right-click the **PDF** link, select **Save target as** (or similar command based on your browser), and save to a local folder.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Revision History

The following table shows the dates and descriptions of revisions to the S-Cz9.2.0 ACLI Configuration Guide.

Date	Description
March 2025	• Initial release.

1

Oracle Communications Session Border Controller Basics

This chapter introduces some basic concepts that apply to your Oracle Communications Session Border Controller (SBC). Understanding these concepts is necessary to configure your SBC. This chapter provides only a high-level overview of some important SBC concepts. Specific configuration instructions are found in later chapters.

What Is a Realm

A realm is a logical way of identifying a domain, a network, a collection of networks, or a set of addresses. Realms are used when an Oracle Communications Session Border Controller (SBC) communicates with multiple network elements over a shared connection. Defining realms allows flows to pass through a connection point between two networks.

From an external perspective, a realm is a collection of systems that generates real-time sessions comprised of signaling messages and media flows, or a group of multiple networks containing these systems. These systems may be session agents such as call agents, softswtiches, SIP proxies, H.323 gatekeepers, IP PBXs, etc., that can be defined by IP addresses. These systems can also be IP endpoints such as SIP phones, IADs, MTAs, media gateways, etc.

From an internal perspective, a realm is associated with SBC configurations to define interfaces and resources in a logical way. Realms are used to support policies that control the collection of information and statistics from systems or networks that generate media sessions. Realms are referenced by other configuration elements and the SBC uses realms to make routing decisions.

Nested Realms

Nested Realms is a Oracle Communications Session Border Controller feature that supports hierarchical realm groups. One or more realms may be nested within higher order realms. Realms and sub-realms may be created for media and bandwidth management purposes. This feature supports:

- Separation of signaling & media on unique network interfaces
- Signaling channel aggregation for Hosted IP Services applications
- Configuration scalability
- Per-realm media scalability beyond single **phy-interface** capacity
- Nested bandwidth admission control policies

What Is a Session Agent

A session agent defines an internal signaling endpoint. It is an internal next hop signaling entity that applies traffic shaping attributes to flows. For each session agent, concurrent session capacity and rate attributes can be defined. Service elements such as gateways, softswitches,

and gatekeepers are defined automatically within the Oracle Communications Session Border Controller as session agents. The Oracle Communications Session Border Controller can also provide load balancing across the defined session agents.

SIP session agents

SIP session agents can include the following:

- Softswitches
- SIP proxies
- Application servers
- SIP gateways

H.323 session agents

H.323 session agents can include the following:

- gatekeepers
- gateways
- MCUs

Why You Need Session Agents

You can use session agents to describe next or previous hops. You can also define and identify preferred carriers to use for traffic coming from session agents. This set of carriers is matched against the local policy for requests coming from the session agent. Constraints can also be set for specific hops.

In addition to functioning as a logical next hop for a signaling message, session agents can provide information regarding next hops or previous hops for SIP packets, including providing a list of equivalent next hops.

How to Use Session Agents

You can use session agents and session agent groups (along with local policies) to define session routing for SIP and H.323 traffic. You can associate a realm with a session agent to identify the realm for sessions coming from or going to the session agent.

What is a Session Agent Group

A session agent group contains individual session agents bundled together. A SAG indicates that its members are logically equivalent and can be used interchangeably. This allows for the creation of constructs like hunt groups for application servers or gateways. Session agent groups also assist in load balancing among session agents.

Session agent groups can be logically equivalent to the following:

- Application server cluster
- Media gateway cluster
- Softswitch redundant pair
- SIP proxy redundant pair

- Gatekeeper redundant pair

High Availability

High Availability (HA) is a network configuration used to ensure that planned and unplanned outages do not disrupt service. In an HA configuration, Oracle Communications Session Border Controllers (SBC) are deployed in a pair to deliver continuous high availability for interactive communication services. Two SBCs operating in this way are called an HA node. The HA node design ensures that no stable call is dropped in the event of an outage.

In an HA node, one SBC operates in the active mode and the other SBC operates in the standby mode.

- Active. The active member of the HA node is the system actively processing signal and media traffic. The active member continuously monitors itself for internal processes and IP connectivity health. If the active member detects a condition that can interrupt or degrade service, it hands over its role as the active member of the HA node to the standby member.
- Standby. The standby member of the HA node is the backup system. The standby member is fully synchronized with the active member's session status, but it does not actively process signal and media traffic. The standby member monitors the status of the active member and it can assume the active role without the active system having to instruct it to do so. When the standby system assumes the active role, it notifies network management using an SNMP trap.

The SBC establishes active and standby roles in the following ways.

- If an SBC boots up and is alone in the network, it is automatically the active system. If you pair a second SBC with the first one to form an HA node, the second system automatically establishes itself as the standby.
- If both SBCs in the HA node boot up at the same time, they negotiate with each other for the active role. If both systems have perfect health, then the SBC with the lowest HA rear interface IPv4 address becomes the active SBC. The SBC with the higher HA rear interface IPv4 address becomes the standby SBC.

If the rear physical link between the two SBCs is unresponsive during boot up or operation, both will attempt to become the active SBC. In this circumstance, processing does not work properly.

The standby SBC assumes the active role when:

- it does not receive a checkpoint message from the active SBC for a certain period of time.
- it determines that the active SBC health score declined to an unacceptable level.
- the active SBC relinquishes the active role.

To produce a seamless switch over from one SBC to the other, the HA node members share their virtual MAC and virtual IP addresses for the media interfaces in a way that is similar to Virtual Router Redundancy Protocol (VRRP). Sharing these addresses eliminates the possibility that the MAC address and the IPv4 address set on one SBC in an HA node will be a single point of failure. Within the HA node, the SBCs advertise their current state and health to one another in checkpointing messages to apprise each one of the other one's status. Using the Oracle HA protocol, the SBCs communicate with UDP messages sent out and received on the rear interfaces. During a switch over, the standby SBC sends out an ARP request using the virtual MAC address to establish that MAC address on another physical port within the Ethernet switch. To the upstream router, the MAC address and IP address are still alive. Existing sessions continue uninterrupted.

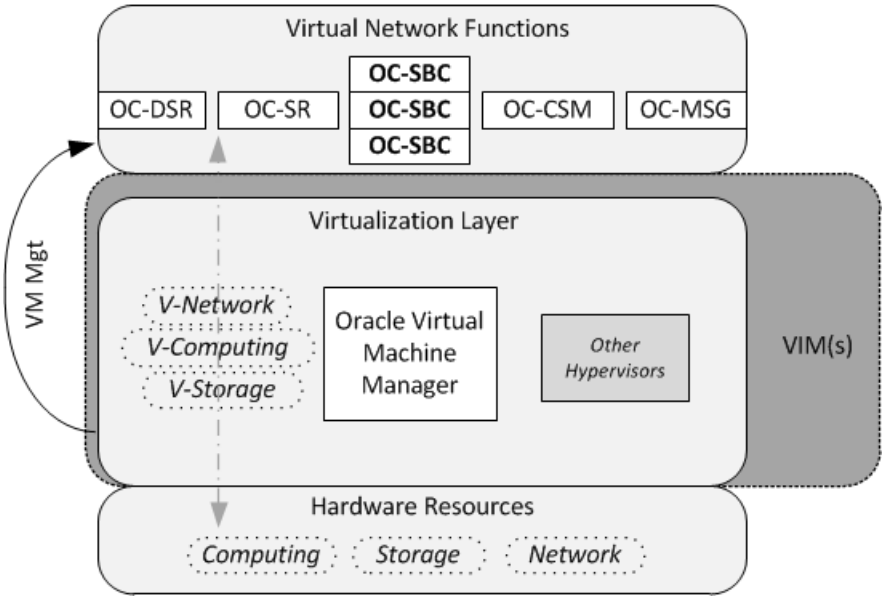
Network Functions Virtualization Overview

Network Functions Virtualization (NFV) is a set of architectural standards to leverage enterprise and cloud technologies for virtualization and lifecycle resource automation in telecommunications core networks. It is focused on identifying the use cases and needs of communications service providers for automated, virtualized infrastructure and network functions, along with an architecture to frame discussions on meeting those needs, and an analysis of the gaps with existing technologies. Standards defining the NFV architecture addressed in this document include "ETSI GS NFV 002 Architectural Framework", ETSI GS NFV 004 "Virtualization Requirements" and related. Oracle can provide its SBC and Session Router, as well as other network systems, as Virtual Network Functions (VNFs) or Management Systems for operation within an NFV-enabled environment.

One of the main drivers for NFV is capital equipment cost reduction for CSPs. Virtual functions running on fewer COTS servers is much cheaper than using multiple purpose built appliances. NFV meets the needs for cost reduction in addition to allowing the CSP to turn up new services and capacity more rapidly.

It is generally recognized now that operators desire to move their core network functions into automated and virtualized environments based on these technologies, starting with large operators in North America, Europe, and Asia. To accomplish this requires changes in the infrastructure that hosts the network functions, changes in the network functions, and additions to the management and automation of the life cycle of network services.

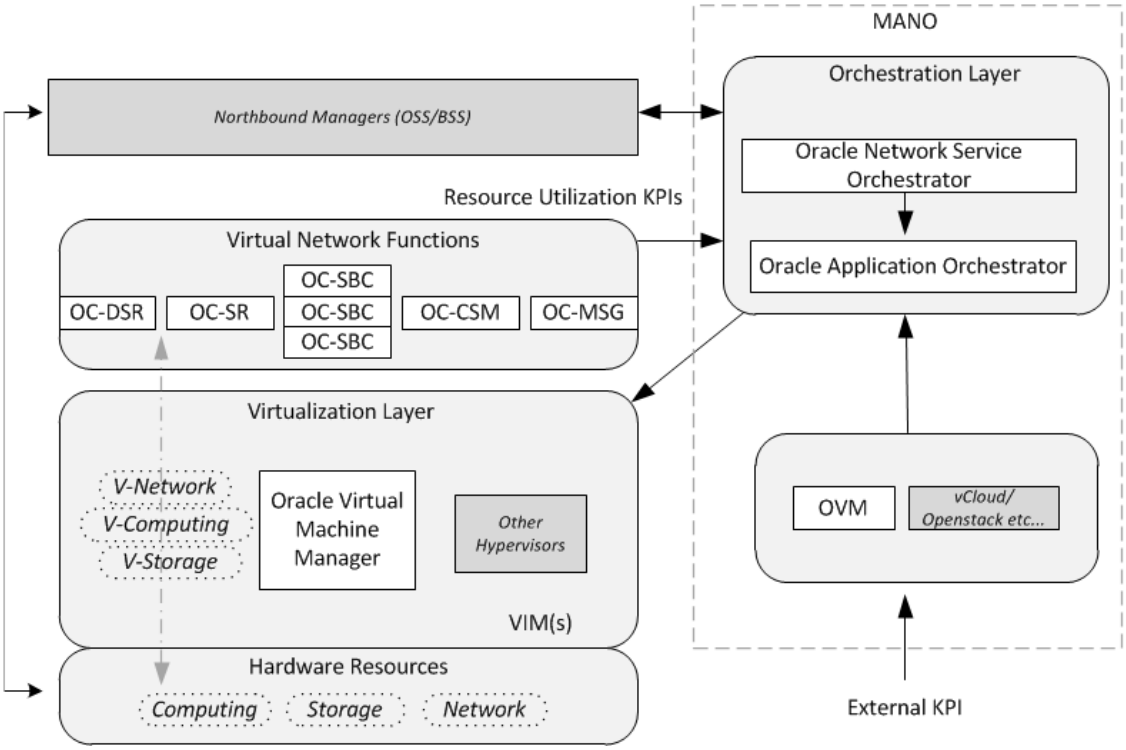
In the NFV Industry Specification Group under ETSI, which is where the bulk of industry focus on defining the use cases and architecture for NFV have been, an architectural framework, shown below, has been defined that is the basis for most discussions of the various functional components needed for an NFV environment. These components are divided into the Hardware, the VNFs and the virtualization layer. Commodity hardware provides the physical components for computing, and are the same as those without virtualization. The virtualization layer provides the means through which virtual machines can operate. The VNFs are the virtual machines themselves, instantiated as Virtual Network Functions (VNFs). These components establish a VNF deployment that can be fully operational, but without orchestration.



Extending upon these basic operational components are the Management and Orchestration components (MANO) components. MANO provides a means by which operators approach a

zero-touch orchestration, operation and management deployment. The resulting automated, elastic construct more fully addresses the OPEX efficiencies and real-time responsiveness desired from NFV.

To complete the NFV picture, MANO components are included in the diagram below, consisting of the orchestration layer and VIM components associated with managing your NFV. This diagram also adds the northbound OSS/BSS systems. The orchestration layer supports NFV instantiating and de-commissioning based on operational requirements and status, as well as on business requirements and preferences. The MANO model also provides for basic FCAPS management, encapsulating those management functions outside of orchestration.



Network operators establish criteria for automated VNF instantiation within the orchestration layer, by configuring the overall system to use external and internal status within KPIs to define the triggers for changes to VNF resources. VNFs themselves provide critical status information that, when combined with organizational policy and other status triggers, the MANO uses to adjust network resources to network requirements in real time.

Oracle's NFV solution harnesses the extensive assets from its portfolio to deliver a vision of a network orchestration being driven by a business orchestration, which in turn delivers business value to the customer. This vision builds on the investment that Oracle Communications has made in the BSS/OSS domain to deliver a BSS/OSS that is orchestrated along business processes. The Oracle Communication Network Service Orchestrator bridges Network Orchestration and Business Orchestration.

Oracle's NFV solution builds on the investment Oracle has made in the network domain to have an extensive portfolio of Network Functions as well as network orchestration.

2

Getting Started

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

This chapter offers information that will help you:

- Review hardware installation procedures
- Connect to your Oracle Communications Session Border Controller using a console connection or SSH (secure shell)
- Become familiar with the Oracle Communications Session Border Controller's boot parameters and how to change them if needed
- Set up product-type, features, and functionality
- Load and activate a Oracle Communications Session Border Controller software image
- Choose a configuration mechanism: ACLI, Oracle Communcations Session Element Manager or ACP/XML
- Enable RADIUS authentication
- Customize your login banner

Installation and Start-Up

After you have completed the hardware installation procedures outlined in the the relevant *Hardware Installation Guide*, you are ready to establish a connection to your Oracle Communications Session Border Controller. Then you can load the software image you want to use and establish basic operating parameters.

Hardware Installation Process

Installing the Oracle Communications Session Border Controller hardware in a rack requires the following process.

1. Unpack the Oracle Communications Session Border Controller hardware.
2. Install the Oracle Communications Session Border Controller hardware into the rack.
3. Install the power supplies.
4. Install the fan modules.
5. Install the physical interface cards.
6. Cable the Oracle Communications Session Border Controller hardware.

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

For more information, see the hardware documentation.

Connecting to Your Oracle Communications Session Border Controller

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

> **Note:**
>
> By default, SSH and SFTP connections to your Oracle Communications Session Border Controller are enabled.

Create a Console Connection

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

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

To make a console connection to your hardware:

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

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

SSH Remote Connections

Connect to the Oracle Communications Session Border Controller (SBC) using SSH. The SBC supports five concurrent SSH and SFTP sessions. Only one SSH session may be in configuration mode at a time.

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

You can manage incoming SSH connections from the ACLI:

- SSH service is enabled by default.
- To view the users who are currently logged into the system, use the ACLI **show users** command. You can see the ID, timestamp, connection source, and privilege level for active connections.
- From Superuser mode in the ACLI, you can terminate the connections of other users in order to free up connections. Use the **kill <sftp | ssh | web>** command with the corresponding connection ID.
- If you reboot your SBC from a SSH session, you lose IP access and therefore your connection.

There are two ways to use SSH to connect to the SBC. Either connect via SSH without specifying users and SSH user passwords, or initiate the SSH connection using custom SSH credentials.

> Old SSH and SFTP clients that use weak ciphers may not be able to connect to the SBC. If a verbose connection log shows the server and client cannot agree on a cipher, upgrade your client.

Accessing the System Via User and Admin Accounts

You may access the Oracle Communications Session Border Controller via SSH connection without specifying users and SSH user passwords.

1. Open your SSH client (with an open source client, etc.).
2. At the prompt in the SSH client, type the **ssh** command, a Space, the IPv4 address of your Oracle Communications Session Border Controller, and then press Enter.
   The SSH client prompts you for a password before connecting to the Oracle Communications Session Border Controller. Enter the Oracle Communications Session Border Controller's User mode password. After it is authenticated, an SSH session is initiated and you can continue with tasks in User mode or enable Superuser mode.

Manage SSH Keys

Use the **ssh-key** command to manage SSH keys for the SBC.

Add an SSH Authorized Key

To authenticate to the SBC using public key authentication rather than a password, use the **ssh-key** command with the **authorized-key import** argument.

1. On the SSH client, convert the public key of the SSH client into RFC 4716 format.

> **Note:**
>
> Valid RSA key sizes are 2048, 3072, or 4096 bytes. The only valid DSA key size is 1024 bytes.

To do this on Oracle Linux, use the **ssh-keygen** command.

```
[bob@client ~]$ ssh-keygen -e -f .ssh/id_rsa.pub
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "4096-bit RSA, converted by bob@client from OpenSSH"
AAAAB3NzaC1yc2EAAAADAQABAAACAQDOTDujYoQXzjTt9I8YvJMvfSVlWZ6iDzfRx06R3l
Rj/lrjxlWDMc/Y/uEd2sJ+5wdlCnJPREOuCGbU8S6295486D1kbu76cEDxE+adca3/9+qo
7FQVugkRJBD0ZOj/3qcuKDOh6ZsalF9LaaNMPNWNiQ5n3bWBnQ1tMMEes58JvoNgjn9FOz
hbOdOe91K/OdRA0/YzrguaCA6/vE/tUP+xDD/GOu7KyvN1dsgo1vnYZLG7p8vGgt61eTyC
V6qMEkceGatQvfiBb4XZCeODtC2KBv4pbJpt1zPKOpF4XFb2LferPxAL9rsSRSUOk9tZNc
x1GM3+UUYwT9dF8bcUfomZCKd07kzPh206nZr/uCElXVtCqghgVRQW8uiFRh6ycVWY/pBq
uhPfihKHilZEahOOc08ax14XTK89ovJzjbHezaV/NghkfWpn3W7gDNJTbLbxpbrLDkJBPJ
IltJ5QqwVK/Hi+69x9CxFOkyNpxWFexHPIeq4q0liPoah42MBPAQl30bWULgBP+K0ugzqQ
cSPAhi9FMq6ZVFTmaiPX8JH8JAceswd500x9jMmV91obzTZmXAQsfVpi0asxRhfficEIfs
UJ/FHwW2p13YmDVH1AjVmCDn9T46I05Cq+ImrUBX+JAEa6yQU6R6/s7maVDqpdtkpFp0ql
CWQHHw9J1fYS4w==
---- END SSH2 PUBLIC KEY ----
[user@client ~]$
```

2. On the SBC, use the **ssh-key** command with the **authorized-key import** argument.

   The command syntax:

   ```
   ssh-key authorized-key import <name> <class>
   ```

   The `<name>` parameter is the identifier for the SSH client. The `<class>` is one of the two authorization classes on the SBC: either `user` or `admin`.

   ```
   ORACLE# ssh-key authorized-key import bob admin

   IMPORTANT:
           Please paste SSH public key in the format defined in RFC 4716.
           Terminate the key with ";" to exit.......

   ---- BEGIN SSH2 PUBLIC KEY ----
   Comment: "4096-bit RSA, converted by bob@client from OpenSSH"
   AAAAB3NzaC1yc2EAAAADAQABAAACAQDOTDujYoQXzjTt9I8YvJMvfSVlWZ6iDzfRx06R3l
   Rj/lrjxlWDMc/Y/uEd2sJ+5wdlCnJPREOuCGbU8S6295486D1kbu76cEDxE+adca3/9+qo
   7FQVugkRJBD0ZOj/3qcuKDOh6ZsalF9LaaNMPNWNiQ5n3bWBnQ1tMMEes58JvoNgjn9FOz
   hbOdOe91K/OdRA0/YzrguaCA6/vE/tUP+xDD/GOu7KyvN1dsgo1vnYZLG7p8vGgt61eTyC
   V6qMEkceGatQvfiBb4XZCeODtC2KBv4pbJpt1zPKOpF4XFb2LferPxAL9rsSRSUOk9tZNc
   x1GM3+UUYwT9dF8bcUfomZCKd07kzPh206nZr/uCElXVtCqghgVRQW8uiFRh6ycVWY/pBq
   uhPfihKHilZEahOOc08ax14XTK89ovJzjbHezaV/NghkfWpn3W7gDNJTbLbxpbrLDkJBPJ
   IltJ5QqwVK/Hi+69x9CxFOkyNpxWFexHPIeq4q0liPoah42MBPAQl30bWULgBP+K0ugzqQ
   cSPAhi9FMq6ZVFTmaiPX8JH8JAceswd500x9jMmV91obzTZmXAQsfVpi0asxRhfficEIfs
   UJ/FHwW2p13YmDVH1AjVmCDn9T46I05Cq+ImrUBX+JAEa6yQU6R6/s7maVDqpdtkpFp0ql
   CWQHHw9J1fYS4w==
   ---- END SSH2 PUBLIC KEY ----;
   ```

> **Note:**
>
> If the Admin Security entitlement is enabled, the SSH client keys must be at least 2048 bits.

> **Note:**
>
> Oracle recommends keys be at least 2048 bits.

3. Save and activate the configuration.

Export an Authorized Key

To export a previously imported SSH public key, use the **ssh-key** command with the **authorized-key export** argument.

1. List the available **ssh-key** elements.

```
ORACLE# show running-config ssh-key
ssh-key
        name                                    bob
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.20
        last-modified-date                      2020-05-12 13:58:39
ssh-key
        name                                    alice
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-12 14:23:47
ssh-key
        name                                    logserver
        type                                    known-host
        encryption-type                         rsa
        size                                    2048
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-11 15:18:36
```

2. For any **ssh-key** element whose type is **authorized-key**, use the **ssh-key authorized-key export <name>** command to export the user's public key.

```
ORACLE# ssh-key authorized-key export bob
public-key 'bob' (RFC 4716/SECSH format):

---- BEGIN SSH2 PUBLIC KEY ----
Comment: "4096-bit rsa"
AAAAB3NzaC1yc2EAAAADAQABAAACAQDOTDujYoQXzjTt9I8YvJMvfSVlWZ6iDzfRx06R3l
Rj/lrjxlWDMc/Y/uEd2sJ+5wdlCnJPREOuCGbU8S6295486D1kbu76cEDxE+adca3/9+qo
7FQVugkRJBD0ZOj/3qcuKDOh6ZsalF9LaaNMPNWNiQ5n3bWBnQ1tMMEes58JvoNgjn9FOz
hbOdOe91K/OdRA0/YzrguaCA6/vE/tUP+xDD/GOu7KyvN1dsgo1vnYZLG7p8vGgt61eTyC
```

```
V6qMEkceGatQvfiBb4XZCeODtC2KBv4pbJpt1zPKOpF4XFb2LferPxAL9rsSRSUOk9tZNc
x1GM3+UUYwT9dF8bcUfomZCKd07kzPh206nZr/uCElXVtCqghgVRQW8uiFRh6ycVWY/pBq
uhPfihKHilZEahOOc08ax14XTK89ovJzjbHezaV/NghkfWpn3W7gDNJTbLbxpbrLDkJBPJ
IltJ5QqwVK/Hi+69x9CxFOkyNpxWFexHPIeq4q0liPoah42MBPAQl30bWULgBP+K0ugzqQ
cSPAhi9FMq6ZVFTmaiPX8JH8JAceswd500x9jMmV91obzTZmXAQsfVpi0asxRhfficEIfs
UJ/FHwW2p13YmDVH1AjVmCDn9T46I05Cq+ImrUBX+JAEa6yQU6R6/s7maVDqpdtkpFp0ql
CWQHHw9J1fYS4w==
---- END SSH2 PUBLIC KEY ----

ORACLE#
```

Delete an Authorized Key

To delete a previously imported SSH public key, use the **ssh-key** command with the **authorized-key delete** argument.

1. List the available **ssh-key** elements.

```
ORACLE# show running-config ssh-key
ssh-key
        name                                    bob
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.20
        last-modified-date                      2020-05-12 13:58:39
ssh-key
        name                                    alice
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-12 14:23:47
ssh-key
        name                                    logserver
        type                                    known-host
        encryption-type                         rsa
        size                                    2048
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-11 15:18:36
```

2. For any **ssh-key** element whose type is **authorized-key**, use the **ssh-key authorized-key delete <name>** command to delete the user's public key.

```
ORACLE# ssh-key authorized-key delete bob
SSH public key deleted successfully....
WARNING: Configuration changed, run "save-config" command to save it
and run "activate-config" to activate the changes
ORACLE#
```

3. Save and activate the configuration.

Add an SSH Known Host Key

For the SBC to authenticate over SSH to an SFTP server, the public key of the SFTP server needs to be imported into the known_hosts file of the SBC.

1. Convert the public key of the SFTP server into RFC 4716 format.

   There are two ways to do this.

   a. SSH to the SFTP server and run the **ssh-keygen** command on the server's host key. For OpenSSH implementations, host keys are generally found at `/etc/ssh/ssh_host_rsa_key.pub`. Other SSH implementations may differ. To do this on Oracle Linux, use the **ssh-keygen** command.

   ```
   [user@logserver ~]$ ssh-keygen -e -f /etc/ssh/ssh_host_rsa_key.pub
   ---- BEGIN SSH2 PUBLIC KEY ----
   Comment: "2048-bit RSA, converted by user@logserver from OpenSSH"
   AAAAB3NzaC1yc2EAAAADAQABAAABAQDwifpOpBKoDhzJXglzdoOfZ39TiU7jhygbPGQTw0
   j3zISW57PRbSulVw1hBHwqJwZZc6nr1JXaiHN7ieYT/96QCXQ56JH9Lcjej6iHplfhJO44
   qIgZIlRtD0e5y6YBzDgcI3T8J6n0jHwksvwKttObk8SoZl1mqE4xPXSiTVB1PzMNxF0dWV
   rgvGK227PsOfPLypL3RhnmqFbVRIhMKW7a80p7I+T6mAoq8UdzejbyhEK+e0Ge3F9i1g49
   oHWHNnSvU64F1ADybbZrclvvt8vofIzraGMBRjLs5Yl8bbdId/4UBci1fONmIUzxVse5NM
   PwNj0cjvNPS1/LOcKUgQxN
   ---- END SSH2 PUBLIC KEY ----
   [user@logserver ~]$
   ```

   b. Run the **ssh-keyscan** command from a Linux client and convert that key with the **ssh-keygen** command.

   ```
   ssh-keyscan -t rsa 10.0.0.6 | sed 's/.*ssh/ssh/' > key.pub
   ssh-keygen -ef key.pub
   ```

2. On the SBC, use the **ssh-key** command to import the host key of the SFTP server into the known_hosts file of the SBC.

   The command syntax:

   ```
   ssh-key known-host import <name>
   ```

   For SFTP to work properly, the <name> parameter must be the valid and reachable IP address of the SFTP server.

   The following example shows adding multiple servers with different IP addresses. Note that you must add each server in an individual command.

   ```
   ssh-key known-host import 10.10.10.1
   ssh-key known-host import 192.168.1.1
   ```

   Alternatively, you may prepend a server "alias" to the start of the IP address string as follows.

   ```
   ssh-key known-host import serverA@10.10.10.1
   ssh-key known-host import serverB@192.168.1.1
   ```

   If you need to have multiple server aliases that have the same IP address, you can do the following.

   ```
   ssh-key known-host import serverA@10.10.10.1
   ssh-key known-host import serverB@10.10.10.1
   ```

3. Paste the public key with the bracketing Begin and End markers at the cursor point.
4. Enter a semi-colon (;) to signal the end of the imported host key.

   The entire import sequence is shown below.

```
ORACLE# ssh-key known-host import 10.0.0.12

IMPORTANT:
        Please paste SSH public key in the format defined in RFC 4716.
        Terminate the key with ";" to exit.......

---- BEGIN SSH2 PUBLIC KEY ----
Comment: "2048-bit RSA, converted by user@logserver from OpenSSH"
AAAAB3NzaC1yc2EAAAADAQABAAABAQDJXglzdiU7jhywifpOpBKoDhoOfZ39TzgbPGQTw0
j357PRbSulHwaiHN7zEVw1hBISWie6nrQ56JH9Lcjej1JX96QCYT/qJwZZcX6iHplfhJO4
q8J6nIlRtD0e5y60jHwgZYBzDksvwKk8SSiTVB10ttObdWVoZl1mqPzMNxFE4xPXIgcI3T
rgvGKR27PsOfPLy80p7IpLhnmqFjbyhEK+e0KW7a+T6mbV23RIhMzeAoq8UdGe3F9i1g49
oHWs5mDybHNnBRjLbZrcSvU64F1AMlvvtUzxVse5NM8vofIzraGIYl8bbdId/4UBci1fON
PwNPS1/LONj0cjvcKUgQxN
---- END SSH2 PUBLIC KEY ----;

SSH public key imported successfully....
WARNING: Configuration changed, run "save-config" command to save it
and run "activate-config" to activate the changes
```

   Import both the RSA key and the DSA key if you are not sure which one the SFTP server uses.

5. Save and activate the configuration.

Delete an SSH Known Hosts Key

Delete expired SSH keys from the known_hosts file of the SBC.

1. List the available **ssh-key** elements.

```
ORACLE# show running-config ssh-key
ssh-key
        name                                    bob
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.20
        last-modified-date                      2020-05-12 13:58:39
ssh-key
        name                                    alice
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-12 14:23:47
ssh-key
        name                                    10.0.0.12
        type                                    known-host
        encryption-type                         rsa
        size                                    2048
```

```
last-modified-by                        admin@10.0.0.37
last-modified-date                      2020-05-11 15:18:36
```

2. Use the **ssh-key** command to remove a key whose type is known-host.

   The command syntax:

   ```
   ssh-key known-host delete <name>
   ```

   The <name> parameter is an alias or handle assigned to the imported host key.

   ```
   ORACLE# ssh-key known-host delete 10.0.0.12
   ```

3. Save and activate the configuration.

Add a Certificate Authority Key

When authenticating with certificates, clients send certificates to establish their identity and authorization. The public key of the Certificate Authority (CA) used for signing these client certificates must be imported into the SBC.

1. On the server you'll use for a certificate authority, create a `keys` directory for storing keys.

   ```
   [user@host ~]$ mkdir keys
   [user@host ~]$ cd keys/
   ```

2. Generate an SSH key pair to use for signing certificates.

   ```
   [user@host keys]$ ssh-keygen -t rsa -b 4096 -f ./ca_key
   ```

3. Export the CA key to RFC 4716 format.

   ```
   [user@host keys]$ ssh-keygen -ef ./ca_key.pub
   ---- BEGIN SSH2 PUBLIC KEY ----
   Comment: "4096-bit RSA, converted by user@host from OpenSSH"
   AAAAB3NzaC1yc2EAAAADAQABAAACAQDOTDujYoQXzjTt9I8YvJMvfSVlWZ6iDzfRx06R3l
   Rj/lrjxlWDMc/Y/uEd2sJ+5wdlCnJPREOuCGbU8S6295486D1kbu76cEDxE+adca3/9+qo
   7FQVugkRJBD0ZOj/3qcuKDOh6ZsalF9LaaNMPNWNiQ5n3bWBnQ1tMMEes58JvoNgjn9FOz
   hbOdOe91K/OdRA0/YzrguaCA6/vE/tUP+xDD/GOu7KyvN1dsgo1vnYZLG7p8vGgt61eTyC
   V6qMEkceGatQvfiBb4XZCeODtC2KBv4pbJpt1zPKOpF4XFb2LferPxAL9rsSRSUOk9tZNc
   x1GM3+UUYwT9dF8bcUfomZCKd07kzPh206nZr/uCElXVtCqghgVRQW8uiFRh6ycVWY/pBq
   uhPfihKHilZEahOOc08ax14XTK89ovJzjbHezaV/NghkfWpn3W7gDNJTbLbxpbrLDkJBPJ
   IltJ5QqwVK/Hi+69x9CxFOkyNpxWFexHPIeq4q0liPoah42MBPAQl30bWULgBP+K0ugzqQ
   cSPAhi9FMq6ZVFTmaiPX8JH8JAceswd500x9jMmV91obzTZmXAQsfVpi0asxRhfficEIfs
   UJ/FHwW2p13YmDVH1AjVmCDn9T46I05Cq+ImrUBX+JAEa6yQU6R6/s7maVDqpdtkpFp0ql
   CWQHHw9J1fYS4w==
   ---- END SSH2 PUBLIC KEY ----
   [user@host keys]$
   ```

4. Import the CA key into the SBC using the **ssh-key** command with the **ca-key import** argument.

   The command syntax:

   ```
   ssh-key ca-key import <key-name> <class>
   ```

The <key-name> parameter is the key identifier or key ID that will be used when signing client keys as the value of the -I argument in the ssh-keygen command. The <class> is one of the two authorization classes on the SBC: either user or admin.

```
ORACLE# ssh-key ca-key import rootCA admin

IMPORTANT:
        Please paste SSH public key in the format defined in RFC 4716.
        Terminate the key with ";" to exit.......

---- BEGIN SSH2 PUBLIC KEY ----
Comment: "4096-bit RSA, converted by user@server from OpenSSH"
AAAAB3NzaC1yc2EAAAADAQABAAACAQDOTDujYoQXzjTt9I8YvJMvfSVlWZ6iDzfRx06R3l
Rj/lrjxlWDMc/Y/uEd2sJ+5wdlCnJPREOuCGbU8S6295486D1kbu76cEDxE+adca3/9+qo
7FQVugkRJBD0ZOj/3qcuKDOh6ZsalF9LaaNMPNWNiQ5n3bWBnQ1tMMEes58JvoNgjn9FOz
hbOdOe91K/OdRA0/YzrguaCA6/vE/tUP+xDD/GOu7KyvN1dsgo1vnYZLG7p8vGgt61eTyC
V6qMEkceGatQvfiBb4XZCeODtC2KBv4pbJpt1zPKOpF4XFb2LferPxAL9rsSRSUOk9tZNc
x1GM3+UUYwT9dF8bcUfomZCKd07kzPh206nZr/uCElXVtCqghgVRQW8uiFRh6ycVWY/pBq
uhPfihKHilZEahOOc08ax14XTK89ovJzjbHezaV/NghkfWpn3W7gDNJTbLbxpbrLDkJBPJ
IltJ5QqwVK/Hi+69x9CxFOkyNpxWFexHPIeq4q0liPoah42MBPAQl30bWULgBP+K0ugzqQ
cSPAhi9FMq6ZVFTmaiPX8JH8JAceswd500x9jMmV91obzTZmXAQsfVpi0asxRhfficEIfs
UJ/FHwW2p13YmDVH1AjVmCDn9T46I05Cq+ImrUBX+JAEa6yQU6R6/s7maVDqpdtkpFp0ql
CWQHHw9J1fYS4w==
---- END SSH2 PUBLIC KEY ----;
```

> **Note:**
>
> If the Admin Security entitlement is enabled, the key must be at least 2048 bits.

5. Save and activate the configuration.
6. For each SSH client, copy the client's public key into the keys directory.

```
[user@host keys]$ scp acme@client1.com:.ssh/id_rsa.pub ./id_rsa.pub
```

7. Sign the key with the **ssh-keygen** command.

   Use the following arguments:

   - Use -s to identify the private key of the CA key used to sign.
   - Use -z to specify the serial number to be embedded in the certificate to distinguish this certificate from others signed by the same CA.
   - Use -n to specify the username of the client to be included in the certificate.
   - Use -I to specify the key ID. This key ID must match the <key-name> specified when importing the signing CA key into the SBC.
   - Use -V to set the validity interval. To set the validity for one year, starting the previous day, use -1d:+52w.

> **Important:**
>
> The username passed with the `-n` argument of the **ssh-keygen** command must match the username used to authenticate.

> **Note:**
>
> If the **type** attribute of the **authentication** element is set to **local**, the username passed with the -n argument must be set to admin.

```
[user@host keys]$ ssh-keygen -s ca_key -z 1 -n admin -I rootCA -V -1d:+52w
id_rsa.pub
Signed user key id_rsa.pub: id "rootCA" serial 1 for admin valid from
2020-06-21T09:26:41 to 2021-06-21T09:26:41
[user@host keys]$
```

8. Copy the certificate to the client's `.ssh` directory.

```
[user@host keys]$ scp id_rsa-cert.pub acme@client1.com:.ssh/
```

9. Verify the SSH client can connect with the certificate.

Delete a Certificate Authority Key

To delete a previously imported Certificate Authority (CA) key, use the **ssh-key** command with the **ca-key delete** argument.

1. List the available **ssh-key** elements.

```
ORACLE# show running-config ssh-key
ssh-key
        name                                    bob
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.20
        last-modified-date                      2020-05-12 13:58:39
ssh-key
        name                                    alice
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-12 14:23:47
ssh-key
        name                                    rootCA
        type                                    ca-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-11 15:18:36
```

2. For any **ssh-key** element whose type is **ca-key**, use the **ssh-key ca-key delete <key-name>** command to delete the CA key.

```
ORACLE# ssh-key ca-key delete rootCA
SSH public key deleted successfully....
WARNING: Configuration changed, run "save-config" command to save it
and run "activate-config" to activate the changes
ORACLE#
```

3. Save and activate the configuration.

Revoke a User Key

To revoke access to a specific user whose public key was signed by your CA key, import the user's public key into the revocation list.

1. On the SBC, use the **ssh-key** command with the **ca-user-revoke import** argument.

   The command syntax:

```
ssh-key ca-user-revoke import <key-name>
```

   The `<key-name>` parameter uniquely identifies the key you want to revoke.

```
ORACLE# ssh-key ca-user-revoke import bob

IMPORTANT:
        Please paste SSH public key in the format defined in RFC 4716.
        Terminate the key with ";" to exit.......

---- BEGIN SSH2 PUBLIC KEY ----
Comment: "4096-bit RSA, converted by user@server from OpenSSH"
AAAAB3NzaC1yc2EAAAADAQABAAACAQDOTDujYoQXzjTt9I8YvJMvfSVlWZ6iDzfRx06R3l
Rj/lrjxlWDMc/Y/uEd2sJ+5wdlCnJPREOuCGbU8S6295486D1kbu76cEDxE+adca3/9+qo
7FQVugkRJBD0ZOj/3qcuKDOh6ZsalF9LaaNMPNWNiQ5n3bWBnQ1tMMEes58JvoNgjn9FOz
hbOdOe91K/OdRA0/YzrguaCA6/vE/tUP+xDD/GOu7KyvN1dsgo1vnYZLG7p8vGgt61eTyC
V6qMEkceGatQvfiBb4XZCeODtC2KBv4pbJpt1zPKOpF4XFb2LferPxAL9rsSRSUOk9tZNc
x1GM3+UUYwT9dF8bcUfomZCKd07kzPh206nZr/uCElXVtCqghgVRQW8uiFRh6ycVWY/pBq
uhPfihKHilZEahOOc08ax14XTK89ovJzjbHezaV/NghkfWpn3W7gDNJTbLbxpbrLDkJBPJ
IltJ5QqwVK/Hi+69x9CxFOkyNpxWFexHPIeq4q0liPoah42MBPAQl30bWULgBP+K0ugzqQ
cSPAhi9FMq6ZVFTmaiPX8JH8JAceswd500x9jMmV91obzTZmXAQsfVpi0asxRhfficEIfs
UJ/FHwW2p13YmDVH1AjVmCDn9T46I05Cq+ImrUBX+JAEa6yQU6R6/s7maVDqpdtkpFp0ql
CWQHHw9J1fYS4w==
---- END SSH2 PUBLIC KEY ----;
```

2. Save and activate the configuration.

The user's key is added to the revocation list. When authenticating to the SBC, the user may no longer use his or her key or certificate, even though that key was signed by the CA key.

Unrevoke a Revoked User Key

If a user key is added to the revocation list, that user will not be able to authenticate to the SBC. To delete a key from the revocation list, use the **ssh-key** command with the **ca-user-revoke delete** argument.

1. List the available **ssh-key** elements.

```
ORACLE# show running-config ssh-key
ssh-key
        name                                    bob
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.20
        last-modified-date                      2020-05-12 13:58:39
ssh-key
        name                                    alice
        type                                    authorized-key
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-12 14:23:47
ssh-key
        name                                    alice
        type                                    ca-user-revoke
        encryption-type                         rsa
        size                                    4096
        last-modified-by                        admin@10.0.0.37
        last-modified-date                      2020-05-11 15:18:36
```

2. For any **ssh-key** element whose type is **ca-user-revoke**, use the **ssh-key ca-user-revoke delete <key-name>** command to delete the CA key.

```
ORACLE# ssh-key ca-user-revoke delete alice
SSH public key deleted successfully....
WARNING: Configuration changed, run "save-config" command to save it
and run "activate-config" to activate the changes
ORACLE#
```

3. Save and activate the configuration.

Once the user key is removed from the revocation list, the functionality of any existing key is restored.

Configure SSH Ciphers

The **ssh-config** configuration element controls which ciphers the SBC offers during SSH session negotiation when the SBC acts as an SSH server. The ciphers offered when the SBC acts as an SSH client are not configurable.

Each command takes an argument which is either a single word or a comma-separated list within double quotes. Type `?` to see the available algorithms for this release.

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

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ssh-config
```

2. **encr-algorithms**—Select the ciphers for SSH encryption.
3. **hmac-algorithms**—Select the HMAC algorithm.

4. **keyex-algorithms**—Select the Diffie-Hellman key exchange algorithm.
5. **hostkey-algorithms**—Select the algorithm for generating host keys.
6. Type **done**.
7. Save and activate the configuration.

Verify SSH Ciphers

After configuring which ciphers the Oracle Communications Session Border Controller offers during SSH negotiations, verify the settings from an SSH client by starting a new SSH session with verbosity level 2.

1. SSH to the SBC with verbosity level 2.

```
ssh -vv user@10.0.0.1
```

2. Confirm the SBC offers the selected ciphers.

```
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit: first_kex_follows 0
debug2: kex_parse_kexinit: reserved 0
debug2: kex_parse_kexinit: diffie-hellman-group-exchange-sha256
debug2: kex_parse_kexinit: ssh-rsa
debug2: kex_parse_kexinit: AEAD_AES_256_GCM,aes256-ctr
debug2: kex_parse_kexinit: AEAD_AES_256_GCM,aes256-ctr
debug2: kex_parse_kexinit: hmac-sha2-256
debug2: kex_parse_kexinit: hmac-sha2-256
debug2: kex_parse_kexinit: none
debug2: kex_parse_kexinit: none
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit:
```

System Boot

When your Oracle Communications Session Border Controller boots, the following information about the tasks and settings for the system appear in your terminal window.

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

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

Boot Parameters

Boot parameters specify the information that your device uses at boot time when it prepares to run applications.

This section explains how to view, edit, and implement device's boot parameters, and boot flags. Boot parameters:

- Allow you to set the IP address for the management interface (wancom0).
- Allow you to set a system prompt. The target name parameter also specifies the title name displayed in your web browser and SNMP device name parameters.
- Specify the software image to boot and from where the system boots that image.

> **Note:**
>
> You must configure all three components of an IPv6 address, including address, mask and gateway, in your system's boot parameters for wancom0 addressing. Configure the mask as a forslash (/) after the address followed by the mask in number of bits. The system requires all three components for IPv6 Neighbor Discovery to work properly.

Boot flags are arguments to a specific boot parameter, and allow functional settings, such as the use of DHCP for acquiring a management port address, as well as various diagnostic startup configurations.

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

The bootparam configuration list is shown below.

```
[Acme Boot]: p
Boot File         : /boot/bzImage
IP Address        : 172.44.12.89
VLAN              :
Netmask           : 255.255.0.0
Gateway           : 172.44.0.1
IPv6 Address      : 3fff:ac4:6001:0:208:25ff:fe05:f470/64
IPv6 Gateway      : 3fff:ac4:6001::ac4:6001
Host IP           :
FTP username      :
FTP password      :
Flags             : 0x00000040
Target Name       : ORACLE
Console Device    : COM1
Console Baudrate : 115200
Other             :


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

Boot Parameter Definitions

The system displays all boot parameters when you configure them after a boot interrupt. The system hides some boot parameters from the ACLI so that you do not attempt to configure them. If changed improperly, these parameters can cause the system to sop responding.

The following table defines each of the parameters that the system displays when you perform configuration after a boot interrupt.

Boot Parameter	Description
Boot File	The name and path of the software image you are booting. Include the absolute path for a local boot from the local /boot volume and for a net boot when a path on the FTP server is needed.
IP Address	IP address of wancom0.
VLAN	VLAN of management network over which this address is accessed. **Note:** VLANs over management interfaces are supported only on the Acme Packet 1100. **Note:** The acquire-config command is not supported on management interfaces that use both VLANs and IPv6.
Netmask	Netmask portion of the wancom0 IP Address.
Gateway	Network gateway that this wancom0 interface uses.
IPv6 address	Version 6 IP address/mask of wancom0. Configure the mask as a forslash (/) after the address followed by the mask in number of bits.
IPv6 Gateway	Version 6 network gateway that this wancom0 interface uses.
Host IP	IP Address of FTP server from which to download and execute a software image.
FTP Username	FTP server username
FTP password	FTP server password
Flags	Codes that signal the system from where to boot. Also signals the system about which file to use in the booting process. This sequence always starts with 0x (these flags are hexadecimal).
Target Name	Name of the Oracle Communications Session Border Controller as it appears in the system prompt. For example, ORACLE> or ORACLE#. You need to know the target name if you are setting up an HA node. This name must be unique among Oracle Communications Session Border Controllers in your network. This name can be 63 characters or less.
Console Device	Serial output device type, dependent on platform. COM1 applies to virtual serial consoles, VGA to virtual video console. VGA is the default on VMware and KVM. COM1 is the default on OVM .

Boot Parameter	Description
Console Baud Rate	The speed in bits per second which the console port operates at. It operates at 115200 BPS, 8 data bits, no stop bit, parity NONE.
Other	Allows miscellaneous and deployment-specific boot settings.

Boot Flags

Boot flags enable system boot behavior(s). The user can set a single flag, or add hex digits to set multiple flags.

- 0x00000008 Bootloader ~7 seconds countdown
- 0x00000040 Autoconfigure wancom0 via DHCP enable - VM platforms only
- 0x00000080 Use TFTP protocol (instead of FTP) enable - VM platforms only
- 0x00000100 Bootloader ~1 seconds quick countdown - VM platforms only

The following boot flags should only be used as directed by Oracle support:

- 0x00000001 acme.ko network module security override
- 0x00000002 Kernel debug enable
- 0x00000004 Crashdump disable
- 0x00000010 Debug sshd enable
- 0x00000020 Debug console enable getty
- 0x00001000 Userspace debug enable
- 0x00100000 Uniprocessor enable (SMP disable)
- 0x20000000 Fail-safe boot enable
- 0x40000000 Process startup disable (flatspin mode)

Never enter any other values without the direction of Oracle support. Some diagnostic flags are not intended for normal system operation.

Setting Up System Basics

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

New System Prompt

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

Set Initial Passwords for Admin and User

The Oracle Communications Session Border Controller (SBC) requires you to set passwords for the Admin and User accounts the first time you power up a new or factory reset system by way of local access. You cannot access the Admin and User accounts until you set the

corresponding passwords. Use either an SSH or console connection when setting passwords. The following procedure is for local access. If you use remote access, for example, RADIUS or TACACS, use your passwords for those services.

Before you begin, plan your passwords to meet the following requirements:

- 8-64 characters
- Include three of the following:
  - Lower case letters
  - Uppercase letters
  - Numerals
  - Punctuation

The system leads you through the process for setting the Admin and User passwords, as follows:

1. Power up the SBC.

   The system prompts you to set the User account password.
2. At the prompt, type **acme**, and press ENTER.

   The system prompts you to enter the password that you want for the User account.
3. Type the User account password, and press ENTER.
4. Type **enable**, and press ENTER.

   The system prompts you to set the Admin account password.
5. Type **packet**, and press ENTER.

   The system prompts you to enter the password that you want for the Admin account.
6. Type the Admin account password, and press ENTER.

   The system logs you in as Admin.

Using the Oracle Communications Session Border Controller Image

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

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

Obtaining a New Image

You can download a software image onto the Oracle Communications Session Border Controller platform from the following sources.

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

Regardless of the source, use SFTP to copy the image from the source to the Oracle Communications Session Border Controller.

Copy an Image to the Oracle Communications Session Border Controller using SFTP

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

To copy an image on your Oracle Communications Session Border Controller using SFTP:

1. Go to the directory where the image is located.
2. Check the IP address of the Oracle Communications Session Border Controller's management port (wancom0).
3. Create the connection to the Oracle Communications Session Border Controller.

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

   Using your SFTP application, start an SFTP session to the IPv4 address of the Oracle Communications Session Border Controller management port (wancom0). An SFTP username and SFTP password is required to start the session. The username is always admin, and the password is the local admin login password.

   Only the local admin user can write to the /boot directory.
4. Set your SFTP application to copy the image to the **/boot** folder using binary transfer mode.
5. Invoke and confirm your SFTP file transfer to the device.
6. Boot the Oracle Communications Session Border Controller using the image you just transferred.

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

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

System Image Filename

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

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

Booting an Image on Your Oracle Communications Session Border Controller

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

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

> **Note:**
>
> Only the local admin user can SFTP a boot image to the `/boot` directory.

Use the local admin user account to SFTP your boot image to the `/boot` directory or use a supplementary administrator user account (such as a TACACS+ or RADIUS administrator) to upload your boot image to the `/code/images` directory. Then set the Boot File parameter to your uploaded boot file.

Boot Parameter Definitions

The system displays all boot parameters when you configure them after a boot interrupt. The system hides some boot parameters from the ACLI so that you do not attempt to configure them. If changed improperly, these parameters can cause the system to sop responding.

The following table defines each of the parameters that the system displays when you perform configuration after a boot interrupt.

Boot Parameter	Description
Boot File	The name and path of the software image you are booting. Include the absolute path for a local boot from the local /boot volume and for a net boot when a path on the FTP server is needed.
IP Address	IP address of wancom0.

Boot Parameter	Description
VLAN	VLAN of management network over which this address is accessed. **Note:** VLANs over management interfaces are supported only on the Acme Packet 1100. **Note:** The acquire-config command is not supported on management interfaces that use both VLANs and IPv6.
Netmask	Netmask portion of the wancom0 IP Address.
Gateway	Network gateway that this wancom0 interface uses.
IPv6 address	Version 6 IP address/mask of wancom0. Configure the mask as a forslash (/) after the address followed by the mask in number of bits.
IPv6 Gateway	Version 6 network gateway that this wancom0 interface uses.
Host IP	IP Address of FTP server from which to download and execute a software image.
FTP Username	FTP server username
FTP password	FTP server password
Flags	Codes that signal the system from where to boot. Also signals the system about which file to use in the booting process. This sequence always starts with 0x (these flags are hexadecimal).
Target Name	Name of the Oracle Communications Session Border Controller as it appears in the system prompt. For example, ORACLE> or ORACLE#. You need to know the target name if you are setting up an HA node. This name must be unique among Oracle Communications Session Border Controllers in your network. This name can be 63 characters or less.
Console Device	Serial output device type, dependent on platform. COM1 applies to virtual serial consoles, VGA to virtual video console. VGA is the default on VMware and KVM. COM1 is the default on OVM .
Console Baud Rate	The speed in bits per second which the console port operates at. It operates at 115200 BPS, 8 data bits, no stop bit, parity NONE.
Other	Allows miscellaneous and deployment-specific boot settings.

Booting from Local Storage

Once you have installed an image, you can boot your Oracle Communications Session Border Controller from its local storage.

To boot from your local storage:

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

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

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

2. Change any boot configuration parameters that you need to change. It is especially important to change the file name boot configuration parameter. The file name parameter needs to use the /boot value so that the Oracle Communications Session Border Controller boots from the local storage.
3. You can boot from a different image files using the l (small letter L) option from the boot menu prompt. This allows you to select an alternate image file when the boot file gets corrupted. This option is available only when FIPS mode is disabled. You can view the files in a page view when there are more than 15 files in the directories. When Oracle Communications Session Border Controller boots over network using ftp/sftp with the selected image file, this option does not update the boot param. This option displays image files from both /boot and /code/images directories irrespective of R226 license.

```
[Acme Boot]: p
Boot File         : /boot/bzImage
IP Address        : 172.44.12.89
VLAN              :
Netmask           : 255.255.0.0
Gateway           : 172.44.0.1
IPv6 Address      : 3fff:ac4:6001:0:208:25ff:fe05:f470/64
IPv6 Gateway      : 3fff:ac4:6001::ac4:6001
Host IP           :
FTP username      :
FTP password      :
Flags             : 0x00000040
Target Name       : ORACLE
Console Device    : COM1
Console Baudrate : 115200
Other             :


[Acme Boot]: ?
 ?                    - print this list
 @                    - boot (load and go)
 p                    - print boot params
 c                    - change boot params
 o                    - Oracle Rescue Access sub-menu
 v                    - print boot logo with version
 r                    - reboot
 d                    - list diagnostic images
 s                    - show license information
 l                    - show boot images

Boot flags:
   0x02   - enable kernel debug
   0x04   - disable crashdumps and enable minidump
   0x10   - enable debug login
   0x40   - use DHCP for wancom0
   0x80   - use TFTP instead of FTP
```

```
[Acme Boot]:  1
1:    /boot/nnTCZ000c94.bz
2:    /boot/nnTCZ000c93.bz
3:    /boot/nnTCZ000c84.bz
4:    /boot/a.bz
5:    /boot/b.bz
6:    /boot/c.bz
7:    /boot/d.bz
8:    /boot/e.bz
9:    /boot/f.bz
10:     /code/images/nnTCZ000c94.bz
11:     /code/images/nnTCZ000c93.bz
12:     /code/images/nnTCZ000c84.bz
13:     /code/images/a.bz
14:     /code/images/b.bz
15:     /code/images/c.bz
Few more entries press c to continue or select from list to boot or any
other key to quit listing... :c

[Acme Boot]:[Boot Image]: 3
```

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

Setting Up Product-Type, Features, and Functionality

You enable features and functionality primarily through the Setup Entitlements task, where you self-provision features and certain session capacities. The Oracle Communications Session Border Controller is seeded with a default feature set when you first follow the Setup Product task, and is based on software version, the platform on which the software runs, and the product type that you choose.

> **Note:**
>
> Refer to the *Self-Provisioned Entitlements and License Keys* section in the *Release Notes* for a list of the methods used to enable features in this release.

Initial Setup

Prior to system configuration, you must set up the product, self-provision entitlements, and optionally install license keys to activate features as required. If you log onto an unconfigured Oracle Communications Session Border Controller, the system displays a warning message that a valid product type is required.

System Setup

Before you begin configuring the Oracle Communications Session Border Controller, you will set the product type with the **setup product** command, and the features with the **setup entitlements** command.

> **Note:**
>
> Not all of the features are available on all platforms.

Setup Product

1. Type **setup product** at the ACLI. If this is the first time running the command on this hardware, the product will show as Uninitialized.
2. Type **1 <Enter>** to modify the uninitialized product.
3. Type the number followed by **<Enter>** for the product type you wish to initialize.
4. Type **s <Enter>** to commit your choice as the product type of this platform.
5. Reboot your Oracle Communications Session Border Controller.

```
ORACLE# setup product

----------------------------------------------------------------
WARNING:
Alteration of product alone or in conjunction with entitlement
changes will not be complete until system reboot

Last Modified
----------------------------------------------------------------
 1 : Product       : Uninitialized

Enter 1 to modify, d' to display, 's' to save, 'q' to exit. [s]: 1

  Product
    1 - Session Border Controller
    2 - Session Router - Session Stateful
    3 - Session Router - Transaction Stateful
    4 - Subscriber-Aware Load Balancer
    5 - Enterprise Session Border Controller
    6 - Peering Session Border Controller
  Enter choice     : 1

Enter 1 to modify, d' to display, 's' to save, 'q' to exit. [s]: s
save SUCCESS
```

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

Setup Entitlements

1. Type **setup entitlements** at the ACLI. Currently provisioned features are printed on the screen.

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

Editing and Viewing Features

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

The **show entitlements** command displays the currently provisioned features and controlled features. You may also use the **setup entitlements** command and type **d** to display the current features. Upon first executing the **setup entitlements** command, all features (excluding controlled features) are displayed on the screen.

License Keys and Self-Provisioned Entitlements Compatibility

The Oracle Communications Session Border Controller continues to support any license keys originally purchased and installed pre-self-provisioned-entitlements for enabling system features.

The licensing and entitlements processes work with each other, as follows.

- You must use self-provisioning to enable features and session capacity on all platforms.
- Oracle only provides license keys to enable specific features and capacities not available for self-provisioning, such as codecs and regulatory features .
- Upon migrating to self-provisioned entitlements, the system seeds the current range of your installed license keys to the self-provisioned entitlements. The system's functionality remains identical.
- When you upgrade to self-provisioned entitlements and then downgrade the software to an older version that requires license keys, any pre-existing license keys will still function.
- When you upgrade to self-provisioned entitlements and then change the functionality (such as, adding more SIP sessions or removing a feature set), the new functionality will not be present upon downgrade to an older version that requires license keys.

System Setup with Existing License Keys

When changing the Oracle Communications Session Border Controller licensing technique from the legacy license key method to the self-provisioned method, be aware of the following:

- After running **setup product** and **setup entitlements**, the system will be seeded with the existing license keys' functionalities.

- When the system is seeded with its previous functionality to the provisioned entitlements system, functionality may be changed with the **setup entitlements** command.
- You may notice that there are fewer entitlements than there were with license keys. This is normal.
- After setting up self-provisioned features, the **show features** command will still function to display the previously installed license keys.

Concurrent Session License Usage

This feature allows the SBC to track the maximum values of it's licensed session usage over time. Specific 'high water marks' that the system stores includes total sessions, SRTP sessions, and transcoding Sessions on a rolling 365-day period with timestamps for auditing purposes. This can inform the customer and Oracle if and when it has exceeded its licensed session usage over specific windows for up to one year. You configure this feature by setting the **peak-concurrent-license** parameter within the **system-config**.

When enabled, the SBC writes maximum sessions count values, along with timestamps indicating when maximum values occurred, for the applicable sessions types into the following files every fifteen minutes:

- TotalSessions.csv
- SRTPSessions.csv
- AMRWBSessions.csv
- EVRCSessions.csv
- EVRCBSessions.csv
- OPUSSessions.csv
- SILKSessions.csv
- EVSSessions.csv

> **Note:**
>
> If your device does not support any transcoding type or you have not added the applicable licenses, the device does not create the applicable files and entries.

These files include rolling data for the past 365 days. The system stores these files in the /code/peakLicenseData/ folder, which is only visible to those with admin privileges. The system accumulates data for a year, and then drops the oldest day of data each day forward.

When the SBC performs these session capacity checks, it compares the values you configured for your entitlements with the numbers of all active/concurrent sessions on the system. Each time it creates a new session, the SBC compares the active sessions counter with session capacity entitlements setting. If the active sessions are less than session capacity counter, the system increments the active sessions counter by one and creates the new session. Similarly, the system reduces the active sessions counter each time it deletes a session, which occurs when the session state changes to “terminated”.

For the purposes of this feature, the SBC maintains 2 key variables that it uses to track session use high water marks:

- peakValue

- peakTimestamp

The system stores these values after polling in each of the applicable files listed above. Whenever you run the **show peak-concurrent-license-usage** command or load the **Peak Concurrent License Usage** widget from the GUI, the SBC reads each line from file and compares the current peakValue with the peakValue in the file. If the new high value is greater than the stored value, the system updates the file with the new high. Whenever the system updates peakValue, it also updates the peakTimestamp. After finishing reading all lines in one file, the system dumps peakValue and peakTimestamp and repeats the same steps on the next file.

Configuration

You configure the system to use this feature by setting the **peak-concurrent-license** parameter within the **system-config**. No other configuration is required, and the feature is not dependent on or affected by any other configuration. If you disable the option, the system stops increasing and decreasing counters and stops writing to the files.

```
ORACLE(system-config)# peak-concurrent-license enabled
```

ACLI Reporting

The SBC displays the highest watermarks and timestamp for each data file in a rolling 365-day period when you execute the ACLI command **show peak-concurrent-license-usage**.

```
ORACLE# show peak-concurrent-license-usage
SessionType      PeakValue      TimeStamp
Total Sessions   4000           2023-03-08 06:32:22.321
SRTP Sessions    400            2023-03-08 06:15:41.225
AMRWB Sessions   100            2023-03-08 06:38:27.777
EVS Sessions     25             2023-03-08 06:25:54.330
OPUS Sessions    17             2023-03-08 06:33:06.456
SILK Sessions    90             2023-03-08 06:37:09.688
EVRC Sessions    98             2023-03-08 06:34:09.622
EVRCB Sessions   86             2023-03-08 06:26.34.786
```

When you enable this feature the **show peak-concurrent-license-usage** command displays the highest watermark entry for each session type in a rolling 365-day period. If you disable the feature, this command prints **Feature is Disabled**. If there are multiple entries with the same value, the system displays the least recent entries.

SNMP

You can access the data captured by this feature using SNMP. The data returned by SNMP includes a response with the timestamp on which the most recent 15-minute window collection event occurred, and the data captured within that window. Note that you cannot retrieve the output of the **show peak-concurrent-license-usage** command via SNMP.

See the **MIB Guide** for a detailed reference of the applicable SNMP OIDs.

HDR

You can access the data captured by this feature using the system's HDR feature. This requires that you enable collection on the **latest-peak-license-usage** HDR group. This data provided by this function is equivalent to that collected available using SNMP.

If the feature is disabled and you try to start the collection using, for example, the **request collection start latest-peak-license-usage** command, the system throws the following Error, does not create the applicable CSV files and does not attempt to write any of this data.

```
ORACLE# request collection start latest-peak-license-usage
Conflict
% ERROR: Collection not started because peak-concurrent-license flag is
disabled
ORACLE
```

See the **HDR Guide** for a detailed reference of the applicable HDR output files.

Adding and Deleting License Keys

Certain features may only be enabled with license keys, like royalty-based codecs. The following guidelines apply to these license keys:

- Each license key is bound to a specific Oracle Communications Session Border Controller by serial number.
- Multiple license keys can be active on the same Oracle Communications Session Border Controller simultaneously.
- If a feature is covered by more than one license key, the latest expiration date applies.
- You can activate and deactivate license keys in real time.

You can request license keys via the License Codes website at http://www.oracle.com/us/support/licensecodes/acme-packet/index.html

> **Note:**
>
> For a list of features enabled with license keys in your software, see the Feature Entitlements section in your version of the Release Notes.

License Key Expiration

When a license expires, you are no longer able to use the features associated with it. The Oracle Communications Session Border Controller automatically disables all associated functionality.

To avoid a license unexpectedly expiring and therefore potentially disrupting service, you should track expiration dates and renew licenses well in advance of expiration.

Expired licenses appear in the **show features** command until you delete them, although you cannot use those features. Deleting an expired license requires that you take the same steps as you do for deleting a valid one.

Add a License Key

Once you have obtained a license key, you can add it to your Oracle Communications Session Border Controller and activate it.

1. Access the **license** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# license
ORACLE(license)#
```

2. Using the **add** command and the license key you received from Oracle, add the license to your Oracle Communications Session Border Controller.

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

> **Note:**
>
> Do not type **done** after you add the license. If you do, the system will return an error.

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

> **Note:**
>
> Do not type **done** before exiting the **license** configuration element.

4. Type **exit** until you return to the top-level superuser prompt.

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

5. Perform a save- and activate- on the configuration.

```
ORACLE#  save-config
checking configuration
-------------------------------------------------------------------------------
-----
Results of config verification:
   0 configuration error
-------------------------------------------------------------------------------
-----
Save-Config received, processing.
waiting for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
ORACLE# activate-config
Activate-Config received, processing.
waiting for request to finish
Setting phy on Slot=0, Port=0, MAC=00:08:25:22:81:B0,
VMAC=00:08:25:22:81:B0
```

```
Setting phy on Slot=0, Port=1, MAC=00:08:25:22:81:B1,
VMAC=00:08:25:22:81:B1
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
```

Delete a License Key

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

To delete a license from the Oracle Communications Session Border Controller:

1. Access the **license** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# license
   ORACLE(license)#
   ```

2. Type **no** and press Enter. A list of possible licenses to delete appears.
3. Type the number corresponding to the license you want to delete and press Enter.

   ```
   selection:1
   ```

4. If the license has not expired, you will be asked to confirm the deletion.

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

   > **Note:**
   >
   > Do not type **done** before leaving the **license** configuration element.

5. Type **exit** until you return to the top-level superuser prompt.

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

6. Perform a save- and activate- on the configuration.

   ```
   ORACLE#  save-config
   checking configuration
   ----------------------------------------------------------------------------
   -----
   Results of config verification:
      0 configuration error
   ----------------------------------------------------------------------------
   -----
   Save-Config received, processing.
   waiting for request to finish
   ```

```
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
ORACLE# activate-config
Activate-Config received, processing.
waiting for request to finish
Setting phy on Slot=0, Port=0, MAC=00:08:25:22:81:B0,
VMAC=00:08:25:22:81:B0
Setting phy on Slot=0, Port=1, MAC=00:08:25:22:81:B1,
VMAC=00:08:25:22:81:B1
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
```

View Installed Features, Entitlements, and Licenses

Use the **show entitlements** command to display all self-provisioned entitlements and features enabled with license keys.

```
ORACLE# show entitlements
Provisioned Entitlements:
-------------------------
Session Border Controller Base    : enabled
Session Capacity                  : 16000
  Accounting                      : enabled
  IPv4 - IPv6 Interworking        :
  IWF (SIP-H323)                  :
  Load Balancing                  : enabled
  Policy Server                   : enabled
  Quality of Service              : enabled
  Routing                         : enabled
  SIPREC Session Recording        :
Admin Security                    :
IMS-AKA Endpoints                 : 0
IPSec Trunking Sessions           : 0
MSRP B2BUA Sessions               : 0
SRTP Sessions                     : 0

Keyed (Licensed) Entitlements
-----------------------------
LI
Transcode Codec AMR (25 AMR transcoding sessions)
Transcode Codec EVRC (25 EVRC transcoding sessions)
Transcode Codec Opus (25 OPUS transcoding sessions)
Transcode Codec SILK (25 SILK transcoding sessions)
```

Use the **show features** to display currently active features on the system. This command shows the union of features enabled with license keys and with the self-provisioning method.

```
ORACLE# show features
 Total session capacity: 16000
Enabled features:
        16000 sessions, SIP, H323, QOS, ACP, Routing, Load Balancing,
        Accounting, High Availability, LI, External BW Mgmt,
```

```
External CLF Mgmt, External Policy Services, ENUM, NSEP RPH,
Transcode Codec AMR (25 AMR transcoding sessions),
Transcode Codec EVRC (25 EVRC transcoding sessions), IDS,
IDS Advanced,
Transcode Codec Opus (25 OPUS transcoding sessions),
Transcode Codec SILK (25 SILK transcoding sessions)
```

Use the **licenses**' **show** command to see all features enabled by license keys.

```
ORACLE(license)# show
License #1: 16000 sessions, SIP, LI,
            Transcode Codec AMR (25 AMR transcoding sessions),
            Transcode Codec EVRC (25 EVRC transcoding sessions),
            Transcode Codec Opus (25 OPUS transcoding sessions),
            Transcode Codec SILK (25 SILK transcoding sessions)
            expires at 06:28:15 Feb 07 2020
            installed at 17:08:15 May 08 2030
```

> **Note:**
>
> Examples in this section are provided for illustration and may not reflect all available features on your system.

Setup Features on an HA Pair

An HA pair requires that you set up identical features on both systems during the same service window. Peers with mismatched features may exhibit unexpected behavior. You should carefully confirm system synchronization at every step.

This procedure uses the designations system-1 as the original active and system-2 as the original standby.

1. Confirm that system-1 and system-2 are healthy and synchronized. First check system health with the **show health** command on both systems to confirm they are in identical states and healthy.
   - Verify that both systems are in identical health with all processes synchronized with the **show health** command.

     > **Note:**
     >
     > The following examples may not accurately portray your Oracle Communications product and version combination.

     ```
     ORACLE-1# show health

                  Media Synchronized           true
                  SIP Synchronized             true
                  REC Synchronized             disabled
                  MGCP Synchronized            disabled
                  H248 Synchronized            disabled
     ```

```
                XSERV Synchronized            disabled
                Config Synchronized           true
                Collect Synchronized          disabled
                RADIUS CDR  Synchronized      true
                Rotated CDRs Synchronized     true
                IPSEC Synchronized            true
                Iked Synchronized             disabled
                Active Peer Address

Redundancy Protocol Process (v3):
                State
Active
                Health                                                100
                Lowest Local Address                    169.254.1.2:9090
                1 peer(s) on 2 socket(s):
                system-2: v3, Standby, health=100, max silence=1050
                       last received from 169.254.1.1 on wancom1:0


ORACLE-2# show health
                Media Synchronized            true
                SIP Synchronized              true
                REC Synchronized              disabled
                MGCP Synchronized             disabled
                H248 Synchronized             disabled
                XSERV Synchronized            disabled
                Config Synchronized           true
                Collect Synchronized          disabled
                RADIUS CDR  Synchronized      true
                Rotated CDRs Synchronized     true
                IPSEC Synchronized            true
                Iked Synchronized             disabled
                Active Peer Address           169.254.2.2

Redundancy Protocol Process (v3):
                State
Standby
                Health                                                100
                Lowest Local Address                    169.254.1.1:9090
                1 peer(s) on 2 socket(s):
                system-1: v3, Active, health=100, max silence=1050
                       last received from 169.254.2.2 on wancom2:0
```

Next, confirm that both the saved and running configurations across both systems are at the same version number. The following example verifies that system-1 and system-2 all share version 5 of their current and running configurations. If the configurations are out of sync, use the **save-config** and **activate-config** commands to fix this.

- Verify that the current configurations are in sync on both system-1 and system-2 with the **display-current-cfg-version** command.

```
ORACLE-1# display-current-cfg-version
Current configuration version is 5

ORACLE-2# display-current-cfg-version
Current configuration version is 5
```

- Verify that the running configurations are in sync on both system-1 and system-2 with the **display-running-cfg-version** command.

```
ORACLE-1# display-running-cfg-version
Running configuration version is 5

ORACLE-2# display-running-cfg-version
Running configuration version is 5
```

2. Self-provision features on system-1 with the **setup entitlements** command. This procedure is explained in the Setup Entitlements task, but DO NOT reboot the system at this point.
3. Save and activate the configuration on system-1.

```
ORACLE-1# save-config
checking configuration
Save-Config received, processing.
waiting for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
ORACLE-1# activate-config
Activate-Config received, processing.
waiting for request to finish
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
```

4. Install License-key enabled features on system-1 with the **licenses** configuration element. This procedure is explained in the Add a license task, but DO NOT reboot the system at this point. Perform a save- and activate- as performed in the previous step.
5. Repeat steps 2 and 4 on system-2, with the exception of saving and activating on system-2. Saving and activating is not relevant on a system that is currently a standby. The system synchronization process include synchronizing configuration changes.

   At the end of this step, identical features are installed, synchronized, and verified both system-1 and system-2.
6. Confirm once again that system-1 and system-2 are synchronized exactly as you did in step 1.
7. Reboot the standby system-2.

```
ORACLE-2# reboot
```

8. Wait for system-2 to startup and synchronize, then confirm that system-1 and system-2 are fully synchronized as explained in step 1.
9. Trigger a switchover from system-1 so that the standby system transitions to active, and vice-versa.

```
ORACLE-1# notify berpd force
```

10. Wait while system-2 transitions to the active state, then confirm that system-1 and system-2 are fully synchronized as explained in step 1.

11. Reboot the newly-standby system-1.

```
ORACLE-1# reboot
```

12. Wait for system-1 to complete rebooting, then confirm that system-1 and system-2 are fully synchronized as explained in step 1.

    At this point both systems should be healthy, synchronized, and contain identical feature configurations.

13. If desired, trigger a switchover between the two systems in the HA node so the originally active system (system-1) assumes the active role again.

Configuration Assistant Operations

When you first log on to the Oracle Communications Session Border Controller (SBC), the system requires you to set the configuration parameters necessary for basic operation. To help you set the initial configuration with minimal effort, the SBC provides the Configuration Assistant. The Configuration Assistant, which you can run from the Web GUI or the Acme Command Line Interface (ACLI), asks you questions and uses your answers to set various parameters for managing and securing call traffic. You can use the Configuration Assistant for the initial set up as well as for subsequent changes that you want to make to the basic configuration.

The SBC provides many additional configuration and management capabilities, which do not require configuration to get the system running. You can set the additional functions to fit your deployment needs through the ACLI Configuration tree after running the Configuration Assistant.

Topics:

- The Configuration Assistant Work Flow
- Configuration Assistant ACLI Navigation Commands

The Configuration Assistant Work Flow

The Oracle Communications Session Border Controller (SBC) Configuration Assistant guides you through the minimum number of steps necessary to make the software operational on the SBC. Each step requires you to enter information or make a choice based on the configuration template that you choose before beginning the configuration work flow.

While the steps displayed in the work flow may vary from one template to another, the behavior and process for using the Configuration Assistant is consistent. Regardless of the template you choose, the Configuration Assistant informs you of any prerequisites that you need to address and allows you to review the configuration and make changes before you Activate the configuration. Go to https://www.oracle.com/technical-resources/documentation/acme-packet.html and see "Configuration Assistant Templates" to get detailed instructions for each template available.

> **Note:**
>
> In the ACLI, the Configuration Assistant work flow uses the word "deployment" to mean "template".

The process for using the Configuration Assistant includes the following steps.

Acquire a Template to Use

You can get templates from the following sources:

- Included in the software image.
- Upload a configuration from another SBC and save it locally.
- Download the latest Oracle Template Package from Oracle and save it locally. Use this method to get updated templates. Available as a compressed file (template-package.tar.gz), the download overwrites the existing templates and re-populates the Select a PBX Template and SIP Template lists. Using SFTP, save the template package to /code/configAssistantUploads.

Save the acquired template or package locally. You will select it in the Select a Deployment step.

Launch the Configuration Assistant

> **Caution:**
>
> Running the Configuration Assistant removes all of the configuration on the SBC, including the High Availability (HA) configuration. You must reconfigure HA after running the Configuration Assistant.

Log on to the SBC using the ACLI and type `run configuration-assistant`. The system displays the list of available deployments (templates).

Select a Template or Configuration for Deployment

When the Configuration Assistant prompts you to select a deployment, it displays the list of PBX side templates first followed by the list of SIP Trunk side templates. Select a PBX template and then select the corresponding SIP Trunk template. To select a template, type the number of the template that you want to use and press Enter.

After you select the templates and press Enter, the Configuration Assistant displays any Warnings, Prerequisites, and Recommendations that you might need to address before proceeding.

Configure the Prerequisites

The template you select may require prerequisites to complete before performing the configuration. When the system detects prerequisites that you need to address, the Configuration Assistant lists them. If setting any of the prerequisites requires a reboot, the system displays a confirmation message about the need to reboot. Type **y** to reboot.

After the reboot the Configuration Assistant re-starts with the most recently selected template and allows you to proceed with the configuration, if all the prerequisites are satisfied. If not, the Configuration Assistant displays the list again and the process repeats.

> **Note:**
>
> If you Quit before the reboot, the Configuration Assistant closes. When you open the Configuration Assistant again, in either the current session or a new one, the system displays a reminder to reboot.

Perform the Configuration

The configuration work flow starts after you complete the prerequisites and the system reboots. As you complete each section of the work flow, the Configuration Assistant displays the next section.

A typical work flow begins with configuring the PBX side followed by the SIP Trunk side. You must complete all of the sections for a new configuration. In an existing configuration, you can move through the sections without changing anything to get go to a section that you want to edit.

Review the Configuration

After you complete last step, the Configuration Assistant, displays the Summary page. From the Summary page, you can go back to edit any of the preceding sections. See Configuration Assistant ACLI Navigation Commands.

If you chose CSR as the certificate provisioning type, the Configuration Assistant displays the Certificate Signing Request with the Summary. Copy or download the CSR and send it to the Certificate Authority.

Apply the Configuration

When you are ready to Save and Activate the configuration, type **s** and press Enter. The Configuration Assistant displays the Epilogue page, which may list actions you need to perform for the PBX side, the SIP Trunk side, or both sides before applying the configuration. After you resolve any outstanding PBX or SIP Trunk actions, the Configuration Assistant displays a confirmation message.

- First-time configuration (new SBC)—The Configuration Assistant displays the Apply Confirmation message. Type **y**, and press Enter. The system applies the configuration and restarts.
- Previous configuration—The Configuration Assistant displays the Apply Confirmation message. Type **y**, and press Enter. The system erases the previous configuration, applies the new configuration, and restarts.

Address the Post-Configuration Tasks

- If you chose CSR as the certificate provisioning type, import the certificate after the system restarts. The name of the certificate-record must be the name of the downloaded file minus the .csr extension. For example, suppose the downloaded CSR file name is TeamsCSR.csr. The name to enter in the certificate-record configuration is TeamsCSR.
- Optional—If you want to customize your SBC deployment, you can access more configuration objects from the ACLI navigation tree. (Configuration, Monitor and Trace, Widgets, and System)

Configuration Assistant ACLI Navigation Commands

Use the following commands to navigate the within the Configuration Assistant while using the Acme Command Line Interface (ACLI).

-—Previous

?—Help. Highlight a parameter and type ? to see a description or requirement of the parameter. For example,for Realm Name the Help states that the realm name must be unique.

.—Clear

q—Quit

d—Display Summary

s—Save

g—Display the configuration

<integer>—The Configuration Assistant numbers each parameter in the template. To edit a parameter, enter its number and press Enter.

User Accounts

In addition to the two factory accounts user and admin, you may also authenticate using local accounts, RADIUS, or TACACS+.

Named SSH Keys

SSH authorized keys take precedence over other authentication methods and account types. For example, if an administrator imported Alice's SSH key into the admin class, then Alice can authenticate with `ssh alice@10.0.0.1` whether or not a local account, TACACS+ account, or RADIUS account exists. Moreover, if a local account, TACACS+ account, or RADIUS account named `alice` exists in the user class but Alice's SSH authorized-key exists in the admin class, Alice can still authenticate as an administrator because SSH keys take precedence over other authentication methods and account types. Conversely, if Alice's SSH key were imported into the user class but a local account, TACACS+ account, or RADIUS account in the admin class were created for Alice, she would by default log in as an ordinary user and not as an administrator. This happens because SSH clients usually try public key authentication before attempting password-based authentication. To authenticate using password-based authentication when public key authentication is an option, use the `-o` option:

```
ssh -o PubkeyAuthentication=no alice@10.0.0.1
```

SSH authorized keys also take precedence over the default factory accounts. If you disable the factory accounts but import an SSH key as the admin user, you can still authenticate with `ssh admin@10.0.0.1` even when factory accounts are disabled.

When removing a user from a system, remember to remove any named SSH keys.

Local User Accounts

The SBC comes with two local, factory accounts for access. System administrators may create additional local accounts for each user or administrator who needs to access the SBC. Local accounts ensure your ability to audit an individual's activity on the SBC.

When creating local accounts, you must specify the username and the user class. Usernames must be unique, and neither `user` nor `admin` may be used.

There are two user classes: `user` and `admin`. Local accounts in the user class have the same access level as the factory user account, and local accounts in the admin class have the same access level as the factory admin account.

After a second administrator account has been created, you may disable the factory user and admin accounts. The SBC requires at least one administrator account. Only administrators may delete accounts, and administrators may not delete their own account. Use the command `factory-accounts` to disable or re-enable the factory accounts.

The file `cli.audit.log` records the timestamp, the local account name, the connecting IP address, and the command run by any user or administrator.

```
2020-10-01 15:35:06.530 TaskID: 0xab7c8710, admin@10.2.2.7 : 'show users'
2020-10-01 15:36:14.112 TaskID: 0xab7c8710, alice@10.2.2.8 : 'show users'
```

Local Accounts and TACACS+

When the `tacacs-authentication-only` attribute is enabled in the `security` configuration element or when the Admin Security entitltement is enabled, authentication to a local account changes when TACACS+ is configured. If a TACACS+ server is configured and available, then authentication uses TACACS+ and the SBC rejects attempts to authenticate to local accounts. If a TACACS+ server is configured but unavailable, the SBC allows authentication to local accounts. This ensures that, when TACACS+ is configured, authentication to local accounts is only possible when the TACACS+ server is down. If no TACACS+ server is configured, local accounts are accessible.

Manage Local Accounts

Use the `local-accounts` command to create, delete, or modify individual accounts. Use the `factory-accounts` command to disable or re-enable the default user and admin accounts.

Create a Local Account

The syntax to add a local account:

```
local-accounts add <username> <class>
```

Usernames must start with a lower case letter or an underscore; use only lower case letters, digits, underscores, or dashes; and not exceed 31 characters. The two options for `<class>` are `user` and `admin`.

1. Create an account.
   To create an account for a user named Jamie:

   ```
   ORACLE# local-accounts add jamie user
   ```

To create an account for an administrator named Jamie:

```
ORACLE# local-accounts add jamie admin
```

2. Enter and confirm the password for the new account.
3. Save and activate the configuration.

Modify the Password of a Local Account

Local administrator accounts may change the password of any local account, but they may not change the password of the factory default accounts.

The syntax to change the password of a local account:

```
local-accounts change-password <username>
```

1. Log in as an administrator.
2. Use the `local-accounts` command to change the password of a local account.

```
local-accounts change-password jamie
```

3. Enter the current password for that local account.
4. Enter and confirm a new password for that local account.

The SBC saves and activates the configuration.

Reset a Local Account Password

The syntax to reset a local account password:

```
local-accounts reset <username>
```

1. Log in as an administrator.
2. Reset a user's password by creating a temporary password.

```
ORACLE# local-accounts reset jamie
```

3. Confirm you want to reset the local account password.
4. Enter and confirm the temporary password for that user.
5. Communicate the temporary password to that user.

The SBC saves and activates the configuration.

The SBC will force the user `jamie` to choose a new password the next time that user logs in.

Delete a Local Account

The syntax to delete a local account:

```
local-accounts delete <username>
```

1. Log in as an administrator.

2. Delete the account.

```
ORACLE# local-accounts delete jamie
```

3. Confirm you want to delete the account.
4. Save and activate the configuration.
5. Delete any saved authorized keys for that user.

```
ORACLE# ssh-key authorized-key delete jamie
```

6. Use the `show users` command to display active sessions.

```
ORACLE# show users
Index     remote-address    IdNum  duration  type         state        User
-------------------------------------------------------------------------------
---
    2 10.0.0.1:59378        7849  00:01:46      ssh       priv *
admin
    1 10.0.0.1:59373        7842  00:01:57      ssh       user
jamie
    0 127.0.0.1             2701  04:17:39  console       user
```

7. Kill any active sessions of the old user.

```
ORACLE# kill ssh 1
Killing ssh session [1]
Successfully killed session [ssh-jamie@10.0.0.1] at index[1]
```

Viewing Local Accounts

To view the local accounts on the SBC, use the `show configuration local-accounts` command.

```
ORACLE# show configuration local-accounts
local-accounts
        user-name                               jamie
        user-class                              user
        user-password                           ******
        last-modified-by                        admin@10.0.0.1
        last-modified-date                      2020-09-28 17:11:38
ORACLE#
```

> **Note:**
>
> The `local-accounts` argument to the show command must be written out in full.

Disable the Default Accounts

If you have created a second administrator account, you can disable the default user and admin accounts.

1. Log in as an administrator.

2. Run the `factory-accounts` command.

```
ORACLE# factory-accounts disable
```

3. Save and activate the configuration.

Re-enable the Default Accounts

If you have disabled the default user and admin accounts, you can re-enable them.

1. Run the `factory-accounts` command.

```
ORACLE# factory-accounts enable
```

2. Save and activate the configuration.

RADIUS Authentication

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

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

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

- The SBC restricts access and requires authentication via the RADIUS server; the SBC communicates with the RADIUS server using either port 1812 or 1645, but does not know if the RADIUS server listens on these ports
- Your RADIUS server provides an alternative method for defining SBC users and authenticating them via RADIUS; the RADIUS server supports the VSA called ACME_USER_CLASS, which specifies what kind of user is requesting authentication and what privileges should be granted. Supported values are `admin` or `user`, and must be lowercase.

The SBC also supports the use of the Cisco Systems Inc.™ Cisco-AVPair vendor specific attribute (VSA). This attribute allows for successful administrator login to servers that do not support the Oracle authorization VSA. While using RADIUS-based authentication, the SBC authorizes you to enter Superuser mode locally even when your RADIUS server does not return the ACME_USER_CLASS VSA or the Cisco-AVPair VSA. For this VSA, the Vendor-ID is 1 and the Vendor-Type is 9. The list below shows the values this attribute can return, and the result of each:

- shell:priv-lvl=15—User automatically logged in as an administrator
- shell:priv-lvl=1—User logged in at the user level, and not allowed to become an administrator
- Any other value—User rejected

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

- What type of authentication you want to use on the SBC

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

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

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

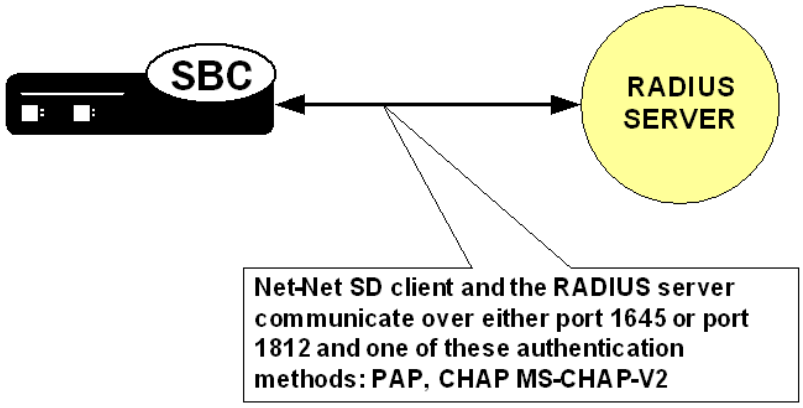
- None—All access denied
- User—Monitoring privileges are granted; your user prompt will resemble ORACLE>
- Admin—All privileges are granted (monitoring, configuration, etc.); your user prompt will resemble ORACLE#

After it selects a RADIUS server, the SBC initiates communication and proceeds with the authentication process. The authentication process between the SBC and the RADIUS server takes place using one of three methods, all of which are defined by RFCs:

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

Note:

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



PAP Handshake

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

PAP Client Request Example

```
Radius Protocol
Code: Access Request (1)
  Packet identifier: 0x4 (4)
  Length: 61
  Authenticator: 0x0000708D00002C5900002EB600003F37
  Attribute value pairs
    t:User Name(1) l:11, value:"TESTUSER1"
      User-Name: TESTUSER1
    t:User Password (2) l:18, value:739B3A0F25094E4B3CDA18AB69EB9E4
    t:NAS IP Address(4) l:6, value:168.192.68.8
      Nas IP Address: 168.192.68.8(168.192.68.8)
    t:NAS Port(5) l:6, value:118751232
```

PAP RADIUS Response

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

CHAP Handshake

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

CHAP Client Request Example

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

CHAP RADIUS Response

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

MS-CHAP-v2 Handshake

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

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

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

MS-CHAP-v2 Client Request Example

Some values have been abbreviated.

```
Radius Protocol
  Code: Access Request (1)
  Packet identifier: 0x5 (5)
  Length: 80
  Authenticator: 0x0000024C000046B30000339F00000B78
  Attribute value pairs
    t:User Name(1) l:11, value:"TESTUSER1"
```

```
      User-Name: TESTUSER1
    t:Vendor Specific(26) l:24, vendor:Microsoft(311)
    t:MS CHAP Challenge(11) l:18, value:0000024C000046B30000339F00000B78
    t:Vendor Specific(26) l:58, vendor:Microsoft(311)
    t:MS CHAP2 Response(25) l:52,
value:00000000024C000046B30000339F00000B78...
    t:NAS IP Address(4) l:6, value:168.192.68.8
      Nas IP Address: 168.192.68.8(168.192.68.8)
    t:NAS Port(5) l:6, value:118751232
```

MS-CHAP-v2 RADIUS Response

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

Management Protocol Behavior

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

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

- SFTP in non-pass-through mode—The “user” or admin accounts are authenticated locally, not via the RADIUS server. For all other accounts, the configured RADIUS servers are used for authentication.

RADIUS Authentication Configuration

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

Global Authentication Settings

To configure the global authentication settings, which apply to your configured authentication type:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

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

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

   - local | radius | tacacs

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

   - pap | chap | ascii | mschapv2

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

   - 1645 | 1812 | 3799

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

   When enabled, the Oracle Communications Session Border Controller ignores RADIUS or TACACS restrictions and allows all users to locally enable Superuser (administrative) mode.

RADIUS Server Settings

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

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

To configure a RADIUS server to use for authentication:

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

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

2. **address**—Set the remote IP address for the RADIUS server. There is no default value, and you are required to configure this address.
3. **port**—Set the port at the remote IP address for the RADIUS server. The default port is set to **1812**. The valid values are:
   - 1645 | 1812 | 3799
4. **state**—Set the state of the RADIUS server. Enable this parameter to use this RADIUS server to authenticate users. The default value is **enabled**. The valid values are:
   - enabled | disabled
5. **secret**—Set the password that the RADIUS server and the Oracle Communications Session Border Controller share. This password is transmitted between the two when the request for authentication is initiated; this ensures that the RADIUS server is communicating with the correct client.
6. **nas-id**—Set the NAS ID for the RADIUS server. There is no default for this parameter.
7. **retry-limit**—Set the number of times that you want the Oracle Communications Session Border Controller to retry for authentication information from this RADIUS server. The default value is **3**. The valid range is:
   - Minimum—1
   - Maximum—5

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

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

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

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

      This parameter has a specific relationship to the global protocol parameter for the authentication configuration, and you should exercise care when setting it. If the authentication method that you set for the RADIUS server does not match the global authentication protocol, then the RADIUS server is not used. The Oracle Communications Session Border Controller simply overlooks it and does not send authentication requests to it. You can enable use of the server by changing the global authentication protocol so that it matches.
13. Use the **management-servers** attribute to identify one or more RADIUS servers available to provide AAA services.

    Servers are identified by IP address, participate in the configured **management-strategy**, and must have been previously configured as described above.

    The following example identifies three available RADIUS servers. The list is delimited by left and right parentheses, and list items are separated by space characters.

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

    The following example deletes the current list.

    ```
    ORACLE(authentication)# management-servers ()
    ORACLE(authentication)#
    ```
14. Save your work and activate your configuration.

TACACS+

TACACS+ (Terminal Access Controller Access Control System Plus) is a protocol originally developed by Cisco Systems, and made available to the user community by a draft RFC, *TACACS+ Protocol, Version 1.78* (draft-grant-tacacs-02.txt). TACACS+ provides AAA

(Authentication, Authorization, and Accounting) services over a secure TCP connection using Port 49.

TACACS+ Overview

Like Diameter and Remote Authentication Dial-In User Service (RADIUS), SBC uses a client-server model in which a Network Access Server (NAS) acts in the client role and a TACACS+ equipped device (a daemon in TACACS+ nomenclature) assumes the server role. For purposes of the current implementation, the SBC functions as the TACACS+ client. Unlike RADIUS, which combines authentication and authorization, TACACS+ provides three distinct applications to provide finer grade access control.

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

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

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

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

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

Details on the TACACS+ functions and configuration can be found in the *Oracle Communications Session Border Controller ACLI Configuration Guide.*.

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

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

Other relevant documents include

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

RFC 1334, *PPP Authentication Protocols* .

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

> **Note:**
>
> TACACs documentation in this guide excludes per-message definitions that duplicate IETF standards documentation.

TACACS+ Authentication

The Oracle Communications Session Border Controller (SBC) uses Terminal Access Controller Access-Control System Plus (TACACS+) authentication services solely for the authentication of user accounts. Administrative users must be authenticated locally by the SBC.

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

ASCII Log In

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

PAP Log In

Password Authentication Protocol (PAP) is defined in RFC 1334, *PPP Authentication Protocols*. PAP offers minimal security because passwords are transmitted as unprotected clear text. PAP log in differs from ASCII log in because the username and password are transmitted to the authenticating peer in a single authentication packet, as opposed to the two-step prompting process used in ASCII log in.

CHAP Log In

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

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

Authentication Message Exchange

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

START and CONTINUE packets are always sent by the Oracle Communications Session Border Controller, the TACACS+ client. START packets initiate an authentication session, while CONTINUE packets provide authentication data requested by the TACACS+ daemon. In response to every client-originated START or CONTINUE, the daemon must respond with a

REPLY packet. The REPLY packet contains either a decision (pass or fail), which terminates the authentication session, or a request for additional information needed by the authenticator.

Restricting Logon to TACACS

For deployments that include TACACS authentication, the Oracle Communications Session Border Controller (SBC) allows the user to configure a restriction that prevents users from logging into the system using mechanisms other than TACACS. The function that manages this restriction evaluates the availability of TACACS infrastructure and allows alternate login mechanisms if TACACS servers are unavailable due to either network or server issues.

Users who wish to restrict SBC login authentication to TACACS enable the **authentication** element's **tacacs-authentication-only** parameter. If there are two or more TACACS+ servers configured and the SBC either fails to establish a connection or an exiting connection fails, it tries to connect to the next available server. If there are two or more TACACS+ servers configured, then the system shall try to connect to all of them for a single login attempt and determine that they are all unavailable before falling back to using local login authentication.

> **Note:**
>
> The **tacacs-authentication-only** parameter is not functional on systems that have the Admin Security feature enabled.

The SBC uses all of the following criteria to determine that a TACACS+ server is available for login authentication:

- The system is able to establish a TCP connection to a TACACS+ server, AND
- TACACS+ server is responsive (e.g., no timeouts), AND
- TACACS+ server responds with an authentication PASS or FAIL status

The SBC uses any of the following criteria to determine that a TACACS+ server is unavailable for login authentication:

- TACACS+ server is unreachable, OR
- TACACS+ server response is not received (e.g., timeout), OR
- TACACS+ server responds with an authentication ERROR status

For a login attempt that reach a TACACS server but subsequently fails, the SBC rejects the login attempt with a standard login failure and records the login attempt in the Audit log.

In addition to the above and when **tacacs-authentication-only** is enabled, the SBC responds to authentication attempts that fail to reach a TACACS server by generating an SNMP trap and an associated alarm. The system also applies both the clear-alarm and clear-trap logic when TACACS again becomes available.

Traps and Associated Alarms

Traps supporting this feature, in ap-smgmt.mib, include indications that local authentication was used, and that the condition that caused local authentication is cleared.

Table 2-1 Trap and Clear Trap for TACACS Authentication Failure

Trap	Description
apSysMgmtTacacsDownLocalAuthUsedTrap 1.3.6.1.4.1.9148.3.2.6.0.88	This trap is generated when a user remotely logs into a system configured for TACACS+ authentication and is authenticated locally by the system because all of the configured and enabled TACACS+ servers have become unreachable or unresponsive
apSysMgmtTacacsDownLocalAuthUsedClearTrap 1.3.6.1.4.1.9148.3.2.6.0.89	This trap is generated when a user remotely logs into a system configured for TACACS+ authentication and is successfully authenticated (i.e., access accepted or denied) remotely by a configured and enabled TACACS+ server.

The alarm associated with this trap is APP_ALARM_TACACS_DOWN_LOCAL_AUTH_USED (327721), shown below.

```
ID      Task    Severity       First Occurred        Last Occurred
327721  69      3         2017-10-31 07:17:37  2017-10-31 07:17:37
Count   Description
1       User Bob authenticated locally due to unavailability of TACACS+
server(s)
```

Process System and Audit Log Entries

This feature writes entries into the ACLI process log files (e.g., log.acliSSH) to record each occurrence of a user remotely logging into the system because the TACACS+ servers are unreachable or unresponsive.

```
Oct 31 13:55:56.280 [AUTH] (0) authenticate_secure_user: user 'user'
authenticated locally due to unavailability of TACACS+ server.
```

This feature also writes a syslog message to record each occurrence of a user remotely logging into the system because the TACACS+ servers are unreachable or unresponsive.

```
Oct 31 13:55:56 172.41.3.90 acliSSH0@SBC1: AUTH[] authenticate_secure_user:
user 'admin' authenticated locally due to unavailability of TACACS+ server.
```

In addition, the system creates an audit log for every login attempt.

```
2017-10-31 13:55:56,ssh-
user@172.41.3.90:34362,security,login,success,authentication,,.
2017-10-31 13:55:56,ssh-
user@172.41.3.90:34362,security,login,failure,authentication,,.
```

TACACS+ Authorization

The Oracle Communications Session Border Controller uses Terminal Access Controller Access-Control System Plus (TACACS+) services to provide administrative authorization. With TACACS+ authorization enabled, each individual ACLI command issued by an admin user is authorized by the TACACS+ authorization service. The Oracle Communications Session

Border Controller replicates each ACLI command in its entirety, sends the command string to the authorization service, and suspends command execution until it receives an authorization response. If TACACS+ grants authorization, the pending command is executed; if authorization is not granted, the Oracle Communications Session Border Controller does not execute the ACLI command, and displays an appropriate error message.

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

TACACS+ Authorization Command & Arguments Boundary

When sending the Authorization query to the TACACS+ server, by default the SBC sends everything typed at the ACLI in the `cmd` parameter. For commands, this includes the command plus all of its arguments (for example, `cmd=show interfaces brief`). For configurations, this includes the full path of the configuration element plus its attributes and values (for example, `cmd=configure terminal security authentication type tacacs`). In the TACACS+ query, the `cmd-arg` parameter is set to `<cr>`.

The parameter **tacacs-authorization-arg-mode** changes this default behavior. Parameter values, and their behavior with respect to splitting the entry values include:

- **disabled**—Retain default behavior.
- **enabled**—Applies the fully split cmd and cmd-arg behavior to all ACLI input through TACACS+, with the exception of the **show** command. Behavior includes:
  - All **show** commands follow the pattern: `cmd=show <required-word>`, `cmd-arg=<optional-word1>`, `cmd-arg=<optional-word2>`, and so on. If no optional words are used, the `cmd-arg` parameter is set to `<cr>`.
    For example:

    ```
    cmd=show uptime, cmd-arg=<cr>
    cmd=show tacacs, cmd-arg=stats
    cmd=show running-config, cmd-arg=authentication, cmd-arg=to-file, cmd-arg=auth.conf
    ```

  - All other commands follow the pattern: `cmd=<first-word>`, `cmd-arg=<second-word>`, `cmd-arg=<third-word>`, and so on. If the command is a single-word command, the `cmd-arg` parameter is set to `<cr>`.
    For example:

    ```
    cmd=verify-config, cmd-arg=<cr>
    cmd=configure, cmd-arg=terminal
    cmd=ssh-key, cmd-arg=authorized-key, cmd-arg=import, cmd-arg=admin, cmd-arg=admin
    ```

  - All configurations follow one of two patterns:
    - For navigating the ACLI: `cmd=<full-path>`, `cmd-arg=<cr>`.
    - For setting an attribute to a value: `cmd=<full-path> <attribute>`, `cmd-arg=<value>`

For example, the following ACLI interaction produces this sequence of TACACS+ queries.

```
ORACLE# conf term
ORACLE(configure)# security
ORACLE(security)# authentication
ORACLE(authentication)# select
ORACLE(authentication)# type tacacs


cmd=configure, cmd-arg=terminal
cmd=configure terminal security, cmd-arg=<cr>
cmd=configure terminal security authentication, cmd-arg=<cr>
cmd=configure terminal security authentication select, cmd-arg=<cr>
cmd=configure terminal security authentication select type, cmd-
arg=tacacs
```

- **enabled-for-show**—Applies the fully split cmd and cmd-arg behavior to all ACLI input through TACACS+, including the **show** command.
  - All **show** commands follow the pattern: `cmd=show`, `cmd-arg=<required-word>`, `cmd-arg=<optional-word1>`, `cmd-arg=<optional-word2>`, and so on. If no optional words are used, the `cmd-arg` parameter is set to `<cr>`.
    For example:

    ```
    cmd=show, cmd-arg=uptime
    cmd=show, cmd-arg=tacacs, cmd-arg=stats
    cmd=show, cmd-arg=running-config, cmd-arg=authentication, cmd-arg=to-
    file, cmd-arg=auth.conf
    ```

  - All configurations follow one of two patterns presented above in the explanation of the **enabled** value.

Authorization Message Exchange

All Terminal Access Controller Access-Control System Plus (TACACS+) authorization packets consist of a common header and a message body. Authorization packets are of two types: REQUEST and RESPONSE.

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

TACACS+ Authorization for the REST API

TACACS+ users can access the SBC from the REST interface and their actions can be restricted by TACACS+ authorization and recorded with TACACS+ accounting. First enable the **rest-authorization-accounting** attribute in the **authentication** element. After saving and activating the configuration, you can define permitted actions on your TACACS+ server. TACACS+ authorization queries for the REST API follow the pattern:

```
<method> <path-parameters> [<query-parameters>]
```

Test each API thoroughly to understand the TACACS+ authorization query.

TACACS+ Authorization for all REST APIs

The first two path parameters of the HTTP request (`rest` and the version) are excluded from the path parameters of the TACACS+ authorization query. The following HTTP requests generate the following `cmd` strings in the TACACS+ authorization query.

```
POST /rest/v1.1/configuration/lock
cmd=POST configuration lock
```

```
POST /rest/v1.2/configuration/unlock
cmd=POST configuration unlock
```

TACACS+ users who need to modify the configuration through the REST API must be authorized to lock and unlock the configuration. If the configuration lock is required to access a specific API, the SBC first checks whether the TACACS+ user has the configuration lock. If the user does not possess the configuration lock, the request is rejected immediately without sending a TACACS+ authorization query.

The token API, the supported-versions API, and all the statistics APIs are excluded from TACACS+ authorization and accounting.

- `/rest/{version}/auth/token`
- `/rest/api/supportedversions`
- `/rest/{version}/statistics/*`

The accounting format matches the authorization format.

Admin APIs

For the REST API under the Admin section, TACACS+ authorization is enabled only for POST requests.

```
POST /rest/v1.2/admin/switchover
cmd=POST admin switchover
```

```
GET /rest/v1.2/admin/test/lrt
cmd=GET admin test lrt
```

TACACS+ authorization is not enabled on the `/admin/asyncstatus` API.

Configuration APIs

For the REST APIs under the Configuration section, the `elementType` value is included in the TACACS+ authorization query.

```
GET /rest/v1.2/configuration/configElements?elementType=access-control
cmd=GET configuration configElements access-control
```

When creating or updating a configuration element, the value of the XML `<elementType>` node is sent as the final word of the TACACS+ authorization query. This allows fine grained control over which configuration elements the TACACS+ user can create or modify.

```
POST /rest/v1.2/configuration/configElements
cmd=POST configuration configElements access-control
```

Other URL query parameters are not sent as part of the TACACS+ authorization query.

```
DELETE /rest/v1.2/configuration/configElements?elementType=access-
control&realm-id=core&source-address=10.9.9.0/24&application-
protocol=SIP&transport-protocol=TCP
cmd=DELETE configuration configElements access-control
```

Configuration Management APIs

For the REST APIs under the Configuration Management section, the `action` query parameter and value are included in the TACACS+ authorization query.

```
PUT /rest/v1.2/configuration/management?action=save
cmd=PUT configuration management action save
```

```
PUT /rest/v1.2/configuration/management?action=verify
cmd=PUT configuration management action verify
```

```
POST /rest/v1.2/configuration/management?action=activate
cmd=POST configuration management action activate
```

```
POST /rest/v1.2/configuration/management?
action=backup&backupSource=running&filename=bkupRunConfig.gz
cmd=POST configuration management action backup
```

```
DELETE /rest/v1.2/configuration/management/files?
filename=backupRunningConfig.gz
cmd=DELETE configuration management files backupRunningConfig.gz
```

System APIs

The TACACS+ queries for REST APIs under the System section follow the pattern of the Configuration APIs.

```
POST /rest/v1.2/system/features
cmd=POST system features


DELETE /rest/v1.2/system/features
cmd=DELETE system features


PUT /rest/v1.2/system/features
cmd=PUT system features


GET /rest/v1.2/system/storage
cmd=GET system storage


GET /rest/v1.2/system/information
cmd=GET system information
```

TACACS+ Accounting

The Oracle Communications Session Border Controller uses Terminal Access Controller Access-Control System Plus (TACACS+) accounting to log administrative actions. With accounting enabled, each individual ACLI command executed by an admin user is logged by the accounting service.

Accounting Message Exchange

All Terminal Access Controller Access-Control System Plus (TACACS+) accounting packets consist of a common header and a message body. Accounting packets are of two types: REQUEST and REPLY.

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

A TACACS+ accounting session proceeds as follows.

1. Immediately following successful authorization of an admin user, the SBC sends an accounting REQUEST START packet.
2. The daemon responds with an accounting REPLY packet, indicating that accounting has started.
3. For each ACLI command executed by an admin user, the SBC sends an accounting REQUEST WATCHDOG packet requesting accounting of the ACLI command. As the SBC sends the WATCHDOG only after an admin user's access to the ACLI command is authorized, the accounting function records only those commands executed by the user, not those commands for which authorization was not granted.

4. The daemon responds with an accounting REPLY packet, indicating that the ACLI operation has been recorded by the accounting function.
5. Steps 3 and 4 are repeated for each authorized ACLI operation.
6. Immediately following logout (or timeout) of an admin user, the SBC sends an accounting REQUEST STOP packet.
7. The daemon responds with an accounting REPLY packet, indicating that accounting stopped.

TACACS+ Configuration

Configuration of Terminal Access Controller Access-Control System Plus (TACACS+) consists of the following steps.

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

Enable TACACS+ Client Services

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

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

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

2. **type** — Configure this parameter to specify the authentication protocol. The default value is **local**. Specify **tacacs** to enable the TACACS+ AAA protocol.
   - **diameter** — DIAMETER authentication (not yet supported)
   - **local** — authentication determinations are referred to a local database (default)
   - **radius** — RADIUS authentication
   - **tacacs** — TACACS+ authentication
3. **tacacs-authentication-only**— Enable this parameter to require remote authentication via TACACS+ unless the TACACS+ infrastructure is not available.
   - **disabled** (default)
   - **enabled**
4. **tacacs-authorization**— Configure this parameter to enable or disable command-based user authorization. The default value is **enabled** when the value of **type** is **tacacs**.
   - **disabled**
   - **enabled** (default)
5. **tacacs-authorization-arg-mode** — Configure this parameter to enable or disable sending TACACS+ authorization commands and their arguments separately to the TACACS+ server. Select **enabled-include-show** to send show commands separately. The default value is **disabled**.
   - **disabled** (default)
   - **enabled**

- **enabled-include-show**

6. **tacacs-accounting** — Configure this parameter to enable or disable accounting of admin ACLI operations. The default value is **enabled** when the value of **type** is **tacacs**.
   - **disabled**
   - **enabled** (default)
7. **server-assigned-privilege** — Configure this parameter to enable or disable a proprietary TACACS+ variant that, after successful user authentication, adds an additional TACACS+ request/reply exchange. During the exchange, the Security Gateway requests the privilege level of the newly authenticated user. In response, the TACACS+ daemon returns the assigned privilege level, either user or admin. Set this attribute to **enabled** to initiate the proprietary variant behavior. User accounts are denied access to the **enabled** command, thus barring them from configuration level commands. The default value is **disabled** (no privilege level information is exchanged).
   - **disabled** (default)
   - **enabled**
8. **management-strategy** — Configure this parameter to identify the selection algorithm used to choose among multiple available TACACS+ daemons. Retain the default value of **hunt** when only a single daemon is available.
   - **hunt** (default) — for the first transaction the Security Gateway selects the initially configured TACACS+ daemon. When that daemon is online and operational, the Security Gateway directs all AAA transactions to it. Otherwise, the Security Gateway selects the second-configured daemon. If the first and second daemons are offline or non-operational, the next-configured daemon is selected, and so on through the group of available daemons.
   - **roundrobin** — for the first transaction the Security Gateway selects the initially configured TACACS+ daemon. After completing the first transaction, it selects each daemon in order of configuration — in theory, evenly distributing AAA transactions to each daemon over time.
9. Type **done** to save your configuration.

Specify TACACS+ Servers

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

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

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

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

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

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

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

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

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

   Select enabled (the default) or disabled.

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

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

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

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

6. Retain the default value for the **authentication-methods** attribute to specify support for all TACACS+ authentication methods (pap, chap, and ascii).

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

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

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

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

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

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

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

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

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

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

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

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

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

    Servers are identified by IP address, participate in the configured **management-strategy**, and must have been previously configured as described above.

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

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

    The following example deletes the current list.

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

Managing TACACS+ Operations

Terminal Access Controller Access-Control System Plus (TACACS+) management is supported by the following utilities.

TACACS+ MIB

An Oracle proprietary MIB provides external access to Terminal Access Controller Access-Control System Plus (TACACS+) statistics.

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

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

apSecuritysTacacsPlusStats Table (1.3.6.1.4.1.9148.3.9.9.4)

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

SNMP Trap

SNMP traps are issued when

- a Terminal Access Controller Access-Control System Plus (TACACS+) daemon becomes unreachable
- an unreachable TACACS+ daemon becomes reachable
- an authentication error occurs
- an authorization error occurs

TACACS+ Faults

The Oracle Communications Session Border Controller (SBC) supports (TACACS+) traps to notify you of operational status. Traps from the apSysMgmt tree include:

- apSysMgmtTacacsDownTrap (1.3.6.1.4.1.9148.3.2.6.0.78) - Generated when a TACACS+ server becomes unreachable.
- apSysMgmtTacacsDownClearTrap (1.3.6.1.4.1.9148.3.2.6.0.79) - Generated when a TACACS+ server that was unreachable becomes reachable.

The SBC searches for a TACACS+ server until it finds an available one and then stops searching. However, in the TACACS+ SNMP implementation, SNMP expects the SBC to make connection attempts to all servers.

- When there is only one TACACS+ server and that server goes down, the SBC behaves normally, sending a apSysMgmtTacacsDownTrap trap when the server goes down, and a apSysMgmtTacacsDownClearTrap trap when the server comes back up.
- When there is more than one TACACS+ server and the active server goes down, an apSysMgmtTacacsDownTrap trap is sent, indicating that some servers are down and the next server is tried.
  - If all servers fail, an apSysMgmtTacacsDownTrap is sent indicating that all servers are down.
  - If one of the servers comes back up while the rest are still down, an apSysMgmtTacacsDownTrap is sent indicating that some servers are still down.

Traps from the apSecurity tree include:

- apSecurityTacacsFailureNotification (1.3.6.1.4.1.9148.3.9.3.1.0.4) - Generated when the system detects TACACS daemon reachability changes as well as TACACS authentication and authorization errors.
- apSecurityTacacsDownLocalAuthUsedTrap (1.3.6.1.4.1.9148.3.9.3.9.0.1) - Generated when a user remotely logs into a system configured for TACACS+ authentication and is authenticated locally by the system because all of the configured and enabled TACACS+ servers have become unreachable or unresponsive
- apSecurityTacacsDownLocalAuthUsedClearTrap (1.3.6.1.4.1.9148.3.9.3.9.0.2) - Generated when a user remotely logs into a system configured for TACACS+ authentication and is successfully authenticated (i.e., access accepted or denied) remotely by a configured and enabled TACACS+ server.

ACLI show Command

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

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

TACACS+ Logging

All messages between the Oracle Communications Session Border Controller and the Terminal Access Controller Access-Control System Plus (TACACS+) daemon are logged in a clear text format, allowing an admin user to view all data exchange, except for password information.

TACACS+ over IKEv2/IPsec

You can configure the SBC to connect to a TACACS server over an IKEv2/IPsec secured connection. This communication must occur over the management interface wancom0. The **ikev2-ipsec-wancom0-params** element enables this configuration.

Before configuring the **ikev2-ipsec-wancom0-params** element, make sure you have an IPsec license installed. Then create the default **ike-config** element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-config
ORACLE(ike-config)# select
ORACLE(ike-config)# done
```

This procedure ensures that IKEv2 is set globally on the SBC.

When booting up, the SBC reads the **ikev2-ipsec-wancom0-params** configuration element. If this element does not exist, no action is taken. If the element exists and **auto** is set to **ondemand**, an IKEv2/IPsec connection is set up between the SBC and the peer specified in the **remoteip** attribute. The SBC can also act as a responder.

The SBC can authenticate to its IKEv2 peer with an X.509 certificate or a pre-shared key (PSK). To use a PSK:

1. Set the **authby** attribute to **secret**.
2. Set the **shared-password** attribute to your PSK.

To use an X.509 certificate:

1. Review the TLS chapter in the *Configuration Guide*.
2. Create a **certificate-record** for the SBC.
3. Under **security**, **ike**, configure an **ike-certificate-profile** for the SBC.

> **Note:**
>
> The **identity** attribute should match the Subject Alternative Name of the SBC's end-entity certificate.

4. Under **security**, **ikev2-ipsec-wancom0-params**, set **local-certificate-profile-identity** to the **identity** attribute of your previously configured **ike-certificate-profile** for the local peer.
5. Set **authby** to **rsasig**.
6. Set **remote-certificate-identity** to the identity (email address, IP address, FQDN, DNS, or URI) of the remote peer.

> **Note:**
>
> The value should match the Subject Alternative Name of the remote peer's certificate.

Use the following commands to monitor the IKEv2/IPsec connections to the SBC's wancom0 port:

```
show security ipsec wancom0 <sad | spd | tunnels>
show security ike wancom0 <error-stats | sad>
```

Secure wancom0 with IKEv2/IPsec

You can secure TACACS or RADIUS authentication to the management wancom0 interface by tunneling your authentication protocol over IKEv2/IPsec.

1. Navigate to the **ikev2-ipsec-wancom0-params** element.

   ```
   ORACLE# conf term
   ORACLE(configure)# security
   ORACLE(security)# ikev2-ipsec-wancom0-params
   ORACLE(ikev2-ipsec-wancom0-params)#
   ```

2. **name**—Provide a name for this configuration element.

   For example:

   ```
   ORACLE(ikev2-ipsec-wancom0-params)# name SecureTacacs
   ```

3. **remoteip**—Enter the IPv4 or IPv6 address of the remote peer.
4. **remotesubnet**—Enter the IPv4 or IPv6 subnet of the remote peer.

   Omit the number of bits for an exact match. For example, 10.1.1.1 is the same thing as 10.1.1.1/32.
5. **remoteproto**—Keep the default value of ALL or select the remote peer's transport protocol that will be protected within the IPsec tunnel.

   If running TACACS+ over IKEv2/IPsec, you can select TCP because TACACS+ uses TCP as its transport protocol.
6. **remoteport**—Keep the default 0 to match any port or select the port which the remote peer will use to communicate within the IPsec tunnel.

   The default TACACS+ port is 49.
7. **localip**—Confirm the IP address is the SBC's wancom0 IP address.
8. **localsubnet**—Enter the subnet of the local peer.

   This value defaults to a /32 for IPv4 or a /128 for IPv6.
9. **localproto**—Keep the default value of ALL or select the transport protocol of the local peer.

   This is the transport protocol you're going to protect within the tunnel. If running TACACS+ over IKEv2/IPsec, select TCP because TACACS+ uses TCP as its transport protocol. This should be the same value as **remoteproto**.
10. **localport**—Keep the default 0 or select the port that the local peer will communicate on.

    Enter 0 to use ephemeral ports when the SBC acts as a TACACS+ client.
11. **auto**—Keep the default **ondemand** to establish IKEv2/IPsec tunnels on demand.
12. Confirm that the remote peer accepts the default algorithms specified in the **ike-algorithms**, **ipsec-protocol**, and **ipsec-algorithms** attributes.
13. **authby**—Specify how the two peers should authenticate to each other.

    Use **rsasig** for X.509 certificates and use **secret** for a PSK.
14. **shared-password**—If you set **authby** to **secret**, enter a passphrase.
15. **local-certificate-profile-identity**—If you set **authby** to **rsasig**, specify the **identity** attribute of your previously configured **ike-certificate-profile** element for the local peer.
16. **remote-certificate-identity**—If you set **authby** to **rsasig**, specify the email address, IP address, FQDN, DNS, or URI of the remote peer.

> **Note:**
>
> This value should match the Subject Alternate Name of the peer's certificate.

Customizing Your ACLI Settings

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

Disabling the Second Login Prompt

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

Disabling the Second Login Prompt Configuration

You disable the second login prompt in the authentication configuration.

To disable the second login prompt:

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

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

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

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

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

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

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

Persistent ACLI more Parameter

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

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

Persistent ACLI more Parameter Configuration

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

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

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

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

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

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

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

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

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

Customize the Log On Banner Message

When you want to get messaging to your Oracle Communications Session Border Controller (SBC) users, you can put your message into the optional log on banner. The log on banner is a text box that displays before the system asks users for their log on credentials.

To get your message into the log on banner, copy a text file to the log on banner field. You must name the file /code/banners/banner.txt and save it in the /code/banners directory. If that directory does not already exist on your system, the SBC creates it upon boot up.

There is no character limit. The the banner field adjusts to the quantity of text.

3
System Configuration

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

The following configurations are explained in this chapter:

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

Virtual SBCs

Operating the Oracle Communications Session Border Controller (SBC), as well as Oracle's other session delivery products, as a VM introduces configuration requirements that define resource utilization by the virtual machine. The applicable configuration elements allow the user to optimize resource utilization based on the application's needs and VM resource sharing. See your product and software version's Release Notes to verify your product's support for deployment as a virtual machine.

VM deployment types include:

- A standalone (not orchestrated) instance SBC operating as a virtual machine running on a hypervisor
- Virtual Machine(s) deployed within private or public Cloud environments

Standalone SBC VM deployment instances supported for all platforms. Support within an orchestrated environment is dependent on orchestrator and SBC version. High Availability configurations are supported by both deployment types.

SBC configuration for VM includes settings to address:

- **media-manager** — Set media manager configuration elements to constrain bandwidth utilization, based on traffic type. See Media Manager Configuration for Virtual Machines in the Realms and Nested Realms Chapter.

- **system-config**, [*core configuration parameters*] — Set these parameters to specify CPU resources available to DoS, forwarding and transcoding processes. This configuration applies to initial deployment and tuning tasks. You may need to change the default core configuration for functionality purposes during deployment; you may decide to change core configuration for performance purposes after deployment.
- **system-config**, [**use-sibling-core-datapath**] — Enable this parameter to allow the SBC to utilize the underlying platform's CPU hyperthreading (also called SMT) capabilities for datapath cores, including Forwarding, DoS and transcoding cores. Considerations include your environment's support of the feature and its impact on your implementation. A key consideration beyond support is the ability of your platform to provide information on sibling CPUs to the SBC. The SBC supports hyperthreading of signaling cores without additional configuration.

VLAN Support

Oracle recommends that you evaluate the VLAN support of your deployment's hypervisor and interface I/O mode before implementation to ensure secure support for the transmission and receiving of VLAN-tagged traffic. Please consult your hypervisor's vendor documentation for details.

Note that when you configure a VLAN, the SBC requires VLAN tags to be included in the packets delivered to and from the VM.

Hypervisor and cloud platform and resource requirements are version-specific. Refer to your *Release Notes* for applicable requirements, recommendations and caveats for qualified platforms.

You configure VM support for the SBC and the Session Router identically. This documentation refers to the SBC, but applies equally to the Session Router.

CPU Core Configuration

You can configure CPU core settings using **system-config** parameters. This configuration is based on the specific needs of individual implementations. These parameters allow you to set and change the number of cores you want to assign to forwarding, DoS, and transcoding functionality. The system determines which cores perform those functions automatically.

You can determine and manage your core configuration based on the services you need. The system allocates cores to signaling upon installation. You can add forwarding cores to match your needs for handling media. You can also add DoS and transcoding cores if you need those functions in your deployment. If you want to reduce the size of your SBC deployment footprint or if you do not need the maximum number of cores and amount of memory available, you can deploy the SBC virtually with fewer cores and memory requirements. For smaller scale deployments, the VNF software supports a deployment with 2 virtual cores, 2 GB RAM, 20GB storage, and 2 interfaces.

Note the following:

- By default, core 0 is always set to signaling.
- The system selects cores based on function. You cannot assign core functions.
- The system sets unassigned cores to signaling, with a maximum of 24.

> **Note:**
>
> Your hyperthreading configuration may impact these assignments.

- You must reboot the system for core configuration changes to take effect.

When you make core assignments, the (SBC) provides an error message if the system detects an issue. In addition, the system performs a check when you issue the **verify-config** command to ensure that the total number of forwarding, plus DOS, plus transcoding cores does not exceed the maximum number of physical cores. After you save and activate a configuration that includes a change to the core configuration, the system displays a prompt to remind you that a reboot is required for the changes to take place.

You can verify core configuration from the ACLI, using the **show datapath-config** command or after core configuration changes during the save and activation processes. The SBC uses the following lettering (upper- and lower-case) in the ACLI to show core assignments:

- S - Signaling
- D - DoS
- F - Forwarding
- X - Transcoding

When using hyperthreading, which divides cores into a single physical (primary) and a single logical (secondary) core, this display may differ. SBC rules for displaying cores include:

- Physical cores (no hyperthreading) in upper-case letters
- "Primary" hyperthreaded sibling cores in upper-case letters
- "Secondary" hyperthreaded sibling cores in lower-case letters
- Stale (unused) hyperthreaded cores using the lower-case letter "n"

The **system-config** element includes the following parameters for core assignment:

- **dos-cores**— Sets the number of cores the system must allocate for DOS functionality. A maximum of one core is allowed.
- **forwarding-cores**—Sets the number of cores the system must allocate for the forwarding engine.
- **transcoding-cores**—Sets the number of cores the system must allocate for transcoding. The default value is 0.
- **use-sibling-core-datapath**—Enables the SBC to utilize the platform's SMT capability, impacting how the SBC uses sibling cores.

The SBC does not have a maximum number of cores, but your deployment does, based on host resources. The system checks CPU core resources before every boot, as configuration can affect resource requirements. Examples of such resource requirement variations include:

- There are at least 2 CPUs assigned to signaling (by the system).
- If DoS is required, then there are at least 1 CPU assigned to forwarding and 1 to DoS.
- If DoS is not required, then there is at least 1 CPU assigned to forwarding.

> **Note:**
>
> Poll mode drivers, including vmxnet3, failsafe, MLX4 and Ixgbvf, only support a number of rxqueues that is a power of 2. When using these drivers, you should configure the number of forwarding cores to also be a power of 2. If there is a mismatch, the system changes the number of forwarding cores that it uses to the nearest power of 2 value. The remaining cores become stale; stale cores remain reserved by the system, but are not used.

The system performs resource utilization checks every time it boots for CPU, memory, and hard-disk to avoid configuration and resource conflicts.

Core configuration is supported by HA. For HA systems, resource utilization on the backup must be the same as the primary.

> **Note:**
>
> The hypervisor always reports the datapath CPU usage as fully utilized. This isolates a physical CPU to this work load, but may cause the hypervisor to generate a persistent alarm indicating that the VM is using an excessive amount of CPU. The alarm may trigger throttling. Oracle recommends that you configure the hypervisor monitoring appropriately, to avoid throttling.

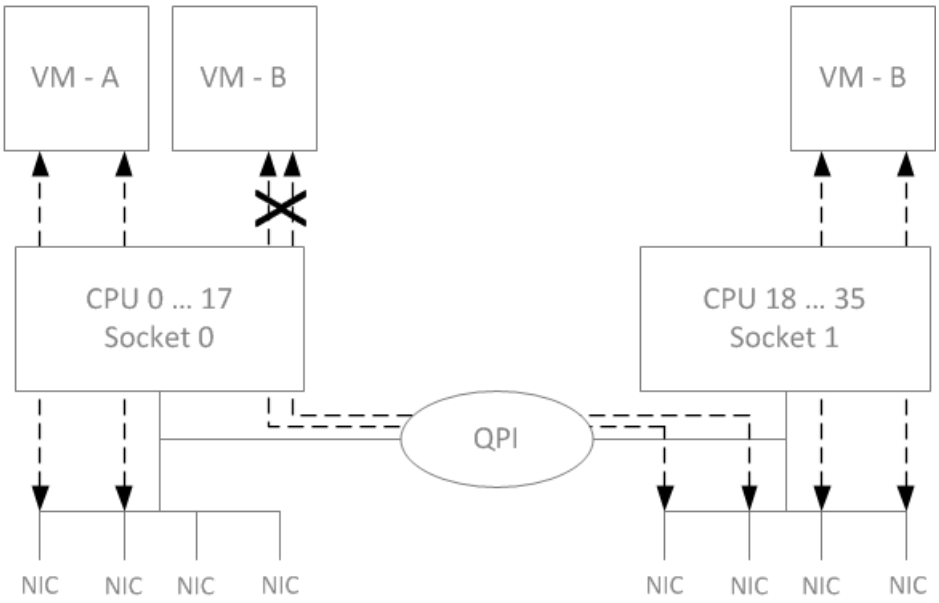
In HA environments, when the primary node's core configuration changes, the SBC raises an alarm to warn that a reboot is required. After the configuration syncs, the secondary node raises the same alarm to warn that a reboot is required.

Host Hypervisor CPU Affinity (Pinning)

Many hardware platforms have built in optimizations related to VM placement. For example, some CPU sockets may have faster local access to Peripheral Component Interconnect (PCI) resources than other CPU sockets. Users should ensure that VMs requiring high media throughput are optimally placed by the hypervisor, so that traversal of cross-domain bridges, such as QuickPath Interconnect (QPI), is avoided or minimized.

Some hypervisors implement Non-Uniform Memory Access (NUMA) topology rules to automatically enforce such placements. All hypervisors provide manual methods to perform CPU pinning, achieving the same result.

The diagram below displays two paths between the system's NICs and VM-B. Without configuring pinning, VM-B runs on Socket 0, and has to traverse the QPI to access Socket 1's NICs. The preferred path pins VM-B to Socket 1, for direct access to the local NICs, avoiding the QPI.



Oracle recommends you configure CPU affinities on the hypervisor to ensure mapping from only one virtual CPU to each physical CPU core. Learn how to configure CPU affinity and pin CPUs from your hypervisor documentation.

> **Note:**
>
> The SBC relies on multiple queues on virtual NICs to scale session capacity. Multiple queues enable the SBC to scale through multiple forwarding cores. This configuration is platform dependent: physical NIC, Hypervisor, virtual NIC, and vSwitch.

Support for Hyperthreading Datapath CPUs

You can configure the SBC to utilize hyperthreading (SMT) support for datapath cores, including forwarding, DoS and transcoding cores. This configuration allows datapath CPUs to utilize two virtual CPUs (vCPUs) as "siblings" on the same physical CPU when the platform host supports hyperthreading. Refer to your software version's Release Notes to determine platforms that support this feature.

If you do not apply this configuration, the SBC only uses one of the two logical virtual CPUs for the datapath (DPDK) cores, and marks the virtual CPU's sibling as stale. Most platforms have their own methods of determining whether hyperthreading is available; some platforms have the support, but do not expose it to the vSBC. Use of hyperthreading by vSBC datapath cores is dependent on both the availability and the visibility of the technology. A quick check to see if you are not utilizing SMT, because the host does not support SMT, does not make its CPU topology visible, or you keep the feature disabled, is that the SBC displays all core assignments in upper case.

If hyper-threading is supported and exposed by the host to the guest configuration, 8 physical cores allocated to the SBC translates to 16 logical cores. Signaling cores automatically set up as siblings. You enable this support for datapath cores using the **use-sibling-core-datapath** parameter in the **system-config** element.

Consider a use case where **use-sibling-core-datapath** remains disabled and you configure 1 forwarding core and 1 DOS core. If the host system allocates 8 vCPUs to this SBC system, the system assigns 2 cores for DPDK (or datapath) and the remaining 6 cores for signaling. This

would consume 8 physical CPUs, displayed by the **show datapath-config** command as (S-S-S-S-S-S-F-D).

By enabling hyper-threading feature in the host system's BIOS and hypervisor and the SBC, the updated core map allocation with the same datapath core configuration (1F, 1D) for 8 vCPUs in the current implementation would consume only 4 physical CPUs, displayed by the **show datapath-config** command as (S-s-S-s-F-n-D-n).

> **Note:**
>
> The Xen hypervisor displays lower-case vCPU siblings together at the end of the vCPU list. Xen pairs the first upper-case vCPU with the first lower-case vCPU, and so forth.

If you have configured CPU pinning in the hypervisor on the host, the SBC enjoys a persistent, one-to-one mapping between physical cores on the host and the vCPUs for the SBC. This can improve performance beyond what is achieved with hyperthreading alone.

> **Note:**
>
> When performing CPU pinning, neither you nor the hypervisor need to allocate cores on the Host in numerical sequence. The SBC does not require that cores be sequentially numbered.

Datapath Hyperthreading Configuration

The **use-sibling-core-datapath** parameter, within the **system-config**, supports two values, and requires that hyperthreading be both enabled and visible from the host:

- disabled (Default)— The system allocates one vCPU sibling, and marks the other as stale.
- enabled—The system allocates all vCPUs siblings.

> **Note:**
>
> When enabled, Oracle recommends you configure an even number of datapath cores for optimal performance.

Platform hosts provide more resources to a physical core on a vSBC than a hyper-threaded core. If your deployment performs Rx and Tx processing on a single core, you should consider leaving hyper-threading disabled.

The **verify-config** command notifies you about invalid configuration, including:

- You enabled the **use-sibling-core-datapath** parameter, but the CPU topology is not exposed to the vSBC. If hyper-threading is not exposed to the vSBC or enabled on the host, the **use-sibling-core-datapath** parameter is not applicable and has no impact on core allocation.
- When the number of signaling cores is less than its minimum (2)
- There is an error with CPU assignment, including improperly configured hyper-threaded sibling CPUs.

Applicable ACLI Command Output

After core configuration, you can use the **show datapath-config** and the **show platform cpu** commands to display CPU core configuration.

You can verify and troubleshoot the SBC CPU assignments using, for example, the **show datapath-config** command.

```
ORACLE# show datapath-config
     Number of cores assigned: 8
     Current core assignments: S-s-S-s-F-n-D-n
        Default hugepage size: 2 MB
     Number of 1 GB hugepages: 0
     Number of 2 MB hugepages: 980
          Total system memory: 7835 MB
 Memory reserved for datapath: 1960 MB
```

You can also use the **show platform cpu** command to see if your host provides SMT awareness.

> **Note:**
>
> This is only true for the SBC products.

```
ORACLE# show platform cpu
CPU count : 8
CPU speed : 2294 MHz
CPU model : Intel Core Processor ...

SMT Topology aware : True
```

Note also the CPU count output, which verifies your configuration.

Datapath CPU Hyperthreading Considerations

You need to reboot your SBC whenever you enable of disable the **use-sibling-core-datapath** parameter. You should consider whether or not to enable the feature as a means of tuning system performance. The feature does not impact or conflict with any other configuration.

You use the following criteria to decide whether or not to enable sibling datapath CPUs:

- Predictability and maintainability
- Effective utilization of all cores
- Throughput

Even if the topology is supported and known to the vSBC, you may not want to enable the new attribute. Consider the fact that the maximum number of forwarding cores is dependent on the maximum number of queue pairs supported on the network interface. An SRIOV interface with the Intel i40e driver, for example, has a limit of 4 Rx/Tx queue pairs, which means the most forwarding cores you can assign to the vSBC is 4. The result of your hyperthreading configurations are:

- Disabled: F-n-F-n-F-n-F-n—System throughput is better when disabled because the host is using 4 full physical cores. If your intent is to achieve highest concurrent session capacity, and there is no constraint on the number of cores available to the vSBC, Oracle recommends you keep this feature disabled.
- Enabled: F-f-F-f—When enabled, it only uses 2. Oracle recommends enabling the feature if your intent is to consume fewer physical cores.

Note the following guidelines when configuring hyperthreading:

- Assign the vCPUs siblings of a single physical core on the host to the same guest machine. Do not mix virtual machines on vCPU siblings.
- If hyper-threading is not supported by your platform, enabling or disabling this parameter has no impact on core allocation.
- If you enable this parameter on supported platforms, Oracle recommends that you configure the number of datapath cores for your vSBC to be divisible by 2 for optimal performance.
- If you enable Hyperthreading on an existing VM you must double-boot the SBC to accommodate the increase in the number of Signaling cores.

Supported Platforms

Of the supported hypervisors, only VMware does not expose SMT capability to the SBC. Of the supported clouds, OCI and AWS enable SMT by default and expose it to the SBC. Azure shapes that enable and expose SMT vary. Please see the Release Notes for your software version to determine which Azure Shapes apply,

System Shutdown

Use the system's **halt** command to gracefully shutdown the VNF.

```
ACMEPACKET# halt

-----------------------------------------------------
WARNING: you are about to halt this SD!
-----------------------------------------------------

Halt this SD [y/n]?:
```

See the *ACLI Reference Guide* for further information about this command.

Configuration Overview

Oracle Communications Session Border Controller Virtual Machine (VM) deployments require configuration of the VM environment and, separately, configuration of the SBC itself. VM-specific configuration on the SBC includes boot parameter configuration, enabling functionality and performance tuning.

During VM installation, you can configure vSBC boot parameters, including:

- IP address
- Host name

During VM installation, the SBC sets default functionality, assigning cores to signaling and media forwarding. If you need DoS and/or transcoding functionality, you configure the

applicable cores after installation. Applicable performance tuning configuration after deployment includes:

- Media manager traffic/bandwidth utilization tuning
- Datapath-related CPU core allocation

> **Note:**
>
> For Xen-based hypervisors, the default boot mode uses DHCP to obtain an IP address for the first management interface (wancom0) unless a static IP is provisioned. Note that DHCP on wancom0 does not support lease expiry, so the hypervisor must provide persistent IP address mapping. If persistent IP address mapping is not provided, the user must manually restart the VM whenever the wancom0 IP address changes due to a manual change or DHCP lease expiry.

Beyond installation, VM-related functional support, and VM-related tuning, you perform basic SBC configuration procedures after installation, including:

- Setting passwords
- Setup product
- Setup entitlements
- Assign Cores
- Enable hyperthreading for forwarding, DoS and transcoding cores
- Service configuration

Configure Cores

The (SBC) allows you to specify the function of the CPU cores available from the host.

Follow the steps below to set up your vSBC cores.

1. Select the **system-config**, as follows.

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

2. Change existing core assignment settings using the **forwarding-cores**, **dos-cores** and **transcoding-cores** parameters in the preceding list. For example, to reserve a core for DoS processing:

```
ORACLE#(system-config) dos-cores 1
```

3. Type **done**, then exit, save and activate your configuration.
4. Reboot your SBC.

Configure Hyperthreading Support

The (SBC) allows you to enable the use of host hyperthreading. This parameter is applicable only when hyper-threading is supported by the platform. You can refer to the platform support list in your software version's Release Notes to identify platform applicability.

If hyper-threading is not supported by your platform, enabling or disabling this parameter has no impact on core allocation. If you enable this parameter on supported platforms, Oracle recommends that you configure the number of datapath cores for your vSBC to be divisible by 2 for optimal performance.

Follow the steps below to enable the **use-sibling-core-datapath** functionality.

1. In Superuser mode, use the following command sequence to access the **system-config**:

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

2. Type **use-sibling-core-datapath** followed by the **enabled** value.

```
ORACLE(system-config)# use-sibling-core-datapath enabled
```

3. Type **done**, then exit, save and activate your configuration.
4. Reboot your SBC.

Session Router Session Capacity Enhancement

You can increase the Session Router's session capacity on X8-2 servers with the sip-threads option by adding it to the **system-config**. The number of sipd threads is calculated based on number of CPUs available and hence you can spawn more sip-threads with this option.

- Default: Based on the number of CPUs
- Values: Min: 1 / Max: 30

Enabling the sip-threads option

To enable this option:

```
ORACLE(system-config)# options +sip-threads=value
```

If you type the option without the plus sign, you overwrite any previously configured options. To append the new option to the options list, prepend the new option with a plus sign as shown in the previous example.

> **Note:**
>
> Contact Oracle Support for assistance with setting the sip-threads option.

General System Information

This section explains the parameters that encompass the general system information on a Oracle Communications Session Border Controller.

System Identification

Global system identification is used primarily by the Oracle Communications Session Border Controller to identify itself to other systems and for general identification purposes.

Connection Timeouts

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

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

After the SSH timeout passes, the SSH session is disconnected. You must use your SSH program to log in to the Oracle Communications Session Border Controller once again to perform any further configuration activity.

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

Cluster Member Graceful Shutdown

When it becomes necessary to temporarily remove an SBC from active service, and make it available only for administrative purposes, the user issues a **set-system-state offline** ACLI command. The SBC begins a graceful shutdown. The shutdown is graceful in that active calls and registrations are not affected, but new calls and registrations are rejected except as discussed below. When the user issues the command, the SBC goes into **becoming offline** mode. Once there are no active SIP sessions and no active SIP registrations in the system, the SBC transitions to **offline** mode. If the SBC is a member of a cluster, the offline status is communicated when the user issues the **set-system-state offline** command, and the SBC excludes the offline SBC in future endpoint (re)balancing algorithms.

The graceful shutdown procedure is limited only to SIP calls and registrations.

Detailed Description of Graceful Shutdowns with Active SIP Calls or Registrations

This is the procedure when active SIP calls or registrations are on an SBC.

When the system receives the **set-system-state offline** command, it transitions to **becoming offline** mode. It begins checking the number of SIP-INVITE-based sessions and the number of SIP registrations, and continues to check them when sessions complete or registrations expire while it is in **becoming offline** mode. When both counts reach zero, the system transitions to **offline mode**. If the system is a member of a Subscriber-Aware Load Balancer cluster, the

Subscriber-Aware Load Balancer client on the SBC changes its cluster status to the **shutdown** state, and informs the Subscriber-Aware Load Balancer that it is **offline**. The Subscriber-Aware Load Balancer ceases to forward new end-points to the SBC and lists the SBC in a **shutdown** state on the Subscriber-Aware Load Balancer. The SBC continues to send heartbeat updates to the Subscriber-Aware Load Balancer as before.

Active calls continue normally when the SBC is in **becoming offline** mode. If SIP refresh registrations arrive for endpoints that have active calls, they are accepted. However, the expiry of these endpoints is reduced to the configurable **retry-after-upon-offline** timer value (in seconds) defined under **sip-config** on the SBC. This timer should be configured to be a much lower time interval than originally requested by the refresh registrations, so that endpoints refresh sooner and thus the registrations expire as closely as possible to when the active call ends. If the new timer value configured in **retry-after-upon-offline** is greater than the existing registration requested refresh value, or if its value is ‘0’ (unconfigured), the original registration refresh request is honored.

Refresh registrations for endpoints that do not have any active calls are rejected with a configurable response code defined in the **sip-config reg-reject-response-upon-offline** parameter. The default for this parameter is the **503 Service Unavailable** message. It includes a **Retry-After** header with a configurable timer set in **retry-after-upon-offline**. If the value of the configuration is 0 (unconfigured), the header is not included in the rejection message. Once these refreshes are rejected, SBC immediately removes such endpoints from its registration cache. It is a force remove. De-registrations are forwarded to the core. There is no local response. Removals are communicated to the Subscriber-Aware Load Balancer.

Any new calls that arrive for endpoints that currently have registration entries are not rejected. The same **retry-after-upon-offline** action is performed.

Any other SIP methods (like SUBSCRIBE or MESSAGE) intended for this endpoint is handled normally and are not rejected. Priority calls are processed as usual by the SBC, regardless of whether an active registration is present in the SBC as long as the SBC is in **becoming offline** state. When the SBC transitions to the **offline** state, even priority calls are rejected. If the priority calls cannot be forwarded to the endpoint, a **380 Alternative Service** response may be sent, depending on the SBC's configuration. However, when the SBC achieves offline mode, even priority calls are rejected. New non-priority calls coming for endpoints that are not currently registered are rejected with the **503 Service Unavailable** error message, as has always been done.

The SBC sends the endpoint removal requests to the Subscriber-Aware Load Balancer so that the Subscriber-Aware Load Balancer removes them from its endpoint table. If a REGISTER message comes in with multiple contacts, it’s possible that one of the contacts has an active call while others do not. In that scenario, the contact without active call has the Expires value in the Contact header changed to 0 and is forwarded to the core. When the response arrives from the core, the Contact with active call has its Expires parameter modified to the **retry-after-upon-offline** value or the UA expires value, whichever is lower. Any contact with no active calls is removed from the cache.

Eventually, all SIP calls end, and all registrations expire. The SBC transitions to the **offline** system state. The SBC continues to send heartbeat updates to the Subscriber-Aware Load Balancer.

At any time after the issuance of the **set-system-state offline** command, a **set-system-state online** command may be issued. If the SBC is in **becoming offline** mode, the process is aborted and the SBC again becomes **online**. The SBC state is forwarded to the Subscriber-Aware Load Balancer, and the SBC once again participates in the Subscriber-Aware Load Balancer's (re)balancing process.

High-level Procedure for Graceful Shutdown

This section describes the graceful shutdown procedure. Details and exceptions to this procedure when there are active calls or registrations are discussed in later paragraphs. The first six actions are performed regardless of whether or not the SBC is part of an Oracle Communications Subscriber-Aware Load Balancer Cluster

- The SBC receives the **set-system-state offline** command.
- The SBC transitions to **becoming offline** mode.
- The SBC accepts calls and subscribes from registered endpoints.
- The SBC rejects calls from non-registered endpoints.
- The SBC rejects new registrations with a **503 Service Unavailable** error message.
- The SBC checks the number SIP INVITE based sessions and number of SIP registrations. When both counts are 0, the SBC transitions to the **offline** state.

> **Note:**
>
> Previous versions only looked at active SIP sessions (calls), without monitoring active SIP registrations.

If the SBC is part of an Subscriber-Aware Load Balancer cluster:

- The Subscriber-Aware Load Balancer client on the SBC changes its cluster status to **shutdown** state.
- The SBC informs the Subscriber-Aware Load Balancer that it is offline.
- The Subscriber-Aware Load Balancer ceases to forward new end-points to the SBC and puts the SBC in a shutdown state.
- Subscriber-Aware Load Balancer continues to forward all messages for existing registered endpoints to the offline SBC.
- The SBC continues to send heartbeat updates the Subscriber-Aware Load Balancer as before.

Configuring General System Information

This section explains how to configure the general system parameters, timeouts, and the default gateway necessary to configure your Oracle Communications Session Border Controller.

To configure general system information:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

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

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

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

System Identification

You must specify identification parameters for this Oracle Communications Session Border Controller.

Set the following parameters to configure the system identification:

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

Configuring Connection and Debug Logging Timeouts

Configure the timeouts for terminal sessions on this Oracle Communications Session Border Controller. These parameters are optional.

Set the following parameters to configure the connection timeouts:

1. **telnet-timeout**—Deprecated. Any value set here is ignored.
2. **console-timeout**—Set the console timeout to the number of seconds you want the Oracle Communications Session Border Controller to wait before it ends the console session. The default value is **0**. The valid range is:
   - Minimum—0

- Maximum—65535

3. **debug-timeout**—Set the time in seconds you want to use for the debug timeout. This is the time allowed before the Oracle Communications Session Border Controller times out log levels for system processes set to debug using the ACLI **notify** and **debug** commands.

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

   The valid range is:

   - Minimum—0
   - Maximum—65535

Phy-Interfaces

Physical interfaces are device ports with which the user connects devices to networks. On the Oracle Communications Session Border Controller (SBC), the user configures the **phy-interface** element, within the **system** branch, for the SBC to use physical interfaces. This section provides an overview of the configuration, and variations of configuration based on platform of the **phy-interface** element.

Physical interface types include:

- Ethernet Management - Non-service interfaces, including:
  - Primary ethernet management - IP-based access to the Command Line Interface (CLI). The interface element is often referred to as wancom0 or eth0.
  - Backup ethernet management - Additional IP-based access to the CLI.
  - High Availability (HA) - Connects the active SBC to a redundant SBC; the redundant SBC immediately resumes signaling and media service if the active fails.
- Media - Interfaces designated for signaling and media service traffic.
- Serial - Direct interface to CLI, which also displays the system's boot sequence and alarm messaging.

The user configures the primary ethernet management and the serial interfaces using boot parameters. This ensures that those interfaces are available even if there is no configuration. The user configures media and backup ethernet management interface via the primary ethernet management interface, often referred to as either eth0 or wancom0, or the serial interface after the system boots.

Interface configuration is platform dependent, with consideration of the following platform types required for successful deployment:

- Acme Packet Platforms
- Virtual Machine Platforms
- Commercial Off the Shelf (COTS) Platforms

Ethernet Management Interfaces

The primary ethernet management interface does not use the **phy-interface** configuration element. The SBC does not display the primary ethernet management interface in the configuration. Instead, the **inet on ethernet** boot parameter sets this interface's IP address. Backup ethernet management and HA interfaces require **phy-interface** configuration.

Platform considerations include:

- Acme Packet platforms:
  - The system uses the **slot** and **port** configuration to identify the physical interface within the **phy-interface** element. Configuration recommendations include setting the **phy-interface**'s **name** parameter to a value that specifies the interface, such as s0p0 (slot 0 port 0).
  - The system defaults to an APIPA (RFC3927) address by default, which the user can change using the boot parameters.
- Virtual Machine platforms:
  - The user must map the primary ethernet management interface and set that interfaces IP address during installation.
  - The Hypervisor allows the user to map all the SBC management interfaces to be used during the install procedure.
- COTS platforms:
  - Primary management interface is platform dependent, using the platform's integrated management application, such as ILOM, to define access to the primary management interface. Users commonly configure a static IP address on the ILOM port, which defaults to DHCP, to simplify access to the SBC's serial port.
  - The **interface-mapping** tools allow the user to manage the mapping between the configured **phy-interface** and the platform's network interface cards on a per-MAC address basis.

Primary and backup ethernet management interfaces access the SBC's CLI by default. Uses can configure any or all ethernet management interfaces to carry other administrative traffic, including:

- SNMP
- SSH
- ACP/XML
- Logs sent from the Oracle Communications Session Border Controller
- Boot the Oracle Communications Session Border Controller from a remote file server

Media Interfaces

All media interfaces require a **phy-interface** element configuration. The **phy-interface** name is always required and is used in subsequent configuration, including **network-interface** and **realm**. Oracle recommends using the naming convention presented in the **interface-mapping** display. Further media **phy-interface** configuration is dependent on platform, including:

- Acme Packet platforms
  - The system uses the **slot** and **port** configuration to identify the physical interface within the **phy-interface** element.
  - Interface mapping management (MACTAB) is irrelevant.
  - The **phy-interface** configuration for special NICs, including the Enhanced Traffic Control and Transcoding Cards, is the same as standard cards.
- Virtual Machine platforms
  - The **interface-mapping** tools allow the user to manage the mapping between the configured **phy-interface** and the platform's network interface cards on a per-MAC address basis.

  - Hypervisor configuration and application performance may vary based on interface architecture. Applicable architecture examples include PCI Passthrough and Paravirtualized.
  - The **phy-interface**'s **name** parameter only specifies the name to be used in subsequent configuration.
  - The **slot**, **port**, **speed**, **duplex** and **autosense phy-interface** parameters are not relevant.
  - The Hypervisor allows the user to map all media interfaces to be used during the install procedure.
- COTS platforms
  - The **interface-mapping** tools allow the user to manage the mapping between the configured **phy-interface** and the platform's network interface cards on a per-MAC address basis.
  - The **phy-interface**'s **name** parameter only specifies the name to be used in subsequent configuration.
  - The **slot**, and **port phy-interface** parameters are not relevant.

Serial Interface

The serial interface provides direct access to the CLI. The user can configure the SBC's serial interface using boot parameters, which configure port output and speed. Platform-dependent detail includes:

- Acme Packet platforms - Serial access is available via one of two physical ports, depending on platform.
- Virtual Machine platforms - Virtual serial interface access is typically provided directly by the hypervisor. Boot parameters are irrelevant.
- COTS platforms - Virtual serial access is available from the integrated management application.

Refer to the High Availability chapter in this document for configuration description and procedures of HA interfaces. Refer to your release-specific *Installation and Platform Preparation Guide* for description and procedures on configuring boot parameters and using the **interface-mapping** tools.

Before You Configure

This section describes steps you should take prior to configuring **phy-interfaces**.

Before you configure a **phy-interface**:

1. Decide on the number and type of **phy-interfaces** you need.

   For example, you might have one media interface connecting to a private network and one connecting to the public network. You might also need to configure maintenance interfaces for HA functionality.
2. Depending on platform, determine the slot and port numbering you need to enter for the **phy-interfaces** you want to configure. Refer to the platform-specific graphics in the *Installation and Platform Preparation Guide* for slot and port numbering reference.
3. If you are configuring your platform for HA, refer to the HA Nodes documentation and follow the instructions there for setting special parameters in the **phy-interface** configuration.

Phy-Interface Configuration

This section describes how to configure **phy-interfaces**.

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

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

2. **name**—Set a name for the interface using any combination of characters entered without spaces. For example: **s0p0**.
3. **admin-state**—Leave the administrative state parameter set to **enabled** to receive and send traffic on this interface. Select **disabled** to prevent media and signaling from being received and sent. The default for this parameter is **enabled**.
4. **operation-type**—Select the type of **phy-interface** connection to use. Refer to the appropriate platform section to identify how this parameter corresponds to an external interface. The default value is **control**. The valid values are:
   - **media**—Use this value for configuring the media interfaces which carry production traffic.
   - **maintenance**—Use this value for configuring the management **phy-interfaces**, used for management protocols or HA.
   - **control**—This legacy parameter may also be used to configure the management **phy-interfaces**.
5. **slot**—Set the slot number for this **phy-interface**. Refer to the appropriate platform section to identify how this parameter corresponds to an external interface.
6. **port**—Set the port number for this **phy-interface**. Refer to the appropriate platform section to identify how this parameter corresponds to an external interface.
7. **auto-negotiation**—Leave this parameter set to **enabled** so that the Oracle Communications Session Border Controller and the device to which it is linked can automatically negotiate the duplex mode and speed for the link.

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

   - enabled | disabled

   Auto negotiation is a requirement for 1Gbit/sec speeds and higher, per the Ethernet Standard.
8. **duplex-mode**—Set the duplex mode. The default is **full**; this field is only used if the auto-negotiation field is set to disabled.

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

   - half | full

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

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

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

Calculation Overview

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

Alarms

In the **phy-interface** configuration, you can establish up to three alarms per media interface—one each for minor, major, and critical alarm severities. These alarms do not have an impact on your system's health score. You set the threshold for an alarm as a percentage used for receiving and transmitting data.

For example, you might configure the following alarms:

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

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

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

Alarm Configuration

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

Configuring Utilization Thresholds for Media Interfaces

To configure utilization thresholds for media interfaces:

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

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

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

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

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

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

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

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

5. **severity**—Enter the severity for the alarm you want to fine for this interface: **minor** (default), **major**, or **critical**. Since the parameter defaults to minor, you must change the value if you want to define a major or critical alarm.
6. **value**—Enter the percentage of utilization (transmitting and receiving) for this interface that you want to trigger the alarm. For example, you might define a minor alarm with a utilization percentage of 50. Valid values are between 0 and 100, where 0 is the default.
7. Save your work.

Configuring Graceful Call Control

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

To enable graceful call control:

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

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

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

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

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

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

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

Network Interfaces

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

IP Configuration

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

VLANs

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

The primary applications of VLANs on the Oracle Communications Session Border Controller are VPNs and peering. Several peering partners may terminate their connections to a Oracle Communications Session Border Controller on a single **phy-interface**. VLAN tags are used to segregate and correctly route the terminated traffic. The Oracle Communications Session Border Controller can support a maximum of 1024 VLANs per **phy-interface**. Ingress packets that do not contain the correct VLAN tag will be dropped. All packets exiting on an egress interface will have the VLAN tag appended to them.

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

Overlapping Networks

Overlapping networks are when two or more private networks with the same addressing schemes terminate on one **phy-interface**. The problem this creates can easily be solved by using VLAN tagging. For example, two 10.x.x.x networks terminating on one Oracle Communications Session Border Controller network interface will obviously not work. The

Oracle Communications Session Border Controller includes the IP Address, Subnet Mask, and 802.1q VLAN tag in its Network Interface determination. This allows Oracle Communications Session Border Controller to directly interface to multiple VPNs with overlapping address space.

Administrative Applications Over Media Interfaces

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

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

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

Configurable MTU Size

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

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

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

1. MTU settings do not apply to media packets.
2. UDP: MTU settings apply only to packets sent by the Oracle Communications Session Border Controller. The Oracle Communications Session Border Controller will continue to process received packets even if they exceed to the configured MTU.
3. Security Phy (IPsec) hardware only; We subtract 100 bytes from the configured MTU to allow for extra headers added by security protocols. This happens even when Security Phy (IPsec) is in clear mode (no security is being applied). Due to hardware limitations of the Security Phy (IPsec) it only allows one MTU per physical port. The maximum MTU of all network interfaces on a given physical port will be used as the MTU for that physical port.
4. The Call Recording feature is where we make a copy of a packet, encapsulate it in an IP-in-IP header and send it to a configured Call Recording Server (CRS). When Call Recording is enabled, to allow space for IP-in-IP encapsulation we reduce the MTU of the original packets to be to be the lesser of the two options listed below.
   - Original Destination network MTU minus size of IP-in-IP header.

- CRS network interface's MTU minus size of IP-in-IP header.

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

Disabling GARP and ND for out-of-subnet Addresses

You can configure the SBC to limit its use of Gratuitous Address Resolution Protocol (GARP) or Network Discovery (ND). Specifically, you can prevent the system from performing this function for each **sip-interface** that is not in the same subnet as the **network-interface** on which they operate. External systems typically reach these addresses through static routes or other routing configurations, making the use of GARP and ND unnecessary for them.

The SBC sends out a GARP or ND message, for IPv4 or IPv6 respectively, for every configured interface and VLAN during every interface initialization, HA switch over, and media link up event. When the number of configured interfaces and VLANs are large, the SBC may send thousands of these GARP or ND in a very short amount of time. Such GARP avalanches can overload the routers and switches connected to the SBC.

The SBC's GARP rate limiting feature helps to avoid overload to some extent. You can further limit the amount of GARP and ND traffic issued by the SBC when you enable the **disable-garp-out-of-subnet** parameter within the **system-config**. When enabled, this feature prevents the system from issuing GARP or ND messages for every **sip-interface** that is not in the same subnet as the **network-interface** over which they operate. Enabling this parameter also prevents the system from issuing GARP or ND messages to loopback interfaces.

Use the syntax below to enable this parameter.

```
ORACLE(system-config)#disable-garp-out-of-subnet enabled
```

This parameter is Real Time Configurable.

> **Note:**
>
> Although the **disable-garp-out-of-subnet** parameter is visible on the Subscriber Aware Load Balancer (SLB), it does not apply and has no effect on that product.

Reporting

You can refer to the log.l2resolver log for information about this feature. This log presents the total number of entries in the NDP table, the number of NDP entries chosen for GARP processing and the number of out-of-subnet interface IPs that the feature has filtered out.

Network Interface Configuration

This section explains how to access and configure network interface.

Special Considerations

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

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

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

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

Network Interfaces Configuration

This section describes how to configure a network interface.

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

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

IP Configuration and Identification

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

Set the following parameters to configure a network interface:

1. **name**—Set the name for the network interface. This must be the same name as the **phy-interface** to which it corresponds.
2. **description**—Enter a description of the network for easier identification.
3. **hostname**—Set the hostname (FQDN) of this network interface. This parameter is optional.
4. **ip-address**—Set the IP address of this network interface.
5. **netmask**—Set the netmask of this network interface in dotted decimal notation.
6. **gateway**—Set the gateway that this network interface uses to communicate with the next hop.

   Oracle recommends, as a best-practice, that you always configure a **gateway** for the media interfaces. If the only trigger to learn a gateway or a directly connected endpoint is ingress media packets, then the SBC typically drops initial packets of a first call until the gateway or the endpoint L2 MAC address is resolved.
7. **sec-gateway**—Set an additional optional gateway for this network interface

8. **dns-ip-primary**—Set the DNS servers. You can set an additional two DNS servers by using the **dns-ip-backup1** and **dns-ip-backup2** parameters.
9. **dns-domain**—Set the default domain name.
10. **signaling-mtu**—Sets the MTU size for IPv4 or IPv6 transmission.

VLAN Configuration

One parameter is required to configure VLANs on a Oracle Communications Session Border Controller. The **sub-port-id** parameter located in the **network-interfaces** element adds and masks for a specific VLAN tag.

- **sub-port-id**—Enter the identification of a specific virtual interface in a **phy-interface** (e.g., a VLAN tab). If this network interface is not channelized, leave this field blank, and the value will correctly default to **0**. The **sub-port-id** is only required if the operation type is Media. The valid range is:
  - Minimum—0
  - Maximum—4095.

HIP Address Configuration

To configure administrative service functionality on a media interface, you must first define all source IP addresses in the media-interface's network that will exchange administrative traffic with the system. Next you will identify the type of administrative traffic each of those addresses will exchange.

You must configure the **gateway** parameter on this **network-interface** for administrative traffic to successfully be forwarded. You should also ensure that this network interface is not on an overlapping network as any of the administrative networks (wancoms).

Set the following parameters to configure HIP functionality on a network interface:

1. **add-hip-ip**—Configure all possible local IP address(es) to which a remote system can send administrative traffic. This parameter specifies addresses that can reply to requests. You must also configure them in a service list, such as **add-icmp-ip**, to specify the service to which they can reply. This parameter can accept multiple IP addresses. You can later remove this entry by typing **remove-hip-ip** followed by the appropriate IP address.
2. **add-ftp-ip**—This parameter has been deprecated.
3. **add-icmp-ip**—Set all possible local IP address(es) to which a remote system can ping the SBC and expect replies. You must also configure these addresses to the **add-hip-ip** parameter. You can later remove this entry by typing **remove-icmp-ip** followed by the appropriate IP address.

   For security, if the ICMP address and the hip-ip-list are not added for an address, the SBC hardware discards ICMP requests or responses for the address.

   > **Note:**
   >
   > IP address changes to the **add-icmp-ip** and **remove-icmp-ip** parameters during traffic hours may impact established calls

4. **add-snmp-ip**—Set the IP address(es) that will access the system's SNMP process. This lets SNMP traffic enter the SBC and reach the host. You can later remove this entry by typing **remove-snmp-ip** followed by the appropriate IP address.

5. **add-telnet-ip**—This parameter has been deprecated.
6. **add-ssh-ip**—Set the IP address(es) that can connect and access the system through SSH. You can later remove this entry by typing **remove-SSH-ip** followed by the appropriate IP address.

Configurable MTU Size

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

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

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

1. MTU settings do not apply to media packets.
2. UDP: MTU settings apply only to packets sent by the Oracle Communications Session Border Controller. The Oracle Communications Session Border Controller will continue to process received packets even if they exceed to the configured MTU.
3. Security Phy (IPsec) hardware only; We subtract 100 bytes from the configured MTU to allow for extra headers added by security protocols. This happens even when Security Phy (IPsec) is in clear mode (no security is being applied). Due to hardware limitations of the Security Phy (IPsec) it only allows one MTU per physical port. The maximum MTU of all network interfaces on a given physical port will be used as the MTU for that physical port.
4. The Call Recording feature is where we make a copy of a packet, encapsulate it in an IP-in-IP header and send it to a configured Call Recording Server (CRS). When Call Recording is enabled, to allow space for IP-in-IP encapsulation we reduce the MTU of the original packets to be to be the lesser of the two options listed below.
   - Original Destination network MTU minus size of IP-in-IP header.
   - CRS network interface's MTU minus size of IP-in-IP header.

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

System Wide MTU Size

To change system wide MTU settings:

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

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

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

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

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

3. **ipv6-signaling-mtu** or **ipv4-signaling-mtu** Configure MTU in the system config and optionally in the network interface. Default will be 1500 bytes.

```
ORACLE(system-config)# ipv6-signaling-mtu 1500
ORACLE(system-config)# ipv4-signaling-mtu 1600
```

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

Scheduled External Backups

You can configure the SBC to automatically back up your important information to an external SFTP server. This feature enhances system reliability by maintaining an off-system copy of this information and by making restoration processes faster.

This feature provides you with the tools to back up:

- The system's configuration file
- Log files
- The output of the **show support-info** command

You can back up these objects using the **schedule-backup** element in the **system-config**. The **schedule-backup** element includes parameters and sub-elements with which you configure backup detail.

Scheduled External Configuration Backup

You can configure the SBC to automatically back up its current backup configuration file dataDoc.gz, which is available at /code/gzConfig/, to an external SFTP server. This feature enhances system reliability by maintaining an off-system copy of your configuration and by making restoration processes faster.

You configure this backup process using the **schedule-backup** element in the **system-config**. The **schedule-backup** element includes parameters and sub-elements with which you configure your **config-backup** detail. This detail specifies how and when to perform the backups as well as **push-receiver** detail, which specifies the server to which you send your backup configuration.

The SBC provides several typical functions for this feature:

- The SBC names the configuration backup files using the format <SBCNODENAME>-YYYYMMDDHHMMSS where:
  - SBCNODENAME—The target name you configure in the SBC bootparam. If you have not configured a target name, the SBC uses the value you configure in the **hostname** parameter within the **system-config**.
    If both the target name in the bootparams and the hostname in the system-config are empty, the system uses the default string value “SBCNONAME” instead of SBCNODENAME.

  - YYYYMMDD—The date when the system created the backup.
  - HHMMSS—The timestamp when the system created the backup, in 24-hour format.
- If you configure more than one **push receiver**, the SBC pushes the configuration file to all of them sequentially.
- The SBC can retry pushing the backup file if it fails.
- The SBC stops attempting to push the backup file when it reaches your configured value for maximum number of retry attempts.
- The SBC generates an alarm and an SNMP trap if it fails to push the backup file to a configured **push receiver**.

To restore the backup configuration on the SBC, you:

- Copy the backup configuration from a target **push receiver** to the /code/bkups folder.
- Run the **restore-backup-config <Backup Filename>** command to restore the configuration.
- Save and Activate the new configuration.

Reporting

The SBC raises an alarm and issues a simultaneous SNMP trap to notify you each time access to a **push-receiver** fails. You can manually clear the alarm using the **clear-alarm** command. There is no associated clear-trap.

The alarm is named **APP_ALARM_SCHBKP_PUSH_FAIL**. The alarm uses the standard format and includes:

- Severity – Warning
- First occurrence timestamp
- Last occurrence timestamp
- Count
- Description – Config backup failed to upload to remote server Hostname: <hostname>, IP: <ipaddress>, Path: <Path>

You can access the SNMP trap using the applicable OID. The name is this trap is **apConfigPushReceiverFailureTrap**. You can access it from the apSip traps using the OID 1.3.6.1.4.1.9148.3.15.7.1.0.1. This trap is reserved for configuration backup push receiver failures only.

In addition, you can review the log.schbkpd file, which is available individually and within the **package-logfiles** package, for operational analysis and troubleshooting this feature.

Scheduled External Log and Support-Info Backup

You can configure the SBC to automatically back up a package of its current log files and the latest support-info command output from the “/opt/archives/logs” folder to an external SFTP server. This feature enhances system reliability by maintaining an off-system copy of your system's state.

You configure this log file and support-info backup using the same procedures as the **config-backup** function. To accommodate this, the **schedule-backup** element in the **system-config** includes the **logs-backup** sub-element.

> **Note:**
>
> Information from the **dump-np-stats** command are excluded from this backup.

The majority of the configuration parameters in **logs-backup** are the same as those in the **config-backup** sub-element. Differences that apply to **logs-backup** include:

- You can configure a **support-info-interval** parameter to specify the number of hours between each stored support-info output. The default value is 3, and the range is from 1 (minimum) to 48 (maximum) hours.
- You can configure a **all-logs-interval** parameter to specify the number of hours after which the system creates a log package. The default value is 12, and the range is from 1 (minimum) to 48 (maximum) hours.
- You can configure a **include-all-logs** parameter to specify that the system includes all logs in each log package in addition to the support-info output. When disabled, the system only backs up the support-info output.
- You can configure a **local-backups-count** parameter to specify the number of packaged **show support-info** files to be maintained for rotation. The default value is 5, and the range is from 1 (minimum) to 24 (maximum) times.
- You can configure a **local-logs-backup-count** parameter to specify the number of packaged “all logs” files to be maintained for rotation. The default value is 5, and the range is from 1 (minimum) to 12 (maximum).

The majority of operational behaviors for this **logs-backup** feature are the same as those in the **config-backup** feature:

- Target file naming function is basically the same. The target file name has to differ, using the syntax <SBCNODENAME>-logs-YYYYMMDDHHMMSS.
- If you configure more than one **push receiver**, the SBC pushes the configuration file to all of them sequentially.
- The SBC can retry pushing the backup file if it fails.
- The SBC stops attempting to push the backup file when it reaches your configured value for maximum number of retry attempts.
- The SBC generates an alarm and an SNMP trap if it fails to push the backup file to a configured **push receiver**.

Alarm and Associated Trap

The SBC raises an alarm and issues a simultaneous SNMP trap to notify you each time access to a **push-receiver** fails. You can manually clear the alarm using the **clear-alarm** command. There is no associated clear-trap.

The alarm is named **APP_ALARM_SCHBKP_LOGS_PUSH_FAIL**. The alarm uses the standard format and includes:

The name of the trap is apPushReceiverFailureTrap. You can access it from the apSip traps using the OID 1.3.6.1.4.1.9148.3.16.2.2.8.0.1. This trap is reserved for configuration backup push receiver failures only.

In addition, you can review the log.schbkpd file, which is available individually and within the **package-logfiles** package, for operational analysis and troubleshooting this feature.

Configure Scheduled Log and Support Info Backups

This task explains how to establish an automatic, timed backup of your Log and Support Info output and send it to a **push-receiver**.
Push receiver configurations establish a connection to the SFTP server to which the SBC pushes records. You can configure the SBC to include copies of your log and **show support-info** output backup within these records.

To configure scheduled log file and **show support-info** output backups:

1. Access the **schedule-backup** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# system-config
ORACLE(system-config)# select
ORACLE(system-config)# schedule-backup
ORACLE(schedule-backup)#
```

2. **logs-backup**—Access the **logs-backup** sub-element.

```
ORACLE (schedule-backup)# logs-backup
ORACLE(logs-backup)#
```

3. **admin-state**—Enable or disable this backup feature. This parameter is disabled by default.
   - **disabled**(default)
   - **enabled**
4. **retry-interval**—Specify the interval the system uses to determine when it tries to resend a configuration backup after an attempt fails. The range is 5 to 10 minutes and the default is 5 minutes.
5. **retry-count**—Specify the maximum number of times the system tries to resend a configuration backup after prior attempts fail. The range is 1 to 3 retries and the default is 3 retries.
6. **push-failure-alarm**—Enable or disable the generation of alarms and traps in case of any push-receiver failures. This alarm is of severity type Warning.
   - **enabled** (default)
   - **disabled**
7. **support-info-interval**—Specify the number of hours between each stored **show support-info** output.
   - **3** (default)
   - **Range is 1 to 48 hours**
8. **all-logs-interval**—Specify the number of hours after which the system creates another log package.
   - **12** (default)
   - **Range is 1 to 48 hours**
9. **include-all-logs**—Enable or disable the system to include all logs in each log package in addition to the support-info output. When disabled, the system only backs up the support-info output.

10. **local-backups-count**—Specify the number of packaged “show support-info” files to be maintained for rotation.
    - **5** (default)
    - **Range is 1 to 24**
11. **local-logs-backup-count**—Specify the number of packaged “all logs” files to be maintained for rotation.
    - **5** (default)
    - **Range is 1 to 12**
12. **local-logs-dir-size**—Specify the desired size (in KB) of the “/opt/archives/logs” folder to be maintained for storage.
    - **1024 MB** (default)
    - **Range is 512 to 2048 MB**
13. **push-receiver**—Access the **push-receiver** sub-element.

    ```
    ORACLE(push-receiver)#
    ```

14. **address**—Enter the IPv4 address of the push receiver to which you want records sent.
    - Default: 0.0.0.0
15. **username**—Enter the username that the SBC uses when it sends records to this server.
16. **password**—Enter the password that the SBC uses when it sends records to this server.

    ```
    ORACLE(push-receiver)# password
    Enter password:
    Retype password:
    Password updated
    ORACLE(push-receiver)#
    ```

17. **data-store**—Enter the absolute path on the remote server where you want the collected data placed.
18. **protocol**—Enter the protocol used to push configuration data to the server.
    - **sftp** (default)

> **Note:**
>
> For SFTP to work, you must import the public key of the SFTP server into the SBC. See the "Manage SSH Keys" in the *ACLI Configuration Guide*.

Configure Scheduled Configuration Backups

This task explains how to establish an automatic, timed backup of your configuration and send it to a **push-receiver**.
Push receiver configurations establish an SFTP server to which the SBC pushes records. You can configure the SBC to include a copy of your backup configuration file as one of these records.

To configure scheduled configuration backups:

1. Access the **schedule-backup** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# system-config
ORACLE(system-config)# select
ORACLE(system-config)# schedule-backup
ORACLE(schedule-backup)#
```

2. **admin-state**—Enable or disable this configuration backup feature. This parameter is disabled by default.
   - **disabled**(default)
   - **enabled**
3. **config-backup**—Access the **config-backup** sub-element.

```
ORACLE(config-backup)#
```

4. **interval**—Specify the interval the system uses to determine when to push backups. This parameter is **weekly** by default.
   - **daily**—24 hours from last backup
   - **weekly**—7 days from last backup
   - **monthly**—30 days from last backup
5. **retry-interval**—Specify the interval the system uses to determine when it tries to resend a configuration backup after an attempt fails. The range is 5 to 30 minutes and the default is 5 minutes.
6. **retry-count**—Specify the maximum number of times the system tries to resend a configuration backup after prior attempts fail. The range is 2 to 10 retries and the default is 5 retries.
7. **push-failure-alarm**—Enable or disable the generation of alarms and traps in case of any push-receiver failures. This alarm is of severity type Warning.
   - **enabled** (default)
   - **disabled**
8. **push-receiver**—Access the **push-receiver** sub-element.

```
ORACLE(push-receiver)#
```

9. **address**—Enter the IPv4 address of the push receiver to which you want records sent.
   - Default: 0.0.0.0
10. **username**—Enter the username that the SBC uses when it sends records to this server.
11. **password**—Enter the password that the SBC uses when it sends records to this server.

```
ORACLE(push-receiver)# password
Enter password:
Retype password:
Password updated
ORACLE(push-receiver)#
```

12. **data-store**—Enter the absolute path on the remote server where you want the collected data placed.
13. **protocol**—Enter the protocol used to push configuration data to the server.
    - **sftp** (default)

> **Note:**
>
> For SFTP to work, you must import the public key of the SFTP server into the SBC. See the "Manage SSH Keys" in the *ACLI Configuration Guide*.

IP Identification (ID) Field

By default, non-fragmented UDP packets generated by media interfaces have the ID field set to 0. You can configure the Oracle Communications Session Border Controller to populate this field with an incrementing value by adding the **increment-ip-id** option in the media manager. Every non-fragmented packet sent will have its ID increased by one from the previous packet sent.

Using a packet trace application, egress packets from the Oracle Communications Session Border Controller will have an ID field that appears to be incrementing. Enabling the ID field can help distinguish a retransmitted non-fragmented application layer packet from a packet retransmitted by the network layer in monitoring or lab situations.

IP Identification Field Configuration

To enable ID field generation in media-manager:

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

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

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. **options**—Set the options parameter by typing **options**, a Space, the option name **increment-ip-id** with a plus sign in front of it, and then press Enter.

```
ORACLE(media-manager)# options + increment-ip-id
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.
4. Save and activate your configuration.

SNMP

The Simple Network Management Protocol (SNMP) is a part of the Internet Protocol Suite, defined by the Internet Engineering Task Force (IETF). It allows you to monitor system and health conditions for an Oracle Communications Session Border Controller (SBC) through an external network management (northbound) system, such as the Oracle Communications Session Delivery Manager or an SNMP manager. The system supports SNMPv3, v2 or v1 to interface with a range of external NMS systems.

Detail on SNMP operation, configuration and data is documented in the *Oracle Communications Session Border Controller MIB Reference Guide*. The first chapter of the Oracle Communications SNMP Reference Guide provides a configuration overview and procedures. The rest of the guide serves as a reference for the MIB.

SNMP Configuration

SNMP configuration on the SBC typically includes defining:

- Administrative management information
- SNMP messaging, including:
  - Trap information—Sent by the SBC to the northbound system, similar to alarms.
  - System detail information—Collected by the northbound system from the SBC. This is typically referred to as read operation.
  - System detail information—Configured by the northbound system on the SBC. This is typically referred to as write operation.

SNMP Data

You must understand SNMP data to determine your actions when you see it. SNMP data is organized into a hierarchical numbering scheme in the form of Object Identifiers (OIDs). OIDs are collected and presented within the context of Management Information Bases (MIBs). A text file external to the SBC system code called a miboid maintains a correlation between object numbers and text names. The system code and miboid numbers correlate each OID to a software or hardware construct that typically has a value and is of interest to the people who monitor them. A set of .mib text files contain the data presented to the human user, referenced by the object names, for each hierarchical information group. You get the applicable files from the device vendor and load them into SNMP managers

OID numbering is, to a large extent, defined and managed by the IETF. This management benefits equipment vendors by preventing information conflation and identifier overlaps. Similar to a MAC address, the IETF provides equipment vendors with numerical identities under which they can create their own hierarchical schemes and define their systems' SNMP information. An example of vendor-specific information is a configuration parameter's value. Similarly, the IETF maintains and shares standard numerical hierarchies used by all equipment vendors so they do not have to create them. An example of standard information is interface speed.

Syslog and Process Logs

Logging events is a critical part of diagnosing misconfigurations and optimizing operations. Oracle Communications Session Border Controllers can send both syslog and process log data to appropriate hosts for storage and analysis.

Overview

The Oracle Communications Session Border Controller generates two types of logs, syslogs and process logs. Syslogs conform to the standard used for logging servers and processes as defined in RFC 3164.

Process logs are Oracle proprietary logs. Process logs are generated on a per-task basis and are used mainly for debugging purposes. Because process logs are more data inclusive than syslogs, their contents usually encompass syslog log data.

Syslog and process log servers are both identified by an IPv4 address and port pair.

Process Log Messages

Process log messages are sent as UDP packets in the following format:

```
<file-name>:<log-message>
```

In this format, <filename> indicates the log filename and <log-message> indicates the full text of the log message as it would appear if it were written to the normal log file.

Syslog and Process Logs Configuration

This section describes how to configure syslog and process log servers.

To configure syslogs and process logs:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can set process log parameters. Skip to the following process log configuration section.

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

   ORACLE(system-config)# **syslog-server**

   ORACLE(syslog-config)#

From this point, you can set syslog parameters. The following is an example what an syslog and process log configuration might look like. Parameters not described in this section are omitted below.

```
system-log-level                WARNING
syslog-server
        address                         172.15.44.12
        port                            514
        facility                        4
process-log-level               NOTICE
process-log-ip-address          0.0.0.0
process-log-port                0
```

Syslog Configuration

The Oracle Communications Session Border Controller supports multiple syslog servers. As the number of active syslog increases, the performance level of the Oracle Communications Session Border Controller may decrease. Therefore, we recommend configuring no more than 8 syslog servers.

Set the following parameters to configure syslog servers:

1. **address**—Set the IPv4 address of a syslog server.
2. **port**—Set the port portion of the syslog server. The default is **514**.
3. **facility**—Set an integer to identify a user-defined facility value sent in every syslog message from the Oracle Communications Session Border Controller to the syslog server. This parameter is used only for identifying the source of this syslog message as coming from the Oracle Communications Session Border Controller. It is not identifying an OS daemon or process. The default value for this parameter is **4**. RFC 3164 specifies valid facility values.

   In software release versions prior to Release 1.2, the Oracle Communications Session Border Controller would send all syslog messages with a facility marker of 4.
4. **system-log-level**—Set which log severity levels write to the system log (filename: acmelog). The default is **WARNING**. Valid values are:
   - EMERGENCY | CRITICAL | MAJOR | MINOR | WARNING | NOTICE | INFO | TRACE | DEBUG | DETAIL

Configure the Process Log Server

Set the following parameters to configure the process log server:

1. **process-log-level**—Set the starting log level all processes running on the system use. Each individual process running on the system has its own process log. The default is **NOTICE**. Valid values: EMERGENCY | CRITICAL | MAJOR | MINOR | WARNING | NOTICE | INFO | TRACE | DEBUG | DETAIL
2. **process-log-ip-address**—Set the IPv4 address of the process log server. The default value is **0.0.0.0**, which causes the system to write log messages to the normal log file.
3. **process-log-port**—Set the port number associated with the process log server. The default value is **0**, which causes the system to write log messages to the normal log file. The valid range is: 1025-65535.

Host Routes

Host routes let you insert entries into the Oracle Communications Session Border Controller's routing table. These routes affect traffic that originates at the Oracle Communications Session Border Controller's host process. Host routes are used primarily for steering management traffic to the correct network.

When traffic is destined for a network that is not explicitly defined on a Oracle Communications Session Border Controller, the default gateway (located in the **system-config**) is used. If you try to route traffic to a specific destination that is not accessible through the default gateway, you need to add a host route. Host routes can be thought of as a default gateway override.

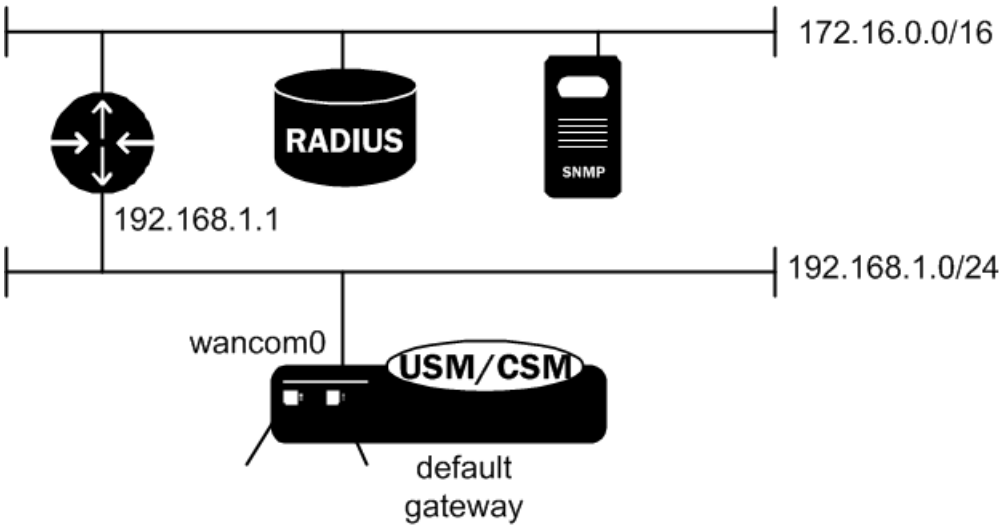
Certain SIP configurations require that the default gateway is located on a media interface. In this scenario, if management applications are located on a network connected to an administrative network, you will need to add a host route for management connectivity.

Host routes to IPv6 addresses do not support a netmask. The system uses the exact match of the dest-network parameter to target an endpoint. The system does not target IPv6 networks with a host route. For IPv4, you need to configure netmask, whereas for IPv6, SBC uses the default value set for netmask.

> **Note:**
>
> Do not configure a **host-route**, **gateway** with an address already used for any existing **network-interface**, **gateway**.

Host Routes Example

When you enable SIP signaling over media interfaces, the default gateway uses an IPv4 address assigned to a media interface. Maintenance services (SNMP and Radius) are located on a network connected to, but separate from, the 192.168.1.0/24 network on wancom0. To route Radius or SNMP traffic to an NMS (labeled as SNMP in the following example), a host route entry must be a part of the Oracle Communications Session Border Controller configuration. The host route tells the host how to reach the 172.16.0.0/16 network. The actual configuration is shown in the example in the next section of this guide.



Host Route Configuration

To configure a host route:

1. Access the **host-route** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# host-route
ORACLE(host-route)#
```

2. **dest-network**—Set the IP address of the destination network that this host route points toward.
3. **netmask**—Set the netmask portion of the destination network for the route you are creating. The netmask is in dotted decimal notation.
4. **gateway**—Set the gateway that traffic destined for the address defined in the first two elements should use as its first hop.
5. Type **done** to save your configuration.

Setting Holidays in Local Policy

This section explains how to configure holidays on the Oracle Communications Session Border Controller.

You can define holidays that the Oracle Communications Session Border Controller recognizes. Holidays are used to identify a class of days on which a local policy is enacted. All configured holidays are referenced in the **local-policy-attributes** configuration subelement as an H in the **days-of-week** parameter. Because holidays are entered on a one-time basis per year, you must configure a new set of holidays yearly.

Holidays Configuration

To configure holidays:

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

```
ORACLE# configure terminal
```

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

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

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

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

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

```
ORACLE(session-router-config)# holidays
ORACLE(session-router-holidays)#
```

From this point, you can configure the holidays subelement. To view all holidays parameters, enter a **?** at the system prompt.

```
holiday
                date                            2005-01-01
                description                     New Years Day
```

To configure a holiday, add an entry for the following parameters in the holidays element:

5. **date**—Enter the holiday's date in YYYY-MM-DD format.
6. **description**—Enter a short description for the holiday you are configuring. If the description contains words separated by spaces, enter the full description surrounded by quotation marks.

Opening TCP Ports 3000 and 3001

This section explains how to open TCP ports 3000 and 3001 primarily for use with an element manager.

- TCP ports 3000 (used when notify commands are issued remotely, i.e. via an element management system) and 3001 (used for remote configuration, i.e. via an element management system), can be enabled or disabled in the system configuration

This configuration is not RTC enabled, so you must reboot your Oracle Communications Session Border Controller for changes to take effect.

Enable System to Connect to SDM

To control TCP ports 3000 and 3001 in the system configuration:

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

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

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

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

3. The parameter controlling ports 3000 and 3001 is called remote-control, and its default is enabled. To disable the ports, set this parameter to disabled.

```
ORACLE(system-config)# remote-control disabled
```

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

5. Reboot your Oracle Communications Session Border Controller. Type a **y** and press Enter to reboot.

```
ORACLE# reboot
----------------------------------------
WARNING: you are about to reboot this SD!
----------------------------------------
Reboot this SD [y/n]?:y
```

DNS on the OCSBC

DNS service is best known for providing resolution of internet domain names to IP addresses. Domain names are easy to remember, but connections require IP addresses. DNS deployments can also provide more comprehensive services, if required. For example, the a DNS client may need the resolution of multiple IP addresses to a single domain name, or the types of service provided by a given server. The Oracle Communications Session Border Controller (SBC) uses DNS predominantly for resolving FQDNs to IP addresses so that it can support sessions.

When configured, the SBC performs DNS client functions per RFC1034 and RFC1035. The user can define one primary DNS server and two backup DNS servers for the SBC to query a domain for NAPTR (service/port), SRV (FQDN), AAAA (IPv6), and A (IP address) information. A common example of the SBC using DNS is to locate a SIP server via server location discovery, as described in RFC 3263. An applicable context is identifying a callee so the SBC can place a call.

There are multiple reasons for the SBC to query a DNS server. In each case, the SBC follows this high level procedure:

1. The system determines the egress realm.
2. The system identifies the egress network interface.
3. From the egress network interface, the system refers to the configured DNS server(s).
4. The system issues the DNS query to the primary server, then any configured backup servers, based on the function and the initial information it has.
5. The system performs recursive lookups or subsequent queries based on, for example, information provided in NAPTR resource responses, until it has one or more resolutions for the FQDN.
6. The system continues processing using the resolved FQDN(s) or indicates it cannot reach that FQDN.

> **Note:**
>
> DNS queries may require host routes.

The SBC also has a DNS Application Layer Gateway (ALG) function that operates independently of its client function. See the DNS ALG Chapter in this document for information about using this ALG.

Closely related to DNS, ENUM service also provides a method of defining a target endpoint, translating E.164 phone numbers to FQDNs. The SBC uses configured ENUM objects for routing calls. ENUM uses Naming Authority Pointers (NAPTR) records defined in RFC 2915 in order to identify available ways or services for contacting a specific node identified through the

E.164 number. See the Session Routing and Load Balancing chapter for information on ENUM services and configuration.

The SBC can cache NAPTR, SRV and A records to speed up DNS and ENUM query processes. The user configures the applicable enum-config to cache these records, providing ENUM and, when configured, DNS with applicable resolutions without having to re-query a server. These resolutions become available to all internal lookup processes that may be generated within the SBC.

DNS Configuration

DNS configuration includes procedures to the **network-interface**, **realm-config**, and **session-agent** elements.

To make DNS operational, configure addressing that is version compatible to the **network-interface** address on the network interface itself.

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

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

2. **dns-ip-primary**—Set the first DNS server with which the interface conducts query procedures.
3. **dns-ip-backup1**—Set the second DNS server with which the interface conducts query procedures should the first server fail.
4. **dns-ip-backup2**—Set the third DNS server with which the interface conducts query procedures should the second server fail.

The system performs DNS query procedures with these servers every time processing encounters an FQDN for which the system needs resolution. After booting up the system, it queries **dns-ip-primary** for resolutions. It queries **dns-ip-backup1** if **dns-ip-primary** fails, and queries **dns-ip-backup2** if **dns-ip-backup1**. The system returns to using **dns-ip-primary** if the second backup fails or you reboot the system.
Review the ensuing sections and configure DNS components to refine DNS operation to your environment, including interface, realm, session agent, and ENUM operation refinement.

Retransmission Logic

The retransmission of DNS queries is controlled by three timers. These timers are derived from the configured DNS timeout value and from underlying logic that the minimum allowed retransmission interval should be 250 milliseconds; and that the Oracle Communications Session Border Controller should retransmit 3 times before timing out to give the server a chance to respond.

- Init-timer is the initial retransmission interval. If a response to a query is not received within this interval, the query is retransmitted. To safeguard from performance degradation, the minimum value allowed for this timer is 250 milliseconds.
- Max-timer is the maximum retransmission interval. The interval is doubled after every retransmission. If the resulting retransmission interval is greater than the value of max-timer, it is set to the max-timer value.

- Expire-timer: is the query expiration timer. If a response is not received for a query and its retransmissions within this interval, the server will be considered non-responsive and the next server in the list will be tried.

The following examples show different timeout values and the corresponding timers derived from them.

```
timeout >= 3 seconds
Init-timer = Timeout/11
Max-Timer = 4 * Init-timer
Expire-Timer = Timeout
timeout = 1 second
Init-Timer = 250 ms
Max-Timer = 250 ms
Expire-Timer = 1 sec
timeout = 2 seconds
Init-Timer = 250 ms
Max-Timer = 650 ms
Expire-Timer = 2sec
```

DNS Support for IPv6

The Oracle Communications Session Border Controller supports the DNS resolution of IPv6 addresses; in other words, it can request the AAAA record type (per RFC 1886) in DNS requests. In addition, the Oracle Communications Session Border Controller can make DNS requests over IPv6 transport so that it can operate in networks that host IPv6 DNS servers.

For mixed IPv4-IPv6 networks, the Oracle Communications Session Border Controller follows these rules:

- If the realm associated with the name resolution is an IPv6 realm, the Oracle Communications Session Border Controller will send the query out using the AAAA record type.
- If the realm associated with the name resolution is an IPv4 realm, the Oracle Communications Session Border Controller will send the query out using the A record type.

In addition, heterogeneous address family configuration is prevented for the **dns-ip-primary**, **dns-ip-backup1**, and **dns-ip-backup2** parameters.

DNS Transaction Timeout

This section explains how to configure the DNS transaction timeout interval on a per network-interface basis. You can currently configure the Oracle Communications Session Border Controller with a primary and two optional backup DNS servers. The Oracle Communications Session Border Controller queries the primary DNS server and upon not receiving a response within the configured number of seconds, queries the backup1 DNS server and if that times out as well, then contacts the backup2 DNS server.

DNS Transaction Timeout Configuration

To configure DNS transaction timeout:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

From this point, you can configure network interface parameters. To view all network interface parameters, enter a **?** at the system prompt.

4. **dns-timeout**—Enter the total time in seconds you want to elapse before a query (and its retransmissions) sent to a DNS server would timeout. The default is **11** seconds. The valid range is:
   - Minimum—1
   - Maximum—999999999.

     If a query sent to the primary DNS server times out, the backup1 DNS server is queried. If the query times out after the same period of time elapses, the query continues on to the backup2 DNS server.
5. Save and activate your configuration.

DNS Entry Maximum TTL

DNS maximum time to live (TTL) is user-configurable and complies with RFCs 1035 and 2181.

One can set the DNS maximum TTL on the Oracle Communications Session Border Controller permitting the DNS entry information to be held until that time is exceeded. One can specifiy the **dns-max-ttl** parameter per network interface and/or to support the DNS ALG feature. The default value is 86400 seconds (24 hours). When the Oracle Communications Session Border Controller configured maximum value has been exceeded, the DNS TTL value is set to the configured maximum and a log entry is written. Otherwise the Oracle Communications Session Border Controller honors the lower value in the DNS response. The Oracle Communications Session Border Controller restricts all DNS entries minimum TTL value of 30 seconds, which the system's implementation of SIP requires.

DNS Entry Max TTL Configuration per Network Interface

Set parameter for DNS entry maximum time to live (TTL) value per network interface.

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

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

2. Select the **network-interface** object to edit.

```
ORACLE(network-interface)# select
<name>:<sub-port-id>:
1: wancom0:0 ip=10.0.0.2 gw=10.0.4.1
```

```
selection: 1
ORACLE(network-interface)#
```

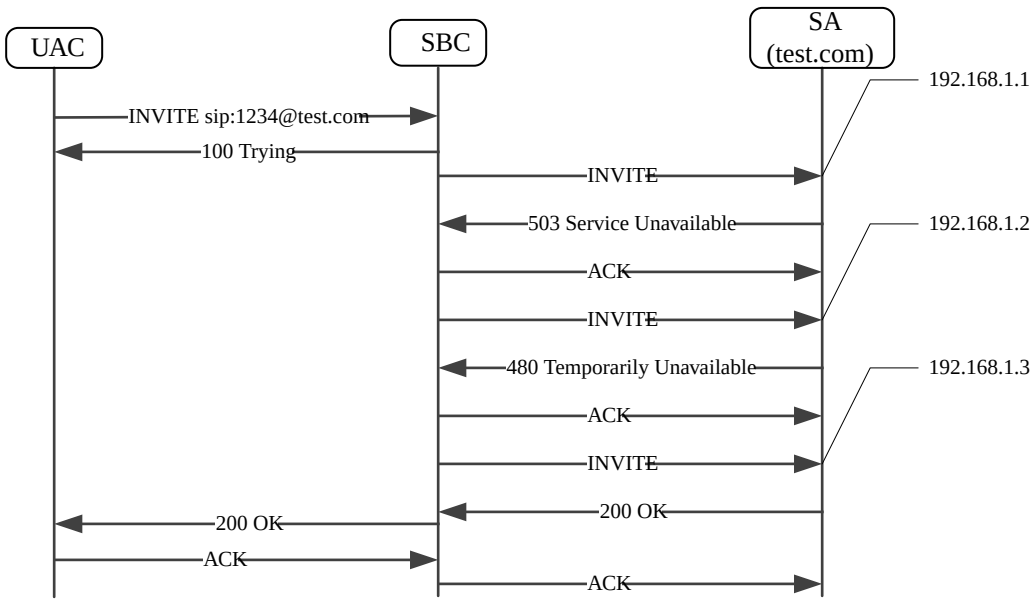
3. **dns-max-ttl**— set to the maximum time for a DNS record to remain in cache.
   - **Minimum: 30**— The lowest value to which the **dns-max-ttl** parameter can be set (in seconds)
   - **Maximum: 2073600**— The maximum value (in seconds) for which the **dns-max-ttl** parameter can be set.
   - **Default: 86400**— The value in seconds which the system uses by default.
4. Type **done** to save your configuration.

DNS-SRV Session Agent Recursion Error Handling

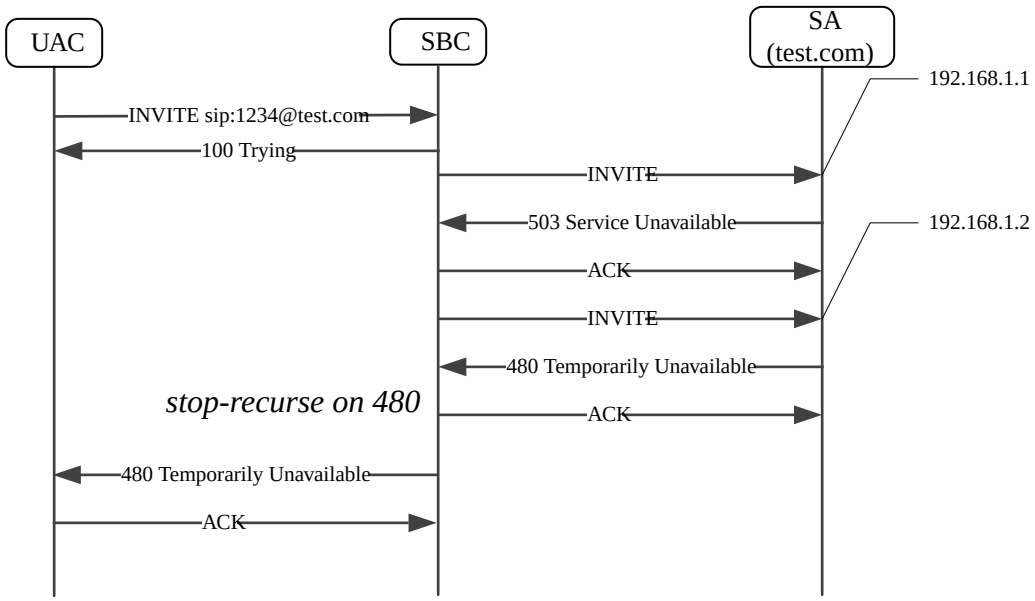
When a session request is sent from the Oracle Communications Session Border Controller to a session agent, and an error response is received (or a transport failure occurs), the Oracle Communications Session Border Controller attempts to reroute the message through the list of dynamically resolved IP addresses. The SBC can be configured to resend session requests through the list of IP addresses under more failure conditions.

This feature concerns the case when a session agent is configured with an FQDN in the hostname parameter and the **dns-load-balance** or **ping-all-addresses** option is configured. This configuration sets up the load balancing / redundancy behavior for the SBC to use all addresses returned in the SRV/A-record for that session agent. In previous versions of the SBC software, only when a 503 failure from the SA was received would the SBC resend the session request to the next dynamically resolved IP address (on the SRV/A record list).

By adding the **recurse-on-all-failures** option to a session agent, the Oracle Communications Session Border Controller will resend a session request to the next address on the list after a 4xx or 5xx failure response has been received from a session agent.



If the SBC receives a failure response from the session agent, and the number of that failure is configured in the **stop-recurse** parameter, no further session requests will be forwarded to additional addresses from the SRV/A record list. The error message will be forwarded back to the UA.



Interface and Realm Support of DNS Servers

You can configure DNS functionality on a per-network-interface, or per-realm basis. Configuring DNS servers for your realms means that you can have multiple DNS servers in connected networks. In addition, this allows you to specify which DNS server to use for a given realm because the DNS might actually be in a different realm with a different network interface.

This feature is available for SIP only.

To configure realm-specific DNS in the ACLI:

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

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

2. **dns-realm**—Enter the name of the network interface that is configured for the DNS service you want to apply in this realm. If you do not configure this parameter, then the realm will use the DNS information configured in its associated network interface.

> **Note:**
>
> Note: If DNS is not configured on the associated **network-interface** of that realm, then following other configured realms shall be used to send out the query. These are listed in order of their preference.
>
> a. **dns-realm** of that realm.
>
> b. **dns-realm** of **egress-realm-id** under **sip-config**, if configured. If not configured, **egress-realm-id** would be used.
>
> c. **dns-realm** of **home-realm-id** under **sip-config**, if configured. If not configured, **home-realm-id** would be used.
>
> This is only applicable for SIP.

DNS Re-query over TCP

The Oracle Communications Session Border Controller DNS supports the truncated (TC) header bit in DNS responses as defined in RFC 2181 and a re-query over TCP.

DNS queries start on UDP ports with the limit of 512 bytes. Longer responses require that the result not be cached and that the truncated (TC) header bit is set. After receiving a DNS response with the TC header set, the Oracle Communications Session Border Controller will initiate a re-query to the DNS server over TCP. The option **dns-tcp-for-truncated-response** in **realm-config** can be set to **no** to disable this behavior.

DNS Re-query over TCP Config

Enable feature to support setting the truncated header bit and initiating a DNS re-query over TCP.

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

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

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **dns-tcp-for-truncated-response**— Set the options parameter by typing **options**, a Space, a plus sign (+), the option name, and queal sign (=) and then **yes** or **no** and then press Enter. The default behavior is to set the truncated header bit and initiate a DNS re-query over TCP.

```
ORACLE(realm-config)# option +dns-tcp-for-truncated-response=no
```

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

Configurable DNS Response Size

When a realm is used for DNS queries, the Oracle Communications Session Border Controller can accept UDP DNS responses configurable up to 65535 bytes.

This functionality is useful when large numbers of SRV records will be returned in a DNS query thereby eliciting a large-sized DNS response. This behavior should be configured on the realm where the DNS servers are located.

To extend the valid DNS response size, add the **dns-max-response-size** option to the realm configuration. If this option is not configured, the Oracle Communications Session Border Controller uses the default maximum response size of 512 bytes, and information past the 512th byte will be ignored.

Do not add the **dns-max-response-size** option to realms where DNS queries are not being performed. Ensure that the realm where this option is configured is referenced in a transport realm's **dns-realm** parameter. Only the local value of the **dns-max-response-size** option is used for the realm; there is no inheritance of this value.

DNS Response Size Configuration

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

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

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. Add the **dns-max-response-size** option to the realm with a value between 513 — 65535.

```
ORACLE(realm-config)#options dns-max-response-size=4196
```

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

Disabling Recursive DNS Queries for ENUM

By default, the Oracle Communications Session Border Controller (SBC) requests DNS query with recursive searches. The Telecommunication Technology Committee's Standard JJ-90.31 specifies that ENUM DNS queries be performed iteratively. The SBC complies with this requirement when remote (server) recursive searches are disabled. You can disable recursive searches on a per **enum-config** basis.

Remote recursive DNS queries instruct DNS servers to query other servers on behalf of the requester to resolve the query. Alternatively, the DNS server can return a list of other DNS servers for the requester to query itself. RFC 1035 specifies that setting the Recursive Data

(RD) bit in the DNS query to 1 requests a remote recursive DNS. Setting RD to 0 requests lists of other servers.

By default, the SBC sets the RD bit to 1. The user can disable this recursion by configuring the **enum-config** element's **remote-recursion** parameter to disabled.

This feature affects queries associated with the **enum-config**:

- Health queries
- CNAM subtype
- ENUM query

Changing this parameter's operational value does not invalidate any current DNS cache entry. The SBC uses the cached information until the cache is aged.

DNS Server Status via SNMP

The Oracle Communications Session Border Controller monitors the status of all configured DNS servers used by a SIP daemon. If a DNS server goes down, a major alarm is sent. If all DNS servers used by a SIP daemon are down, a critical alarm is sent. The apAppsDnsServerStatusChangeTrap is sent for both events.

You can poll the status of a DNS server using the apAppsDNSServerStatusTable in the ap-apps.mib.

Once the apAppsDnsServerStatusChangeTrap has been sent, a 30 second window elapses until the server status is checked again. At the 30 second timer expiration, if the server is still down, another trap and alarm are sent. If the server has been restored to service, the apAppsDnsServerStatusChangeClearTrap is sent.

Persistent Protocol Tracing

This section explains how to configure persistent protocol tracing to capture specific SIP protocol message logs and persistently send them off the Oracle Communications Session Border Controller, even after rebooting the system. This feature is not applicable to log for H.323 or IWF.

About Persistent Protocol Tracing

You can configure sending protocol message logs off of the Oracle Communications Session Border Controller, and have that persist after a reboot. You no longer have to manually issue the notify command each time you reboot.

To support persistent protocol tracing, you configure the following system-config parameters:

- **call-trace**—Enable/disable protocol message tracing (currently only sipmsg.log and alg.log) regardless of the process-log-level setting. If the process-log-level is set to trace or debug, call-trace will not disable.
- **internal-trace**—Enable/disable internal ACP message tracing for all processes, regardless of process-log-level setting. This applies to all *.log (internal ACP message exchange) files other than sipmsg.log and alg.log. If the process-log-level is set to trace or debug, call-trace will not disable.
- **log-filter**—Determine what combination of protocol traces and logs are sent to the log server defined by the process-log-ip parameter value. You can also fork the traces and logs, meaning that you keep trace and log information in local storage as well as sending it

to the server. You can set this parameter to any of the following values: none, traces, traces-fork, logs, logs, all, or all-fork.
The Oracle Communications Session Border Controller uses the value of this parameter in conjunction with the process-log-ip and process-log-port values to determine what information to send. If you have configured the proc-log-ip and proc-log-port parameters, choosing traces sends just the trace information (provided they are turned on), logs sends only process logs (log.*), and all sends everything (which is the default).

> **Note:**
>
> Set the **log-filter** to **all-fork** for the system to include TCP and TLS traces in logs.

About the Logs

When you configure persistent protocol tracing, you affect the following types of logs.

> **Note:**
>
> Enabling logs can have an impact on Oracle Communications Session Border Controller performance.

Process Logs

Events are logged to a process log flow from tasks and are specific to a single process running on the Oracle Communications Session Border Controller. By default they are placed into individual files associated with each process with the following name format:

log.<taskname>

By setting the new log-filter parameter, you can have the logs sent to a remote log server (if configured). If you set log-filter to logs or all, the logs are sent to the log server. Otherwise, the logs are still captured at the level the process-log-level parameter is set to, but the results are stored on the Oracle Communications Session Border Controller's local storage.

Communication Logs

These are the communication logs between processes and system management. The logs are usually named <name>.log, with <name> being the process name. For example, sipd.log.

This class of log is configured by the new internal-trace parameter.

Protocol Trace Logs

The only protocol trace logs included at this time are sipmsg.log for SIP. The H.323 system tracing is not included. All of the logs enabled with the call–trace parameter are sent to remote log servers, if you also set the log-filter parameter to logs or all.

Persistent Protocol Tracing Configuration

Before you configure persistent protocol tracing, ensure you have configured the process logs by setting the system configuration's **process-log-ip** parameter.

To configure persistent protocol tracing:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. **call-trace**—Set to **enabled** to enable protocol message tracing for sipmsg.log for SIP. The default is **disabled**. The valid values are:
   - enabled | disabled
5. **internal-trace**—Set to **enabled** to enable internal ACP message tracing for all processes. The default is **disabled**. The valid values are:
   - enabled | disabled
6. **log-filter**—Choose the appropriate setting for how you want to send and/or store trace information and process logs. The valid values are:
   - **none**—No information will be sent or stored.
   - traces—Sends the trace information to both the log server; includes <name>.log files that contain information about the Oracle Communications Session Border Controller's internal communication processes (<name> is the name of the internal process)
   - **traces-fork**—Sends the trace information to both the log server and also keeps it in local storage; includes <name>.log files that contain information about the Oracle Communications Session Border Controller's internal communication processes (<name> is the name of the internal process)
   - **logs—**Sends the process logs to both the log server; includes log.* files, which are Oracle Communications Session Border Controller process logs
   - **logs-fork**—Sends the process logs to both the log server and also keeps it in local storage; includes log.* files, which are Oracle Communications Session Border Controller process logs
   - **all—**Sends all logs to the log servers that you configure
   - **all-fork**—Sends all logs to the log servers that you configure, and it also keeps the logs in local storage
7. Save and activate your configuration.

System Access Control

You can configure a system access control list (ACL) for your Oracle Communications Session Border Controller that determines what traffic the Oracle Communications Session Border Controller allows over its management interface (wancom0). By specifying who has access to the Oracle Communications Session Border Controller via the management interface, you can provide DoS protection for this interface.

Using a list of IP addresses and subnets that are allowable as packet sources, you can configure what traffic the Oracle Communications Session Border Controller accepts and what it denies. All IP packets arriving on the management interface are subject; if it does not match your configuration for system ACL, then the Oracle Communications Session Border Controller drops it.

> **Note:**
>
> All IP addresses configured in the SNMP community table are automatically permitted.

Adding an ACL for the Management Interface

The new subconfiguration **system-access-list** is now part of the system configuration, and its model is similar to host routes. For each entry, you must define an IP destination address and mask; you can specify either the individual host or a unique subnet.

If you do not configure this list, then there will be no ACL/DoS protection for the Oracle Communications Session Border Controller's management interface.

You access the **system-access-list** via system path, where you set an IP address and netmask. You can configure multiple system ACLs using this configuration.

To add an ACL for the management interface:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type system-access-list and press Enter.

   ```
   ORACLE(system)# system-access-list
   ORACLE(system-access-list)#
   ```

4. **source-address**—Enter the IP address representing for the source network for which you want to allow traffic over the management interface.
5. **netmask**—Enter the netmask portion of the source network for the traffic you want to allow. The netmask is in dotted decimal notation.
6. **description**—Provide a brief description of this system-access-list configuration.
7. **protocol**—Enter a specified protocol or the special value all that specifies by protocol the type of management traffic allowed to access the system. The default value (all) matches all supported transport layer protocols.

   - Default: all
   - Values: all | icmp | ssh | snmp

   An alternate means of configuring values supported by this parameter is the format IP protocol/well-known port. For example, the value 6/22 specifies protocol 6 (TCP) targeting

port 22 (ssh). In addition, you can specify multiple entries using this format. The example (6/22 1/0 17/162) configures multiple entries.

Notes on Deleting System ACLs

If you delete a system ACL from your configuration, the Oracle Communications Session Border Controller checks whether or not there are any active SFTP or SSH client was granted access when the entry was being removed. If such a client were active during ACL removal, the Oracle Communications Session Border Controller would warn you about the condition and ask you to confirm the deletion. If you confirm the deletion, then the Oracle Communications Session Border Controller's session with the active client is suspended.

The following example shows you how the warning message and confirmation appear. For this example, and ACLI has been deleted, and the user is activating the configuration that reflects the change.

```
ORACLE # activate-config
Object deleted will cause service disruption:
 system-access-list: identifier=172.30.0.24
 ** WARNING: Removal of this system-ACL entry will result
             in the lockout of a current SFTP client
Changes could affect service, continue (y/n) y
Activate-Config received, processing.
```

System TCP Keepalive Settings

You can configure the Oracle Communications Session Border Controller to control TCP connections by setting:

- The amount of time the TCP connection is idle before the Oracle Communications Session Border Controller starts sending keepalive messages to the remote peer
- The number of keepalive packets the Oracle Communications Session Border Controller sends before terminating the TCP connection

If TCP keepalive fails, then the Oracle Communications Session Border Controller will drop the call associated with that TCP connection.

In the ALCI, a configured set of network parameters appears as follows:

```
network-parameters
        tcp-keepinit-timer              75
        tcp-keepalive-count             4
        tcp-keepalive-idle-timer        400
        tcp-keepalive-interval-timer    75
        tcp-keepalive-mode              0
```

Then you apply these on a per-interface basis. For example, the H.323 interface (stack) configuration allows you to enable or disabled use of the network parameters settings.

System TCP Keepalive Configuration

TCP setting are global, and then enabled or disabled on a per-interface basis.

To configure TCP keepalive parameters on your SBC:

> **Note:**
>
> If you want to use the default values for TCP keepalive, you can simply set the TCP keepalive function in the H.323 stack configuration, and the defaults for network parameters will be applied.

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

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

2. **tcp-keepinit-timer**—If a TCP connection cannot be established within some amount of time, TCP will time out the connect attempt. It can be used to set the initial timeout period for a given socket, and specifies the number of seconds to wait before the connect attempt is timed out. For passive connections, this value is inherited from the listening socket. The default is **75**. The valid range is:
   - Minimum — 0
   - Maximum — 999999999
3. **tcp-keepalive-count**—Enter the number of packets the SBC sends to the remote peer before it terminates the TCP connection. The default is **8**. The valid range is:
   - Minimum — 0
   - Maximum — 4294967295
4. **tcp-keepalive-idle-timer**—Enter the number of seconds of idle time before TCP keepalive messages are sent to the remote peer if the **SO-KEEPALIVE** option is set. This option is set via the **h323-stack** configuration element. The default is **7200**. The valid range is:
   - Minimum — 30
   - Maximum — 7200
5. **tcp-keepalive-interval-timer**—When the SO_KEEPALIVE option is enabled, TCP probes a connection that has been idle for some amount of time. If the remote system does not respond to a keepalive probe, TCP retransmits the probe after a set amount of time. This parameter specifies the number of seconds to wait before retransmitting a keepalive probe. The default value is **75** seconds. The valid range is:
   - Minimum — 15
   - Maximum — 75
6. **tcp-keepalive-mode**—Set the TCP keepalive response sequence number. The default is **0**. The valid values are:
   - 0—The sequence number is sent un-incremented
   - 1—The number is incremented
   - 2—No packets are sent
   - 3—Send RST (normal TCP operation)

Configurable TCP Timers

You can configure your Oracle Communications Session Border Controller to detect failed TCP connections more quickly so that data can be transmitted via an alternate connection before timers expire. Across all protocols, you can now control the following for TCP:

- Connection establishment
- Data retransmission
- Timer for idle connections

These capabilities all involve configuring an **options** parameter that appears in the network parameters configuration.

Configuring TCP Connection Establishment

To establish connections, TCP uses a three-way handshake during which two peers exchange TCP SYN messages to request and confirm the active open connection. In attempting this connection, one peer retransmits the SYN messages for a defined period of time if it does not receive acknowledgement from the terminating peer. You can configure the amount of time in seconds between the retries as well as how long (in seconds) the peer will keep retransmitting the messages.

You set two new options in the network parameters configuration to specify these amounts of time: **atcp-syn-rxmt-interval** and **atcp-syn-rxmt-maxtime**.

Note that for all configured options, any values entered outside of the valid range are silently ignored during configuration and generate a log when you enter the **activate** command.

To configure TCP connection establishment:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type **network-parameters** and press Enter.

   ```
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **atcp-syn-rxmt-interval=x** (where x is a value in seconds between 2 and 10) with a plus sign in front of it. Then press Enter. This value will be used as the interval between TCP SYN messages when the Oracle Communications Session Border Controller is trying to establish a connection with a remote peer.

   Now enter a second option to set the maximum time for trying to establish a TCP connection. Set the options parameter by typing **options**, a Space, the option name **atcp-**

syn-rxmt-maxtime=x (where x is a value in seconds between 5 and 75) with a plus sign in front of it. Then press Enter.

```
ORACLE(network-parameters)# options +atcp-syn-rxmt-interval=5
ORACLE(network-parameters)# options +atcp-syn-rxmt-maxtime=30
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

> **Note:**
>
> **atcp-syn-rxmt-maxtime=x option is equivalent to the tcp-keepinit-timer parameter, but only affects ATCP.**

5. Save and activate your configuration.

Configuring TCP Data Retransmission

TCP is considered reliable in part because it requires that entities receiving data must acknowledge transmitted segments. If data segments go unacknowledged, then they are retransmitted until they are finally acknowledged or until the maximum number of retries has been reached. You can control both the number of times the Oracle Communications Session Border Controller tries to retransmit unacknowledged segments and the periodic interval (how often) at which retransmissions occur.

You set two new options in the network parameters configuration to specify how many retransmissions are allowed and for how long: **atcp-rxmt-interval** and **atcp-rxmt-count**.

To configure TCP data retransmission:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type **network-parameters** and press Enter.

   ```
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **atcp-rxmt-interval=x** (where x is a value in seconds between 2 and 60) with a plus sign in front of it. Then press Enter. This value will be used as the interval between retransmission of TCP data segments that have not been acknowledged.

   Now enter a second option to set the number of times the Oracle Communications Session Border Controller will retransmit a data segment before it declares the connection failed. Set the options parameter by typing **options**, a Space, the option name **atcp-rxmt-**

count=x (where x is a value between 4 and 12 representing how many retransmissions you want to enable) with a plus sign in front of it. Then press Enter.

```
ORACLE(network-parameters)# options +atcp-rxmt-interval=30
ORACLE(network-parameters)# options +atcp-rxmt-count=6
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Timer for Idle Connections

When enabled to do so, the Oracle Communications Session Border Controller monitors inbound TCP connections for inactivity. These are inbound connections that the remote peer initiated, meaning that the remote peer sent the first SYN message. You can configure a timer that sets the maximum amount of idle time for a connection before the Oracle Communications Session Border Controller consider the connection inactive. Once the timer expires and the connection is deemed inactive, the Oracle Communications Session Border Controller sends a TCP RST message to the remote peer.

To configure the timer for TCP idle connections:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type **network-parameters** and press Enter.

   ```
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **atcp-idle-timer=x** (where x is a value in seconds between 120 and 7200) with a plus sign in front of it. Then press Enter. This value will be used to measure the activity of TCP connections; when the inactivity on a TCP connection reaches this value in seconds, the Oracle Communications Session Border Controllerdeclares it inactive and drops the session.

   ```
   ORACLE(network-parameters)# options +atcp-idle-timer=900
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

RTP TTL

The Oracle Communications Session Border Controller (SBC) allows you to set, on a per media-policy basis, the number of hops RTP packets can traverse before they should be dropped.

This feature uses the standard IPv4 TTL and IPv6 Hop Limit field, comprised of 8 bits in the IP header to specify time to live. The SBC supports this feature over UDP transport. The SBC sets or replaces any existing value in the TTL and Hop Limit fields with your setting before sending packets out the egress interface. The SBC knows if it has already processed any given packet. It therefore, knows to set this value only the first time it processes a packet. In addition, the SBC never decrements this value and, therefore, never discards these packets itself.

To configure, you set **rtp-ttl** in the desired **media-policy**. You also apply the **media-policy** to the desired **realm**(s). The RTP TTL value range is from 0 to 255. By default, the feature is set to zero (Disabled).

```
ORACLE# configuration terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-policy
ORACLE(media-policy)# rtp-ttl 30
```

Display TTL statistics using the **show datapath** command. This command's syntax is platform dependent:

- Virtual Machine, Acme Packet 1100, Acme Packet 3900, Acme Packet 3950, Acme Packet 4900 platforms
  Use **show datapath usdp ppms tos**
- Acme Packet 4600, Acme Packet 6100, Acme Packet 6300 and Acme Packet 6350 platforms
  Use **show datapath etc-stats ppm tos <slot> <port>**

This feature does not work on transcoded packets. After transcoding on Acme Packet platforms, the SBC sets the TTL value in all transcoded packets to 127.

The feature is RTC and is supported for HA deployments.

Bidirectional Forwarding Detection

The Oracle Communications Session Border Controller (SBC) supports Bidirectional Forwarding Detection (BFD) over network interfaces. BFD is a network protocol used to detect faults between two forwarding engines connected by a link. It provides low-overhead detection of faults, even on physical media that doesn't support failure detection of any kind, such as Ethernet, virtual circuits, tunnels and MPLS Label Switched Paths. You configure BFD for functions, including gateway path verification.

BFD is a simple Hello protocol, defined by RFC 5880 and related RFCs, that uses detection mechanisms similar to routing protocols to determine the availability of configured BFD peers. BFD essentially identifies network path failure by transmitting packets periodically between the two peers, and using gaps between the reception of these packets to make the assumption that something in the bidirectional path has failed.

After configuration, each BFD peer identifies itself and sets timing preferences to validate the connection between itself and its peer and establish a BFD session. Peers then negotiate

transmit intervals and 'multipliers' on an ongoing basis to fine tune their monitoring intervals, which are not symmetric. BFD peers use the exchange of BFD control packets to monitor the data path between the peers. BFD uses the "multiplier" to augment the timing intervals, which can account for traffic delays and reduce the impact of false positives. This results in network outage detection and recovery in the range of milliseconds.

The SBC uses BFD to perform two functions:

- Gateway Health Monitoring—The SBC allows the user to configure BFD sessions with applicable gateways. When a session fails, the SBC reduces its health score and raises a network interface alarm. If the session recovers, the SBC resets its health score and clears the alarm. The SBC's HA configuration is independent of this feature.
  You configure primary and secondary sessions on the SBC for this function.
- Triggering Virtual Address Re-routing—The SBC allows the user to configure a BFD session between the virtual address of each media interface and that interface's gateway. The use of BFD extends beyond the layer 2 mechanism of re-assigning a virtual address to a new physical address using, for example, GARP. Using BFD provides for this re-assignment over layer 3 networks by updating the BFD session at the gateway and triggering a dynamic routing update that reconfigures network routing tables.
  You configure Virtual IP (VIP) sessions on the SBC for this function.

Using BFD on the SBC can enable faster HA failover processes, as well as faster health score changes. Failover speed is less noticeable within back-to-back HA deployments, but it can make geographically separated (geo-redundant) HA pair deployments more effective.

> **Note:**
>
> The user may not use BFD in conjunction with the **gw-heartbeat** feature, both of which reside within the **network-interface** element. The SBC displays configuration verification errors if it finds both features configured.

When operating on the SBC, significant BFD detail includes:

- Oracle's implementation of BFD on the SBC implements certain portions of RFCs 5880, 5881, 5882 and 7419.
- The SBC supports BFD's "Asynchronous Mode", within which both endpoints periodically send **hello** packets to each other. Timing mechanisms within the protocol, including minimum receive interval, define a monitoring period within which endpoints must receive traffic. If not, they take the session down. This triggers the use of backup processes.
- The SBC supports only the mandatory components of a BFD control packet, including:
  - Packet header fields, including "Diag", which specifies the session state.
  - Session discriminator (label) used by local host
  - Session discriminator (label) used by remote host
  - Desired minimum TX interval
  - Required minimum RX interval
  - Required minimum Echo RX interval (The SBC does not implement the echo function.)
- The SBC does not support BFD's "Demand Mode".
- The SBC does not support BFD's "Echo function".
- The SBC supports BFD for both IPv4 and IPv6.

- The SBC supports BFD over UDP transport.
- The SBC reports the state of all BFD sessions through all reporting mechanisms.

Gateway Health Checking with BFD

The Oracle Communications Session Border Controller (SBC) allows you to configure Bidirectional Forwarding Detection (BFD) as a means of monitoring and reporting on gateway availability.

This gateway health checking feature monitors the gateway connectivity, and reduces a device's **health-score** when gateway connectivity is lost. For HA, you can configure it on both the active and standby node to enhance operation of the SBC and Subscriber-Aware Load Balancer, especially when deployed in geo-redundant configurations. HA deployments use the changes to health score as a failover trigger; Standalone deployments use the feature to notify you about interface issues.

> **Note:**
>
> Gateway health checking is an alternate means of determining gateway connectivity. Do not use it simultaneously with the **gw-heartbeat** feature.

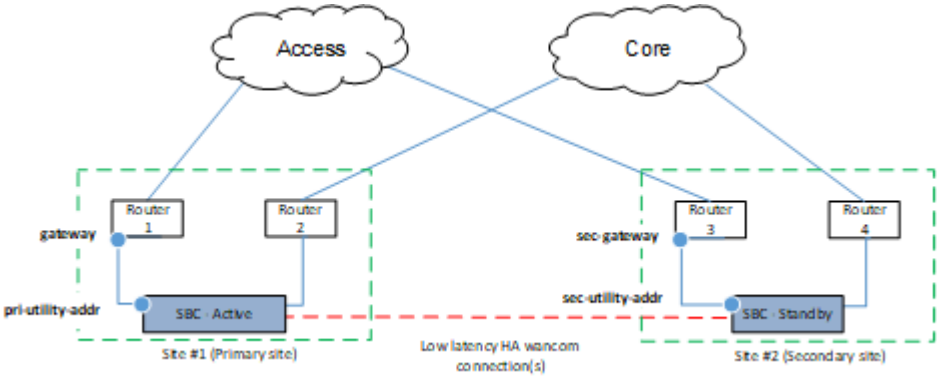
This feature does not apply to:

- Gateways configured on management interfaces (e.g., **wancom0**)
- The default gateway for the system (**system-config**, **default-gateway**)
- Default gateways for host routes (**host-route**, **gateway**)
- Standalone systems—Primary and secondary BFD sessions are not relevant on standalone systems.

For HA deployments, you can configure both **primary** and **secondary** session types, aligning them with the primary and secondary SBCs as follows:

- Configure a **primary** session type for use by the primary node, which uses the **pri-utility-addr** of the **network-interface** as the local IP address.
  These BFD sessions use the **network-interface gateway** as the remote address (i.e., the target of the BFD session).
- Configure a **secondary** session type for use by the secondary node, which uses the **sec-utility-addr** of the **network-interface** as the local IP address.
  These BFD sessions use the **sec-gateway** of the network-interface, if configured, as the remote address (i.e., the target of the BFD session). If no **sec-gateway** address is configured, then a secondary session type uses the **gateway** address for this purpose. This assumes both the primary and secondary nodes are connected to the same gateway.

The diagram below depicts a deployment wherein the primary and secondary node use different gateways.



Session Down Alarm and Trap

The SBC uses the same alarm and SNMP trap to notify you about session status as used for **gw-heartbeat** events. Upon detection of loss of connectivity to a gateway (including at system startup), the SBC raises a **GATEWAY UNREACHABLE** alarm and issues an **apSysMgmtGatewayUnreachableTrap** trap.

Upon detection of the restoration of connectivity to a gateway after loss, the SBC clears the **GATEWAY UNREACHABLE** alarm and issues an **apSysMgmtGatewayUnreachableClear** trap.

In an HA deployment, both the active and standby SBCs can raise this alarm and issue these traps.

BFD Gateway Health Checking Configuration

You configure gateway health checking sessions on the **network-interface**. Configuration includes a bfd-config and subordinate bfd-sessions. You can configure one primary and one secondary session per interface.

```
network-interface
        name                                    M05
...
        ip-address                              172.16.84.12
        pri-utility-addr                        172.16.84.13
        sec-utility-addr                        172.16.84.15
        netmask                                 255.255.0.0
        gateway                                 172.16.84.1
        sec-gateway                             172.16.84.2
        gw-heartbeat
                state                                   disabled
...
        bfd-config
                state                                   enabled
                health-score                            30
                options
                bfd-session
                        bfd-sess-type                           primary
                        admin-state                             enabled
                        admin-session-state                     up
                        min-tx-interval                         5000
                        min-rx-interval                         5000
                        detect-multiplier                       3
                        hold-down-time                          0
```

```
        local-discriminator                 102
bfd-session
        bfd-sess-type                       secondary
        admin-state                         enabled
        admin-session-state                 up
        min-tx-interval                     5000
        min-rx-interval                     5000
        detect-multiplier                   3
        hold-down-time                      0
        local-discriminator                 103
```

> **Note:**
>
> A VIP session type can operate simultaneously with **primary** and **secondary** sessions.

Using BFD To Signal Virtual Address Re-routing

The Oracle Communications Session Border Controller (SBC) allows you to configure a BFD session (or sessions) that monitor virtual address availability. During an HA switchover, this feature provides the routing network with a means of quickly reconstituting routing tables and advertising the new route to the virtual address. You enable this feature in conjunction with the SBC's GARP-based HA mechanism.

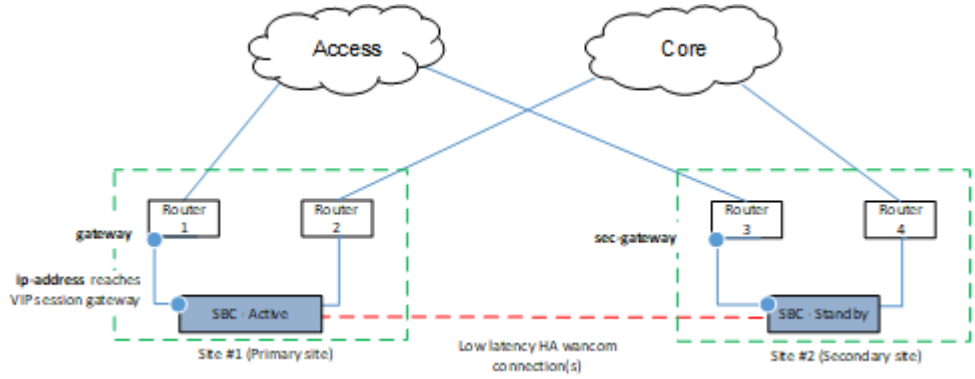
Using BFD, you configure VIP sessions between virtual addresses and the **gateway** configured on each applicable **network-interface**. HA synchronization makes this configuration applicable to the standby's interfaces in case of a fail-over. Upon fail-over, the SBC migrates virtual addresses to the new active. Simultaneously, it starts new VIP sessions between the virtual address and the new gateways on the new active. The layer 3 network, using its own mechanisms, withdraws advertisements to the virtual address over the failed VIP sessions and advertises it via the new VIP sessions.

For example:

1. A gateway device with an active VIP session between itself and an active SBC may advertise the appropriate route to the virtual address, thereby provide connectivity to the active SBC.
2. When the network detects VIP session failure with the active SBC, it may withdraw the route advertisement for the previously active node.
3. When the network detects the new VIP session with the standby SBC, it may issue new advertisements to establish the new route between the new gateway and the virtual address at the new active node.

The local and remote IP addresses for these BFD sessions hosted on the active node include:

- Local IP address: VIP (**address**) configured for the network interface
- Remote IP address, which depends on the active node, as follows:
  - If active is primary node: Configured gateway for the network interface (**gateway**)
  - If active is secondary node: Secondary gateway (**sec-gateway**) if configured, else the configured gateway (**gateway**) for the network interface



Failure of a VIP session has no effect on health score.

VIP Down Alarm

If a VIP session fails, the SBC sends out the following alarm prior to failover.

```
ID        Task  Severity First Occurred        Last Occurred
327724    117   5        2017-12-13 05:31:09   2017-12-13 05:31:09
Count   Description
1       1 VIP BFD session down !!!
```

Standalone systems support VIP sessions. When configured on a standalone, the system establishes a BFD session between the network interface address and its gateway; there are no virtual addresses on a standalone. As a result, you can use this alarm to monitor the interface status. But that is the only benefit to configuring VIP sessions on a standalone.

The SBC does not issue traps on VIP session status.

VIP Session Configuration

You configure VIP sessions on the **network-interface**. Configuration includes a **bfd-config** and a subordinate **bfd-session**. You configure one VIP session per interface. A VIP session can operate simultaneously with a **network-interface**'s gateway health check sessions.

```
network-interface
        name                                    M05
...
        ip-address                              172.16.84.100
        gateway                                 172.16.84.1
        sec-gateway                             182.16.84.2
...
        bfd-config
                state                                   enabled
                health-score                            0
                options
                bfd-session
                        bfd-sess-type                           vip
                        admin-state                             enabled
                        admin-session-state                     up
                        min-tx-interval                         5000
                        min-rx-interval                         5000
                        detect-multiplier                       3
                        hold-down-time                          0
                        local-discriminator                     101
```

Configuring a BFD Config

The Oracle Communications Session Border Controller (SBC) allows you to perform interface- and session-specific configuration for BFD sessions.

Follow the steps below to set up BFD configuration that applies to all sessions on this interface.

1. In Superuser mode, use the following command sequence to access BFD configuration mode:

   ```
   ORACLE# configuration terminal
   ORACLE(configure)# system
   ORACLE(system)# network-interface
   ORACLE(network-interface)# bfd-config
   ```

2. Type state and specify whether this network interface is enabled to use BFD sessions.

   ```
   ORACLE(bfd-config)# state enabled
   ```

3. Type health-score and specify a valid value, between 0 and 100 percent. The default of 0 specifies a deduction of zero, meaning that failed BFD sessions on this interface do not affect health score.

   ```
   ORACLE(bfd-config)# health-score 30
   ```

4. Type bfd-session to enter this subelement and configure individual BFD sessions.

   ```
   ORACLE(bfd-config)# bfd-session
   ```

   If you are adding support for the feature to a pre-existing configuration, then you must select the configuration you want to edit.

   BFD Config configurations accept two independent options:

   - **alarm_on_init**—Requests that BFD session failure alarm be raised at the time of BFD session initialization (when BFD session is enabled). When this option is not specified, the default RFC compliant behavior is to not raise a BFD session failure alarm on initialization to allow failed BFD sessions to re-connect without triggering an unnecessary failover.
   - **exclude_admin_down**—Requests that the BFD session failure alarm be cleared when transitioning to "Admin Down" state from any other state. When this option is not specified, the default behavior is that the BFD session alarm state is not changed when transitioning to "Admin Down".

Configuring BFD Sessions

The Oracle Communications Session Border Controller (SBC) allows you to perform interface- and session-specific configuration for Gateway Health Checking sessions.

Follow the steps below to configure individual sessions. Session types include Primary Gateway Health Checking, Secondary Gateway Health Checking, and VIP sessions.

1. In Superuser mode, use the following command sequence to access **bfd-session** parameters:

```
ORACLE# configuration terminal
ORACLE(configure)# system
ORACLE(system)# network-interface
ORACLE(network-interface)# bfd-config
ORACLE(bfd-config)# bfd-session
```

   If you are adding support for the feature to a pre-existing sub-element, then you must select the configuration you want to edit.

2. Use this parameter to specify this subelement's session type and press Enter. Parameters include primary and secondary, which apply to the availability of the primary and secondary gateways, and VIP, which applies within the context of HA and addresses the connection between the virtual IP address established by a working HA deployment and the applicable gateway.

```
ORACLE(bfd-session)# bfd-sess-type vip
```

3. Use this parameter to specify the admin-state of this BFD session.

```
ORACLE(bfd-session)# admin-state enabled
```

4. Use this parameter to specify the admin-session-state of this BFD session to Up or AdminDown.

```
ORACLE(bfd-session)# admin-session-state up
```

5. Specify the min-tx-interval in milleseconds. Refer to RFC 5880 for more details.

```
ORACLE(bfd-session)# min-tx-interval 5000
```

6. Specify the min-rx-interval in milleseconds. Refer to RFC 5880 for more details.

```
ORACLE(bfd-session)# min-rx-interface 5000
```

7. Specify the detect-multiplier as an integer. Refer to RFC 5880 for more details.

```
ORACLE(bfd-session)# detect-multiplier 3
```

8. Specify the hold-down-time in milleseconds. Zero is disabled. If configured, the system reports the BFD protocol state transition to Up to the application after this duration. After the state transition is eventually reported to the application, the system clears the alarm triggered by the previous BFD session failure and eliminates the corresponding health score deduction (if any).

```
ORACLE(bfd-session)# hold-down-time 0
```

9. Specify the integer used by the system to identify this session. Values range from 1 - 4294967295.

```
ORACLE(bfd-session)# local-discriminator 101
```

10. Type done, then exit, save and activate the configuration.

```
bfd-config
                state                                   enabled
                health-score                            0
                options
                bfd-session
                        bfd-sess-type                           vip
                        admin-state                             enabled
                        admin-session-state                     up
                        min-tx-interval                         5000
                        min-rx-interval                         5000
                        detect-multiplier                       3
                        hold-down-time                          0
                        local-discriminator
101
                bfd-session
                        bfd-sess-type                           primary
                        admin-state                             enabled
                        admin-session-state                     up
                        min-tx-interval                         5000
                        min-rx-interval                         5000
                        detect-multiplier                       3
                        hold-down-time                          0
                        local-discriminator                     102
                bfd-session
                        bfd-sess-type                           secondary
                        admin-state                             enabled
                        admin-session-state                     up
                        min-tx-interval                         5000
                        min-rx-interval                         5000
                        detect-multiplier                       3
                        hold-down-time                          0
                        local-discriminator                     103
```

Displaying Information on BFD Operation

The SBC provides you with commands to display status and statistics on BFD sessions for verification, validation and troubleshooting.

The **show bfd-stats** command displays global status on all active BFD sessions.

```
ORACLE# show bfd-stats
02:52:52-75
--------- -----------   ----------    -----------------  ------
Interface Type    ID    Destination   Logical Interface  State
--------- -----------   ----------    -----------------  ------
M05:0     VIP     101   172.16.84.70  172.16.84.12        Up
M05:0     Primary 102   172.16.84.71  172.16.84.13        Up
--------- -----------   -----------   -----------------
Received packet rate (total): 12 packets/sec
Sent packet rate (total)    : 11 packets/sec
--------- ----------- ----------- -------------
```

The table below provides descriptions of the column information output by the **show bfd-stats** command.

Data	Description
Interface	The Network Interface
Type	Represents the BFD session type, one of primary, secondary or VIP
ID	The configured unique local discriminator of the session
Destination	Destination address of the session
Logical Interface	Local IP address and interface [physical interface:vlan.v4/v6] (similar to network-interface specification in realm-config) of the session
State	BFD session state, one of AdminDown, Down, Init, Up (as specified in RFC 5880)
Received packet rate (total)	The received packet rate for all BFD sessions combined
Sent packet rate (total)	The sent packet rate for all BFD sessions combined

The SBC displays global statistics on BFD traffic from the **show media** command.

```
ORACLE# show media classify 0 0

Slot 0 Port 0 Fastpath Statistics
--------- Ingress Packet Counts ------------|------Egress Packet Counts-----
IPv4                  :  4120               | IPv4                  :
4111
IPv6                  :  0                  | IPv6                  :
5
UDP                   :  4120               | L4                    :
0
TCP                   :  0                  | ARP                   :
490

...

BFD v4 Packets    :  0                  | BFD V4 Packets         :
0
BFD v4 Invalid Packets:  0                  | BFD V4 Invalid Packets :
0
BFD v6 Packets        :  63598              | BFD V6 Packets         :
63547
BFD v6 Invalid Packets:  0                  | BFD V6 Invalid Packets :  0
Media Packets         :  0                  |
MAC Filter Drop       :  2                  | SLB Success            :
0
NAT Miss Drop         :  2                  | SLB L2 Drops           :
0
Standby Drop          :  0                  | SLB L3 Drops           :
0
```

...

BFD-Specific Alarm

The SBC triggers and clears a BFD-specific alarm (example below) when it detects a BFD session state change.

```
ID        Task  Severity First Occurred        Last Occurred
805372204 117   4        2018-02-22 05:36:17   2018-02-22 05:36:17
Count Description
1 gateway 192.168.17.51 unreachable on slot 0 port 1 subport 300
```

For BFD information, set **log-level** to **DEBUG** and capture **log.bfd** for analysis.

- log.bfd: This log file contains process logs and message traces.
- bfd.log: This file contains internal traces between bfd process and other processes that are not call related.

RAMdrive Log Cleaner

The RAMdrive log cleaner allows the Oracle Communications Session Border Controller to remove log files proactively and thereby avoid situations where running low on RAMdrive space is a danger. Because even a small amount of logging can consume a considerable space, you might want to enable the RAMdrive log cleaner.

The RAMdrive cleaner periodically checks the remaining free space in the RAMdrive and, depending on the configured threshold, performs a full check on the /ramdrv/logs directory. During the full check, the RAMdrive cleaner determines the total space logs files are using and deletes log files that exceed the configured maximum lifetime. In addition, if the cleaner finds that the maximum log space has been exceeded or the minimum free space is not sufficient, it deletes older log files until the thresholds are met.

Not all log files, however, are as active as others. This condition affects which log files the log cleaner deletes to create more space in RAMdrive. More active log files rotate through the system more rapidly. So, if the log cleaner were to delete the oldest of these active files, it might not delete less active logs files that could be older than the active ones. The log cleaner thus deletes files that are truly older, be they active or inactive.

Applicable Settings

In the system configuration, you establish a group of settings in the options parameter that control the log cleaner's behavior:

- **ramdrv-log-min-free**—Minimum percent of free space required when rotating log files. When the amount of free space on the RAMdrive falls below this value, the log cleaner deletes the oldest copy of the log file. The log cleaner also uses this setting when performing period cleaning.
- **ramdrv-log-max-usage**—Maximum percent of the RAMdrive the log files can use. The log cleaner removes old log files to maintain this threshold.
- **ramdrv-log-min-check**—Minimum percent of free space on the RAMdrive that triggers the log cleaner to perform a full check of log files.
- **ramdrv-min-log-check**—Minimum time (in seconds) between log cleaner checks.

- **ramdrv-max-log-check**—Maximum time (in seconds) between log cleaner checks. This value must be greater than or equal to the **ramdrv-min-log-check**.
- **ramdrv-log-lifetime**—Maximum lifetime (in days) for log files. You give logs unlimited lifetime by entering a value of 0.

Clean-Up Procedure

The log cleaner checks the amount of space remaining in the RAMdrive and performs a full check of the logs directory when:

- Free space is less than the minimum percent of the RAMdrive that triggers a full check of log files
- The amount of free space has changed by more than 5% of the RAMdrive capacity since the last full check
- A full check of the logs directory has not been performed in the last hour

When it checks the logs directory, the log cleaner inventories the collected log files. It identifies each files as one of these types:

- Process log—Files beginning with log.
- Internal trace file—A <task>.log file
- Protocol trace file—Call trace including sipmsg.log, dns.log, sipddns.log, and alg.log
- CDR file—File beginning with cdr

Next, the log cleaner determines the age of the log files using the number of seconds since the log files were created. Then it orders the files from oldest to newest. The age adjusts such that it always increases as the log file sequence number (a suffix added by file rotation) increases. The log cleaner applies an additional weighting factor to produce a weighted age that favors the preservation of protocol traces files over internal trace files, and internal trace files over process log files. The base log file and CDR files are excluded from the age list and so will not be deleted; the accounting configuration controls CDR file aging.

With the age list constructed, the log cleaner examines the list from highest weighted age to lowest. If the actual file age exceeds the RAMdrive maximum log lifetime, the log cleaner deletes it. Otherwise, the log cleaner deletes files until the maximum percent of RAMdrive that logs can use is no longer exceeded and until the minimum percent of free space required when rotating logs is available.

Clean-Up Frequency

The minimum free space that triggers a full check of log files and the maximum time between log file checks control how often the log cleaner performs the clean-up procedure. When it completes the procedure, the log cleaner determines the time interval until the next required clean-up based on the RAMdrive's state.

If a clean-up results in the deletion of one or more log files or if certain thresholds are exceeded, frequency is based on the minimum time between log cleaner checks. Otherwise, the system gradually increases the interval up to the maximum time between log cleaner checks. The system increases the interval by one-quarter of the difference between the minimum and maximum interval, but not greater than one-half the minimum interval or smaller than 10 seconds. For example, using the default values, the interval would be increased by 30 seconds.

RAMdrive Log Cleaner Configuration

You configure the log cleaner's operating parameters and thresholds in the system configuration. Note that none of these settings is RTC-supported, so you must reboot your Oracle Communications Session Border Controller in order for them to take effect. If you are using this feature on an HA node, however, you can add this feature without impact to service by activating the configuration, rebooting the standby, switching over to make the newly booted standby active, and then rebooting the newly standby system.

Unlike other values for **options** parameters, the Oracle Communications Session Border Controller validates these setting when entered using the ACLI. If any single value is invalid, they all revert to their default values.

To configure the RAMdrive log cleaner:

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

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

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

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

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

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

4. **options**—Set the options parameter by typing options, a Space, **<option name>=X (where X is the value you want to use)** with a plus sign in front of it. Then press Enter.

   Remember that if any of your settings are invalid, the Oracle Communications Session Border Controller changes the entire group of these options back to their default settings.

   The following table lists and describes the supported options.

   - ramdrv-log-min-free—Minimum percent of free space required when rotating log files. When the amount of free space on the RAMdrive falls below this value, the log cleaner deletes the oldest copy of the log file. The log cleaner also uses this setting when performing period cleaning.
     - Default: 40
     - Minimum: 15
     - Maximum: 75
   - ramdrv-log-max-usage—Maximum percent of the RAMdrive the log files can use. The log cleaner removes old log files to maintain this threshold.
     - Default: 40
     - Minimum: 15
     - Maximum: 75
   - ramdrv-log-min-check—Minimum percent of free space on the RAMdrive that triggers the log cleaner to perform a full check of log files.

  - Default: 50
  - Minimum: 25
  - Maximum: 75
- ramdrv-min-log-check—Maximum time (in seconds) between log cleaner checks. This value must be greater than or equal to the ramdrv-min-log-check.
  - Default: 180
  - Minimum: 40
  - Maximum: 1800
- ramdrv-log-lifetime—Maximum lifetime (in days) for log files. You give logs unlimited lifetime by entering a value of 0.
  - Default: 30
  - Minimum: 2
  - Maximum: 9999

  Default=30; Minimum=2; Maximum=9999

```
ORACLE(system-config)# options +ramdrv-log-min-free=50
ORACLE(system-config)# options +ramdrv-log-max-usage=50
ORACLE(system-config)# options +ramdrv-log-min-check=35
ORACLE(system-config)# options +ramdrv-min-log-check=120
ORACLE(system-config)# options +ramdrv-max-log-free=1500
ORACLE(system-config)# options +ramdrv-log-lifetime=7
```

If you type **options** and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Reboot your Oracle Communications Session Border Controller.

System Resource Monitoring

In addition to specific function and operation monitoring, the SBC provides you with the ability to monitor overall system resource utilization to protect against undesirable behavior caused by the cumulative utilization of resources. You configure this feature using the **resource-monitoring-profile** element. This monitoring function operates independently on HA pairs, protecting the standby from resource over utilization independently from the active.

This monitoring feature operates independently of other monitoring functions and takes precedence over those functions and configurations. From a high level, the feature:

- Provides a central service that can monitor multiple critical resources
- Receives resource utilization levels from other tasks and modules
- Takes precautionary actions based on the information to prevent loss of service

This resource monitoring feature segregates system operation into sub-functions so that it can measure those sub-functions' status. This measurement allows the system to determine where resources are being depleted and take action to remedy the problem before they impact system operation. This segregation establishes multiple types of resources that must be measured using different criteria. Ultimately, it calculates percent utilization using that criteria.

This allows it to take actions based on your configured thresholds, which are also percentages. The types of resources monitored include:

- Memory related resources—In these cases, the system measures the amount of memory available to the function against the full capacity of memory.
  - System memory utilization (Heap Memory)
- Session related resources—In these cases, the system measure how many applicable sessions are active against how many sessions are supported. An applicable **resource-type** includes SRTP_SESSIONS.
- PPM related resources—For application-specific packet processing, the SBC establishes Packet Processing Modules (PPMs) to handle that traffic type. The system measures utilization of each resource that uses a corresponding PPM against the maximum number of packets that module can handle before diminishing system performance or behavior.
- CommandQueues—Each SIP message initially enters the command queue from atcpd. Later, the SIP thread processes these messages. If a large number of SIP messages rapidly accumulate in the command queue, it can lead to the queue becoming overloaded. Currently, the queue limit is set to 1500. If more messages arrive beyond this threshold, the queue becomes overloaded.

Operation

The system uses this feature to monitor the resources that you target with configuration, and act when issues arise. After configuration, the system registers the configured resources with this monitoring function. After successful registration, the feature calculates the utilization values of a resource and generates reports to send that data to the Resource Monitoring Module.

Registered resources calculate and send their resource utilization levels to the Resource Monitoring Module in reports every 15 seconds when initiated. The system changes this report timing based on utilization level.

1. If the resource severity crosses Minor threshold value, report timing remains at 15 seconds.
2. If the resource severity crosses Major threshold value, report timing changes to 10 seconds.
3. If the resource severity crosses Critical threshold value, report timing changes to 5 seconds.

Reporting times for command queues are different from the above:

1. If the resource severity crosses Minor threshold value, report timing remains at 5 seconds.
2. If the resource severity crosses Major threshold value, report timing changes to 3 seconds.
3. If the resource severity crosses Critical threshold value, report timing changes to 2 seconds.

> **Note:**
>
> The SIP, MBCD and ATCPD command queues send a deregister request when they go down or generate an error.

When a resource crosses a threshold, it performs the action you configured for that threshold. The system then evaluates each subsequent report to determine if utilization has fallen below that threshold or has triggered a higher threshold.

Configuration

You use the multi-instance **resource-monitoring-profile** element to specify what the system monitors and what actions the system takes. Typically, you create a profile for each resource you want to monitor. To configure this feature, you:

- Enable the **resource-monitoring** parameter in the **system-config** to enable monitoring of all configured resources.
- Set parameters within the **resource-monitoring-profile** especially:
  - The **resource-type**—Specify the resource to monitor. You configure specific resources separately, allowing you to configure behavior per-resource and limit the resources you want to monitor.
  - The **processName**—Applicable to COMMAND_QUEUE resources, this parameter allows you to limit the types of threads you monitor for command_queue issues. If the **resource-type** is anything other than COMMAND_QUEUE, this parameter must be set to **ALL**.
    Note also that you are able to enable new resources for monitoring when you have set **resource-type** to **resource-type** by also specifying the processName. You do this by setting **processName** to the value PROCESS_xxx, where xxx specifies the process. There are many processes that have command queues. Currently SIPD/MBCD/ATCPD are added in command queues monitoring.
  - The threshold configuration sub-elements—Specify the system behavior when resources cross your configured thresholds.
  - The **action** you want the system to take for each condition.

A complete step list for this configuration is provided below.

The system performs the following **verify-config** checks on this feature's parameters when:

- The threshold value of a lower severity (ex: MINOR, MAJOR) is greater than the threshold value of a higher severity (ex: MAJOR, CRITICAL).
- A resource is configured more than once.
- If you configure the **resource-type** to COMMAND_QUEUE and its **action** to **healthscore-decrement-value**, the system throws a **verify-config** error.

Actions

Actions you can configure this module to take include:

- Raise an alarm—There are three alarm severity levels, including OL-1 (MINOR), OL-2 (MAJOR) and OL-3 (CRITICAL), which take precedence over hardcoded system values as well as values you configure within other features.
  The system also issues a trap for the reported severity when it raises an alarm.

> **Note:**
>
> The system triggers a minor alarm if the usage percentage is above the minor threshold but below the major threshold. For example, if the major threshold is set to 60 and the minor threshold is set to 55, the resource monitor raises a minor alarm if the usage percentage is between 55 and 60.

- Decrease the health score, triggering HA switchover when needed—You specify the health score decrement value within each **minor-config**, **major-config** and **critical-config** element individually.

> **Note:**
>
> You cannot use the health score decrement action in conjunction with an monitoring instance that has a **resource-type** set to **CommandQueue**.

Abatement

When resource utilization falls below configure values, the system refers to your configured **abatement** value for that profile and waits for utilization to fall by your abatement setting before reversing an action.

For example, if you configure the threshold to 60, with a minor abatement value of 5, the RM triggers an action if the usage percentage exceeds 60. The system clears the alarm when usage percentage drops to 54.

The system performs the following when utilization reaches the **abatement** value:

- For Raised Alarms, clear the Respective Alarms.
- For Decreasing the system's Health Score, reset that health score to the original value.

> **Note:**
>
> The system decrements health scores via this feature's triggers only when the applicable resource exceeds its critical-threshold 2 consecutive times.

Related Configuration

Specific functional and operational methods that prevent overload within the system, that operate independently of this System Resource Monitoring feature include:

- Dos/DDos protection
- SIP Registration overload protection
- Registration Cache Limit
- CPU/Memory Load Limiting—When you enable this System Resource Monitoring feature to monitor heap, the system disables the alarm threshold function for heap.
- Session Agent Constraints

To accommodate the limitations of these methods, the SBC feeds cumulative information about critical resources to the Resource Monitoring Module.

Reporting on the Resource Monitoring Function

The SBC generates statistics on the monitored resources for debugging purposes on the live systems. The SBC also generates statistics on the Resource Monitoring Function. Mechanism from which you can get this information includes ACLI commands, Alarms, Log Files and SNMP. When deployed in high availability mode, statistics are generated by this function separately on the active and standby nodes.

ACLI Commands

The SBC provides ACLI command You can execute on the runtime system to see what resources are active and what type of action the system has taken for specific thresholds. You can observe these statistics and status reports on the resource monitoring function using the commands below.

- **show resourcemonitor action-information**—Shows the precautionary actions count per resource.

```
TRUNK# show resourcemonitor action-information
Accepted
13:22:17-173
Action Stats
                        TASK-NAME  MINOR  MAJOR  CRITICAL  ALARM&TRAP
DECREMENT_HEALTHSCORE  ClearAction
SRTP_D                  tPpmEntry     0      0         0
0               0                   0
HMU                     tPpmEntry     0      0         0
0               0                   0
2833                    tPpmEntry     0      0         0
0               0                   0
COMMAND_QUEUE              sipd01     0      0         0
0               0                   0
COMMAND_QUEUE              mbcd01     0      0         0
0               0                   0
SRTP_SESSIONS                mbcd     0      0         0
0               0                   0
SRTP_E                  tPpmEntry     0      0         0
0               0                   0
COMMAND_QUEUE            atcpd01     0      0         0
0               0                   0
QOS                     tPpmEntry     0      0         0
0               0                   0
TCP/TLS                     atcpd     0      0         0
0               0                   0
NAT_FLOWS               tPpmEntry     0      0         0
0               0                   0
HEAP                         heap     0      0         0
0               0                   0
TRUNK#
```

- **show resourcemonitor registration-status**—Shows registration status per resource.

```
TRUNK# show resourcemonitor registration-status
Accepted
13:22:43-199
```

```
Registration Status
                              TASK-NAME         STATUS
SRTP_E                        tPpmEntry         INACTIVE
QOS                           tPpmEntry         INACTIVE
2833                          tPpmEntry         INACTIVE
NAT_FLOWS                     tPpmEntry         INACTIVE
COMMAND_QUEUE                 sipd01            INACTIVE
HMU                           tPpmEntry         INACTIVE
COMMAND_QUEUE                 mbcd01            INACTIVE
SRTP_SESSIONS                 mbcd              INACTIVE
COMMAND_QUEUE                 atcpd01           INACTIVE
TCP/TLS                       atcpd             INACTIVE
SRTP_D                        tPpmEntry         INACTIVE
HEAP                          heap              INACTIVE
TRUNK#
```

- **show resourcemonitor resource-registrations**—Shows registration count of resource monitor function resources.

```
TRUNK# show resourcemonitor resource-registrations
Accepted
13:23:05-120
Registration stats                ---- Lifetime ----
                           Recent      Total   PerMax
Register                        0         12        6
Register 4xx                    0         12        6
Register 200OK                  0          0        0
TRUNK#
```

- **show resourcemonitor resource-reports**—Shows the report count for resources in the resource monitor function.

```
TRUNK# show resourcemonitor resource-reports
Accepted
13:23:27-143
Usage-Percent stats                        ---- Lifetime ----
Resource           TASK-NAME          Recent      Total   PerMax
SRTP_D             tPpmEntry               0          0        0
HMU                tPpmEntry               0          0        0
2833               tPpmEntry               0          0        0
COMMAND_QUEUE      sipd01                  0          0        0
COMMAND_QUEUE      mbcd01                  0          0        0
SRTP_SESSIONS      mbcd                    0          0        0
SRTP_E             tPpmEntry               0          0        0
COMMAND_QUEUE      atcpd01                 0          0        0
QOS                tPpmEntry               0          0        0
TCP/TLS            atcpd                   0          0        0
NAT_FLOWS          tPpmEntry               0          0        0
HEAP               heap                    0          0        0
REPORT REJECTION                           0          0        0
TRUNK#
```

- **reset resourcemonitor**—Resets all the statistics produces by the commands above.

The system also provides manual commands to clear out the actions the system has taken for the specified or all resource types.

```
clear-resourcemonitor < "resource-type" > | < all >
```

This command syntax example clears any action performed on heap memory, based on every crossed threshold.

```
ORACLE#clear-resourcemonitor HEAP
```

You can also use the following to view information the system generates within the context of this feature:

- The “log.tResourceMonitor” log file captures log messages generated by the Resource Monitoring Module.
- DEBUG and MINOR level logs in the “log.heap” logfile .
- Whenever required, the SBC issues the applicable SNMP traps and alarms. The applicable objects include:
  - apRmResourcesGroupTrap
  - apRmResourcesGroupClearTrap
  - RESOURCE_MONITOR_ALARM

Logs and Alarms

The SBC includes a log file named log.tPpmEntry that presents PPM resource logs generated by the resource monitor client. There are also some DEBUG and MINOR level log message related to this resource monitoring feature in the “log.sipd”, “log.mbcd” and “log.atcpd” logfile.

There are no separate alarms used only for this feature. Instead, the SBC uses the alarms and traps previously developed for resource issues. Each resource uses its own ALARM ID and processName.

The example alarm below show ID 131100 informing you that the HEAP process, configured with a threshold of 80% is breached.

```
HYDERABAD# display-alarms
2 alarms to show
ID Task Severity First Occurred Last Occurred
131100 3524 4 2023-12-04 06:33:59 2023-12-04 06:34:29
Count Description
2 Resource monitoring module raising alarm - Resource task id = 3524,
threshold = 80%, current value = 82%,processName = heap
```

> **Note:**
>
> Whenever the system raises an alarm within the context of this feature, it simultaneously issues an equivalent trap.

New alarms include:

Resource Monitoring Alarms Table:

AlarmName	Alarm-ID	Description	causes	Action	severity	TrapName
MEM_UTIL_OVER_THRESHOLD	131100	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%, processName = theap , resourceName = HEAP	High memory usage	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap
COMMAND_QUEUE_OVER_THRESHOLD	131426	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,processName = sipd1 , resourceName = COMMAND_QUEUE	High command Queue usage	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap
NAT_OVER_THRESHOLD	131427	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,processName = tPpmEntry, resourceName = NAT_FLOWS	High calls / media sessions	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap

DTMF_2833_OVER_THRESHOLD	131429	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,process Name = tPpmEntry , resourceName = 2833	High DTMF calls usage	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap
HMU_OVER_THRESHOLD	131431	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,process Name = tPpmEntry , resourceName = HMU	High calls with HMU usage	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap
QOS_OVER_THRESHOLD	131430	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,process Name = tPpmEntry, resourceName = QOS	High calls with QOS usage	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap
SRTPE_OVER_THRESHOLD	131432	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,process Name = tPpmEntry , resourceName = SRTP_E	High calls with SRTP_E usage	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap

SRTD_OVER_THRESHOLD	131433	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,process Name = tPpmEntry, resourceName = SRTP_D	High calls with SRTP_D usage	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap
SRTP_SESSIONS_OVER_THRESHOLD	131428	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,process Name = tmbcd, resourceName = SRTP_SESSIONS	High calls of SRTP_SESSIONS	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap
TCP_OVER_THRESHOLD	131434	Resource monitoring module raising alarm - Resource task id = TASK_ID, threshold = 80%, current value = 82%,process Name = tPpmEntry , resourceName = TCP/TLS	High TCP/TLS calls.	Ensure calls are managed in a way that resource usage stays within the specified threshold	minor major critical	apRmResourcesGroupTrap

SNMP

With respect to SNMP, the SBC uses the ap-res-monitor.mib to collate status data on each registered monitoring function, and generate SNMP traps on the configured resource monitoring object. The applicable OIDs use the prefix, 1.3.6.1.4.1.9148.3 under the table apRMResourceObjects, which is within ap-res-monitor.mib.

For traps, the SBC maintains an object named apRMNotifications, which includes the apRMNotificationObjects. This object refers to the apRmTrapType, which in conjunction with apRmTrapValue establishes a usable value pair that specifies each resource's utilization. The

system can compare these values with your configured thresholds and generate the following when an active value crosses above or below your thresholds:

- apRmResourcesGroupTrap (threshold raised trap)
- apRmResourcesGroupClearTrap (threshold came down trap).

See the *MIB Guide* for detailed information about SNMP variables in the Resource Monitor OIDs section and SNMP traps in the apRMResourceObjectsTraps (ap-res-monitor.mib) section.

Configure the Resource Monitoring Function

This task explains how to configure the resource monitoring function on the SBC. This includes one or more **resource-monitoring-profile** configurations to specify the resources that you want to monitor, the actions that you want the system to take to alleviate problems, and the thresholds you set to specify when a problem with a resource occurs.
To configure this feature, you configure each **resource-monitoring-profile** element within the **system** branch, and the **minor-threshold**, **major-threshold**, and **cretical-threshold** parameters as sub-elements of each **resource-monitoring-profile**. Finally, you enable all your profiles by enabling the **resource-monitoring-profile** parameter in the **system-config**:

1. Access the **resource-monitoring-profile** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# resource-monitor-profile
   ORACLE(resource-monitor-profile)#
   ```

2. **resource-type**—Specify the resource to be monitored for this profile. Values include:
   - HEAP
   - COMMAND_QUEUE
   - SRTP_SESSIONS
   - NAT_FLOWS
   - HMU
   - QOS
   - SRTP_E
   - SRTP_D
   - 2833
   - TCP/TLS
3. **state**—Enable or disable this profile. This parameter allows you to enable/disable individual profiles. Values include:
   - **disabled** (default)
   - **enabled**
4. **processName**—Specify the command queue you intend to monitor. For all **resource-type** values except **COMMAND_QUEUE**, you must retain the default of **PROCESS_ALL**. You can choose any of the values below when you set the **resource-type** to **COMMAND_QUEUE**. Set to one of the following:
   - **PROCESS_ALL** (default)

- **PROCESS_MBCD**
- **PROCESS_SIPD**
- **PROCESS_ATCPD**
  You can also specify any process you want to monitor by entering the value PROCESS_xxxx

5. **abatement-threshhold**—Specify the percent utilization decrement that a resource's utilization must fall before the system acts. After utilization falls by this percent below the trigger threshold, the system reverts the action taken when that threshold was crossed. Abatement actions include clearing the respective alarm, re-enabling any non-essential service, and increasing system health score by the value with which it was decreased.
   - **5 percent** (default)
   - **Range is 5 to 50 percent**
6. **minor-config**—Access the **minor-config** sub-element.

   ```
   ORACLE(resource-monitor-profile)# minor-config
   ORACLE(minor-config)#
   ```

7. **minor-threshold**—Specify the value at which the system takes its minor action for this profile.
   - **70** (default)
   - **Range is 50 to 90 percent**
8. **minor-precaution-action**—Set to one of the following:
   - RESOURCE_MONITOR_ALARM—When set, the system sends the RESOURCE_MONITOR_ALARM and associated trap when any **resource-type** crosses any of its major, minor or critical thresholds.
   - DECREMENT_HEALTH_SCORE
9. **healthscore-decrement-value**—This triggers only when this resource exceeds its critical-threshold 2 consecutive times.
   - **30** (default)
   - **Range is 10 to 100 percent**
10. Type **done** to retain your **minor-config** configuration.
11. **major-config**—Access to the **major-config** sub-element.

    ```
    ORACLE(minor-config)# exit
    ORACLE(resource-monitor-profile)# major-config
    ORACLE(major-config)#
    ```

12. Repeat steps 7, 8 and 9 to create your **major-config**. All parameters and values are the same with the exception of the **major-threshold** range and default:
    - **80** (default)
    - **Range is 60 to 90 percent**
13. Type **done** to retain your **major-config** configuration.

14. **critical-config**—Access to the **critical-config** sub-element.

```
ORACLE(major-config)# exit
ORACLE(resource-monitor-profile)# critical-config
ORACLE(critical-config)#
```

15. Repeat steps 7, 8 and 9 to create your **critical-config**. All parameters and values are the same with the exception of the **critical-threshold** range and default:
    - **90** (default)
    - **Range is 70 to 95 percent**
16. Type **done** to retain your **critical-config** configuration.
17. Type **done** to retain your **resource-monitoring-profile** configuration.
18. Repeat the steps above for each resource monitor function you want to configure.
19. Access and enable the **resource-monitoring-profile** parameter in the **system-config**.

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

> **Note:**
>
> The **select** command is required to enter the **system-config** element.

20. Enable the **resource-monitoring-profile** parameter within the **system-config** branch. This parameter allows you to enable or disable all configured profiles simultaneously.

```
ORACLE(system-config)# resource-monitoring-profile enable
```

21. Type **done** to retain your **system-config** configuration.
22. Save and activate your configuration.

Configurable Alarm Thresholds and Traps

The Oracle Communications Session Border Controller supports user-configurable threshold crossing alarms. These configurations let you identify system conditions of varying severity which create corresponding alarms of varying severity. You configure an alarm threshold type which indicates the resource to monitor. The available types are:

- cpu — CPU utilization monitored as a percentage of total CPU capacity
- memory — memory utilization monitored as a percentage of total memory available

> **Note:**
>
> When you configure an **alarm-threshold** for **memory** with **severity** set to **critical**, the Oracle Communications Session Border Controller will stop processing traffic if that configured value is reached, regardless of how low the value is.
> When triggered this alarm issues a corresponding SNMP trap. The system checks utilization every 15 seconds after triggering the alarm and issues a the trap again if the memory utilization is still breaching the threshold. This can generate a significant number of traps sent to SNMP management systems.

- sessions — allowed utilization monitored as a percentage of session capacity
- space — remaining disk space (configured in conjunction with the volume parameter - see the Storage Expansion Module Monitoring section of the *Accounting Guide* for more information.)
- deny-allocation — denied entry utilization monitored as a percentage of reserved, denied entries.

For the alarm type you create, the Oracle Communications Session Border Controller can monitor for 1 through 3 severity levels as minor, major, and critical. Each of the severities is configured with a corresponding value that triggers that severity. For example the configuration for a CPU alarm that is enacted when CPU usage reaches 50%:

```
alarm-threshold
        type                            cpu
        severity                        minor
        value                           50
```

You may create addition CPU alarms for increasing severities. For example:

```
alarm-threshold
        type                            cpu
        severity                        critical
        value                           90
```

The alarm state is enacted when the resource defined with the type parameter exceeds the value parameter. When the resource drops below the value parameter, the alarm is cleared.

SNMP Traps

When a configured alarm threshold is reached, the Oracle Communications Session Border Controller sends an apSysMgmtGroupTrap. This trap contains the resource type and value for the alarm configured in the alarm-threshold configuration element. The trap does not contain information associated with configured severity for that value.

```
apSysMgmtGroupTrap          NOTIFICATION-TYPE
    OBJECTS         { apSysMgmtTrapType, apSysMgmtTrapValue }
    STATUS            current
    DESCRIPTION
          " The trap will generated if value of the monitoring object
```

```
        exceeds a certain threshold. "
    ::= { apSystemManagementNotifications 1 }
```

When the resource usage retreats below a configured threshold, the Oracle Communications Session Border Controller sends an apSysMgmtGroupClearTrap.

```
apSysMgmtGroupClearTrap         NOTIFICATION-TYPE
    OBJECTS        { apSysMgmtTrapType }
    STATUS           current
    DESCRIPTION
          " The trap will generated if value of the monitoring object
          returns to within a certain threshold.  This signifies that
          an alarm caused by that monitoring object has been cleared. "
    ::= { apSystemManagementNotifications 2 }
```

The alarm and corresponding traps available through the User Configurable Alarm Thresholds functionality are summarized in the following table.

Alarm	Severity	Cause	Actions
CPU	minor major	high CPU usage	apSysMgmtGroupTrap sent with apSysCPUUtil
	critical		apSysMgmtTrapValue
memory	minor major	high memory usage	apSysMgmtGroupTrap sent with apSysMemoryUtil
	critical		apSysMgmtTrapValue
sessions	minor major	high provisioned usage	apSysMgmtGroupTrap sent with apSysLicenseCapacity
	critical		apSysMgmtTrapValue
space	minor major	high HDD usage, per volume	apSysMgmtStorageSpaceAvailThresholdTrap sent with:
	critical		apSysMgmtSpaceAvailCurrent
			apSysMgmtSpaceAvailMinorThreshold
			apSysMgmtSpaceAvailMajorThreshold
			apSysMgmtSpaceAvailCriticalThreshold
			apSysMgmtPartitionPath
deny allocation	minor major	high usage of denied ACL entries	apSysMgmtGroupTrap sent with apSysCurrentEndptsDenied
	critical		apSysMgmtTrapValue

Alarm Thresholds Configuration

To configure alarm thresholds:

1. Access the **alarm-threshold** configuration element.

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

2. **type** — Enter the type of resource which this alarm monitors. Valid values include:
   - cpu
   - memory
   - sessions
   - space
   - deny-allocation
3. **volume** — Enter the logical disk volume this alarm monitors (used only in conjunction when type = space).
4. **severity** — Set the severity of the threshold. Valid values include:
   - minor
   - major
   - critical
5. **value** — Enter the value from 1 to 100, indicating the percentage, which when exceeded generates an alarm.
6. Save and activate your configuration.

Alarm Synchronization

Two trap tables in the ap-smgmt.mib record trap information for any condition on the Oracle Communications Session Border Controller that triggers an alarm condition. You can poll these two tables from network management systems, OSS applications, and the Session Delivery Manager to view the fault status on one or more Oracle Communications Session Border Controller s.

The two trap tables that support alarm synchronization, and by polling them you can obtain information about the current fault condition on the Oracle Communications Session Border Controller . These tables are:

- apSysMgmtTrapTable—You can poll this table to obtain a summary of the Oracle Communications Session Border Controller ’s current fault conditions. The table records multiples of the same trap type that have occurred within a second of one another and have different information. Each table entry contains the following:
  - Trap identifier
  - System time (synchronized with an NTP server)
  - sysUpTime
  - Instance number
  - Other trap information for this trap identifier
- apSysMgmtTrapInformationTable—You can poll this table to obtain further details about the traps recorded in the apSysMgmtTrapTable table. The following information appears:
  - Data index
  - Data type
  - Data length
  - The data itself (in octets)

Trap tables do not record information about alarm severity.

The apSysMgmtTrapTable can hold up to 1000 entries, and you can configure the number of days these entries stay in the table for a maximum of seven days. If you set this parameter to 0 days, the feature is disabled. And if you change the setting to 0 days from a greater value, then the Oracle Communications Session Border Controller purges the tables.

Caveats

Note that the Oracle Communications Session Border Controller does not replicate alarm synchronization table data across HA nodes. That is, each Oracle Communications Session Border Controller in an HA node maintains its own tables.

Alarm Synchronization Configuration

You turn on alarm synchronization in the system configuration.

To use alarm synchronization:

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

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

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

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

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

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

4. **trap-event-lifetime**—To enable alarm synchronization—and cause the Oracle Communications Session Border Controller to record trap information in the apSysMgmtTrapTable and the apSysMgmtTrapInformationTable—set this parameter to the number of days you want to keep the information. Leaving this parameter set to 0 (default) turns alarm synchronization off, and you can keep information in the tables for up to 7 days. 7 is the maximum value for this parameter.

Accounting Configuration

The Oracle Communications Session Border Controller offers support for RADIUS, an accounting, authentication, and authorization (AAA) system. In general, RADIUS servers are responsible for receiving user connection requests, authenticating users, and returning all configuration information necessary for the client to deliver service to the user.

You can configure your Oracle Communications Session Border Controller to send call accounting information to one or more RADIUS servers. This information can help you to see usage and QoS metrics, monitor traffic, and even troubleshoot your system.

This guide contains all RADIUS information, as well as information about:

- Accounting for SIP and H.323
- Local CDR storage on the Oracle Communications Session Border Controller , including CSV file format settings

- The ability to send CDRs via FTP to a RADIUS sever (the FTP push feature)
- Per-realm accounting control
- Configurable intermediate period
- RADIUS CDR redundancy
- RADIUS CDR content control

Stream Control Transfer Protocol Overview

The Stream Control Transmission Protocol (SCTP) was originally designed by the Signaling Transport (SIGTRAN) group of IETF for Signalling System 7 (SS7) transport over IP-based networks. It is a reliable transport protocol operating on top of an unreliable connectionless service, such as IP. It provides acknowledged, error-free, non-duplicated transfer of messages through the use of checksums, sequence numbers, and selective retransmission mechanism.

SCTP is designed to allow applications, represented as endpoints, communicate in a reliable manner, and so is similar to TCP. In fact, it has inherited much of its behavior from TCP, such as association (an SCTP peer-to-peer connection) setup, congestion control and packet-loss detection algorithms. Data delivery, however, is significantly different. SCTP delivers discrete application messages within multiple logical streams within the context of a single association. This approach to data delivery is more flexible than the single byte-stream used by TCP, as messages can be ordered, unordered or even unreliable within the same association.

SCTP Packets

SCTP packets consist of a common header and one or more chunks, each of which serves a specific purpose.

- DATA chunk — carries user data
- INIT chunk — initiates an association between SCTP endpoints
- INIT ACK chunk — acknowledges association establishment
- SACK chunk — acknowledges received DATA chunks and informs the peer endpoint of gaps in the received subsequences of DATA chunks
- HEARTBEAT chunk — tests the reachability of an SCTP endpoint
- HEARTBEAT ACK chunk — acknowledges reception of a HEARTBEAT chunk
- ABORT chunk — forces an immediate close of an association
- SHUTDOWN chunk — initiates a graceful close of an association
- SHUTDOWN ACK chunk — acknowledges reception of a SHUTDOWN chunk
- ERROR chunk — reports various error conditions
- COOKIE ECHO chunk — used during the association establishment process
- COOKIE ACK chunk — acknowledges reception of a COOKIE ECHO chunk
- SHUTDOWN COMPLETE chunk — completes a graceful association close

SCTP Terminology

This section defines some terms commonly found in SCTP standards and documentation.

SCTP Association

is a connection between SCTP endpoints. An SCTP association is uniquely identified by the transport addresses used by the endpoints in the association. An SCTP association can be represented as a pair of SCTP endpoints, for example, assoc = { [IPv4Addr : PORT1], [IPv4Addr1, IPv4Addr2: PORT2]}.

Only one association can be established between any two SCTP endpoints.

SCTP Endpoint

is a sender or receiver of SCTP packets. An SCTP endpoint may have one or more IP address but it always has one and only one SCTP port number. An SCTP endpoint can be represented as a list of SCTP transport addresses with the same port, for example, endpoint = [IPv6Addr, IPv6Addr: PORT].

An SCTP endpoint may have multiple associations.

SCTP Path

is the route taken by the SCTP packets sent by one SCTP endpoint to a specific destination transport address or its peer SCTP endpoint. Sending to different destination transport addresses does not necessarily guarantee separate routes.

SCTP Primary Path

is the default destination source address, the IPv4 or IPv6 address of the association initiator. For retransmissions however, another active path may be selected, if one is available.

SCTP Stream

is a unidirectional logical channel established between two associated SCTP endpoints. SCTP distinguishes different streams of messages within one SCTP association. SCTP makes no correlation between an inbound and outbound stream.

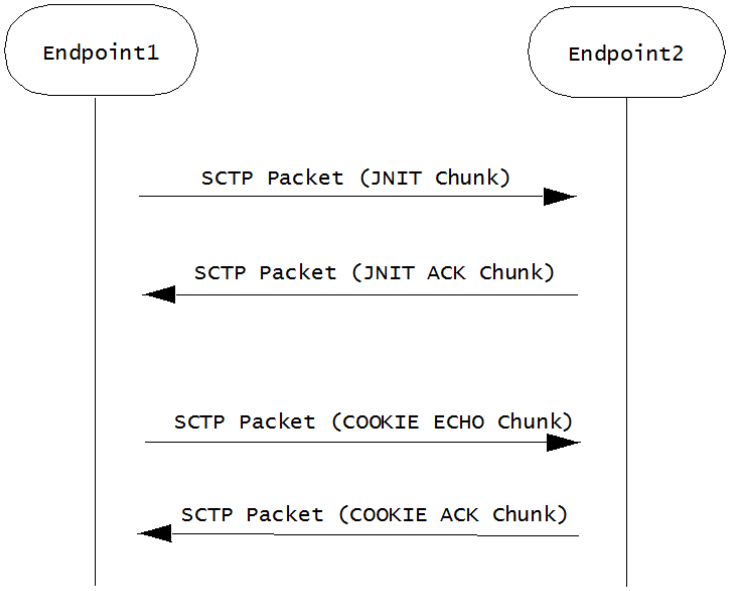
SCTP Transport Address

is the combination of an SCTP port and an IP address. For the current release, the IP address portion of an SCTP Transport Address must be a routable, unicast IPv4 or IPv6 address.

An SCTP transport address binds to a single SCTP endpoint.

SCTP Message Flow

Before peer SCTP users (commonly called endpoints) can send data to each other, an association (an SCTP connection) must be established between the endpoints. During the association establishment process a cookie mechanism is employed to provide protection against security attacks. The following figure shows a sample SCTP association establishment message flow.



Endpoint1 initiates the association by sending Endpoint2 an SCTP packet that contains an INIT chunk, which can include one or more IP addresses used by the initiating endpoint. Endpoint2 acknowledges the initiation of an SCTP association with an SCTP packet that contains an INIT_ACK chunk. This chunk can also include one or more IP addresses at used by the responding endpoint.

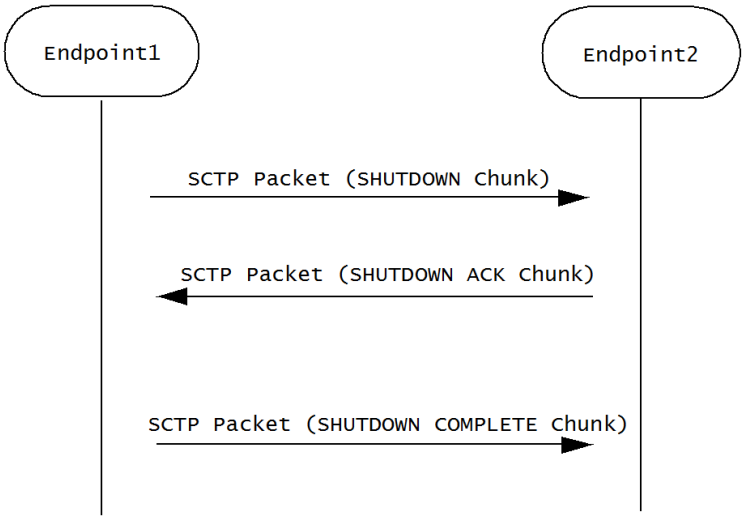
Both the INIT chuck (issued by the initiator) and INIT ACK chunk (issued by the responder) specify the number of outbound streams supported by the association, as well as the maximum inbound streams accepted from the other endpoint.

Association establishment is completed by a COOKIE ECHO/COOKIE ACK exchange that specifies a cookie value used in all subsequent DATA exchanges.

Once an association is successfully established, an SCTP endpoint can send unidirectional data streams using SCTP packets that contain DATA chunks. The recipient endpoint acknowledges with an SCTP packet containing a SACK chunk.

SCTP monitors endpoint reachability by periodically sending SCTP packets that contain HEARTBEAT chunks. The recipient endpoint acknowledges receipt, and confirms availability, with an SCTP packet containing a HEARBEAT ACK chunk.

Either SCTP endpoint can initiate a graceful association close with an SCTP packet that contains a SHUTDOWN chunk. The recipient endpoint acknowledges with an SCTP packet containing a SHUTDOWN ACK chunk. The initiating endpoint concludes the graceful close with an SCTP packet that contains a SHUTDOWN COMPLETE chunk.

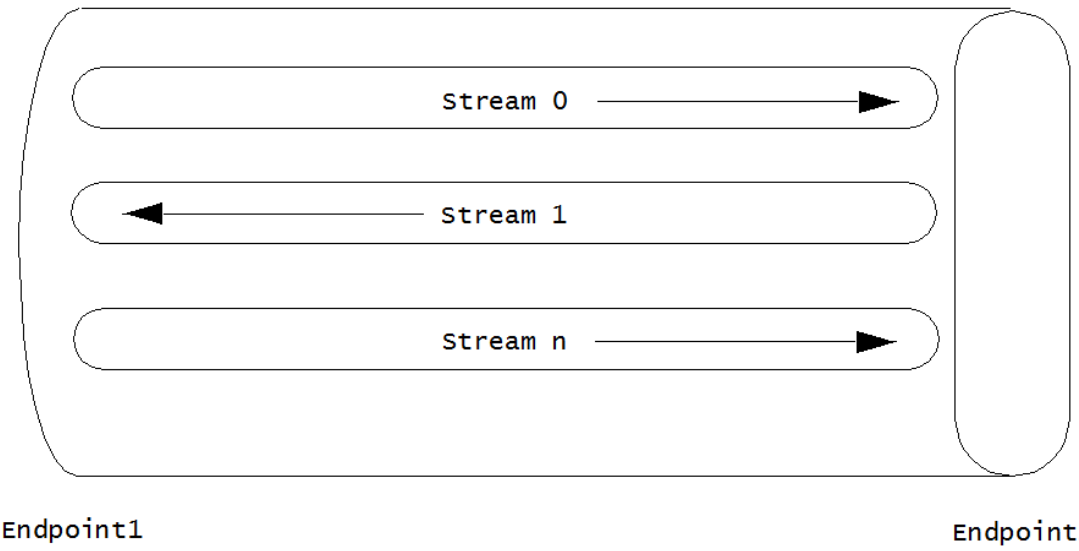


Congestion Control

SCTP congestion control mechanism is similar to that provided by TCP, and includes slow start, congestion avoidance, and fast retransmit. In SCTP, the initial congestion window (cwnd) is set to the double of the maximum transmission unit (MTU) while in TCP, it is usually set to one MTU. In SCTP, cwnd increases based on the number of acknowledged bytes, rather than the number of acknowledgements in TCP. The larger initial cwnd and the more aggressive cwnd adjustment provided by SCTP result in a larger average congestion window and, hence, better throughput performance than TCP.

Multi-Streaming

SCTP supports streams as depicted in the following figure which depicts an SCTP association that supports three streams.



The multiple stream mechanism is designed to solve the head-of-the-line blocking problem of TCP. Therefore, messages from different multiplexed flows do not block one another.

A stream can be thought of as a sub-layer between the transport layer and the upper layer. SCTP supports multiple logical streams to improve data transmission throughput. As shown in the above figure, SCTP allows multiple unidirectional streams within an association. This multiplexing/de-multiplexing capability is called multi-streaming and it is achieved by introducing a field called Stream Identifier contained in every DATA chunk) that is used to differentiate segments in different streams.

SIP transactions are mapped into SCTP streams as described in Section 5.1 of RFC 4168. In what it describes as the simplest way, the RFC suggests (keyword SHOULD) that all SIP messages be transmitted via Stream 0 with the U bit set to 1.

On the transmit side, the current SCTP implementation follows the RFC 4168 recommendation. On the receiving side, a SIP entity must be prepared to receive SIP messages over any stream.

Delivery Modes

SCTP supports two delivery modes, ordered and unordered . Delivery mode is specified by the U bit in the DATA chunk header — if the bit is clear (0), ordered delivery is specified; if the bit is set (1), unordered delivery is specified.

Within a stream, an SCTP endpoint must deliver ordered DATA chunks (received with the U bit set to 0) to the upper layer protocol according to the order of their Stream Sequence Number . Like the U bit, the Stream Sequence Number is a field within the DATA chunk header, and serves to identify the chunk's position with the message stream. If DATA chunks arrive out of order of their Stream Sequence Number, the endpoint must delay delivery to the upper layer protocol until they are reordered and complete.

Unordered DATA chunks (received with the U bit set to 1) are processed differently. When an SCTP endpoint receives an unordered DATA chunk, it must bypass the ordering mechanism and immediately deliver the data to the upper layer protocol (after reassembly if the user data is fragmented by the sender). As a consequence, the Stream Sequence Number field in an unordered DATA chunk has no significance. The sender can fill it with arbitrary value, but the receiver must ignore any value in field.

When an endpoint receives a DATA chunk with the U flag set to 1, it must bypass the ordering mechanism and immediately deliver the data to the upper layer (after reassembly if the user data is fragmented by the data sender).

Unordered delivery provides an effective way of transmitting out-of-band data in a given stream. Note also, a stream can be used as an unordered stream by simply setting the U bit to 1 in all DATA chunks sent through that stream.

Multi-Homing

Call control applications for carrier-grade service require highly reliable communication with no single point of failure. SCTP can assist carriers with its multi-homing capabilities. By providing different paths through the network over separate and diverse means, the goal of no single point of failure is more easily attained.

SCTP built-in support for multi-homed hosts allows a single SCTP association to run across multiple links or paths, hence achieving link/path redundancy. With this capability, and SCTP association can be made to achieve fast failover from one link/path to another with little interruption to the data transfer service.

Multi-homing enables an SCTP host to establish an association with another SCTP host over multiple interfaces identified by different IP addresses. With specific regard to the Oracle

Communications Session Border Controller these IP addresses need not be assigned to the same **phy-interface**, or to the same physical Network Interface Unit.

If the SCTP nodes and the according IP network are configured in such a way that traffic from one node to another travels on physically different paths if different destination IP address are used, associations become tolerant against physical network failures and other problems of that kind.

An endpoint can choose an optimal or suitable path towards a multi-homed destination. This capability increases fault tolerance. When one of the paths fails, SCTP can still choose another path to replace the previous one. Data is always sent over the primary path if it is available. If the primary path becomes unreachable, data is migrated to a different, affiliated address — thus providing a level of fault tolerance. Network failures that render one interface of a server unavailable do not necessarily result in service loss. In order to achieve real fault resilient communication between two SCTP endpoints, the maximization of the diversity of the round-trip data paths between the two endpoints is encouraged.

Multi-Homing and Path Diversity

As previously explained, when a peer is multi-homed, SCTP can automatically switch the subsequent data transmission to an alternative address. However, using multi-homed endpoints with SCTP does not automatically guarantee resilient communications. One must also design the intervening network(s) properly.

To achieve fault resilient communication between two SCTP endpoints, one of the keys is to maximize the diversity of the round-trip data paths between the two endpoints. Under an ideal situation, one can make the assumption that every destination address of the peer will result in a different, separate path towards the peer. Whether this can be achieved in practice depends entirely on a combination of factors that include path diversity, multiple connectivity, and the routing protocols that glue the network together. In a normally designed network, the paths may not be diverse, but there may be multiple connectivity between two hosts so that a single link failure will not fail an association.

In an ideal arrangement, if the data transport to one of the destination addresses (which corresponds to one particular path) fails, the data sender can migrate the data traffic to other remaining destination address(es) (that is, other paths) within the SCTP association.

Monitoring Failure Detection and Recovery

When an SCTP association is established, a single destination address is selected as the primary destination address and all new data is sent to that primary address by default. This means that the behavior of a multi-homed SCTP association when there are no network losses is similar to behavior of a TCP connection. Alternate, or secondary, destination addresses are only used for redundancy purposes, either to retransmit lost packets or when the primary destination address cannot be reached.

A failover to an alternate destination is performed when the SCTP sender cannot elicit an acknowledgement — either a SACK for a DATA chunk, or a HEARTBEAT ACK for a HEARTBEAT chunk — for a configurable consecutive number of transmissions. The SCTP sender maintains an error-counter is maintained for each destination address and if this counter exceeds a threshold (normally six), the address is marked as inactive, and taken out of service. If the primary destination address is marked as inactive, all data is then switched to a secondary address to complete the failover.

If no data has been sent to an address for a specified time, that endpoint is considered to be idle and a HEARTBEAT packet is transmitted to it. The endpoint is expected to respond to the HEARTBEAT immediately with a HEARTBEAT ACK. As well as monitoring the status of

destination addresses, the HEARTBEAT is used to obtain RTT measurements on idle paths. The primary address becomes active again if it responds to a heartbeat.

The number of events where heartbeats were not acknowledged within a certain time, or retransmission events occurred is counted on a per association basis, and if a certain limit is exceeded, the peer endpoint is considered unreachable, and the association is closed.

The threshold for detecting an endpoint failure and the threshold for detecting a failure of a specific IP addresses of the endpoint are independent of each other. Each parameter can be separately configured by the SCTP user. Careless configuration of these protocol parameters can lead the association onto the dormant state in which all the destination addresses of the peer are found unreachable while the peer still remains in the reachable state. This is because the overall retransmission counter for the peer is still below the set threshold for detecting the peer failure.

Configuring SCTP Support for SIP

RFC 4168, *The Stream Control Transfer Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)*, specifies the requirements for SCTP usage as a layer 4 transport for SIP. Use the following steps to:

- configure SCTP as the layer 4 transport for a SIP interface
- create an SCTP-based SIP port
- associate **phy-interfaces/network interfaces** with SIP realms
- identify adjacent SIP servers that are accessible via SCTP
- set SCTP timers and counters (optional)

Configuring an SCTP SIP Port

SIP ports are created as part of the SIP Interface configuration process.

1. From superuser mode, use the following command sequence to access sip-port configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# sip-ports
ORACLE(sip-port)#
```

2. Use the **address** parameter to provide the IPv4 or IPv6 address of the network interface that supports the SIP port.

   This is the primary address of a the local multi-homed SCTP endpoint.

```
ORACLE(sip-port)# address 172.16.10.76
ORACLE(sip-port)#
```

3. Retain the default value, 5060 (the well-known SIP port) for the **port** parameter.

```
ORACLE(sip-port)# port 5060
ORACLE(sip-port)#
```

4. Use the **transport-protocol** parameter to identify the layer 4 protocol.

Supported values are UDP, TCP, TLS, and SCTP.

Select SCTP.

```
ORACLE(sip-port)# transport-protocol sctp
ORACLE(sip-port)#
```

5. Use the **multi-homed-addrs** parameter to specify one or more local secondary addresses of the SCTP endpoint.

   Multi-homed addresses must be of the same type (IPv4 or IPv6) as that specified by the **address** parameter. Like the address parameter, these addresses identify SD network interfaces.

   To specify multiple addresses, bracket an address list with parentheses.

```
ORACLE(sip-port)# multi-homed-addrs 182.16.10.76
ORACLE(sip-port)#
ORACLE(sip-port)# multi-homed-addrs (182.16.10.76 192.16.10.76
196.15.32.108)
ORACLE(sip-port)#
```

6. Remaining parameters can be safely ignored.
7. Use **done**, **exit**, and **verify-config** to complete configuration of this SCTP-based SIP port.

```
ORACLE(sip-port)# done
ORACLE(sip-interface)# exit
ORACLE(session-router)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Configuring the Realm

After configuring a SIP port which identifies primary and secondary multi-homed transport addresses, you identify the network interfaces that support the primary address and secondary addresses to the realm assigned during SIP Interface configuration.

1. From superuser mode, use the following command sequence to access realm-config configuration mode.

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

2. Use the **select** command to access the target realm.
3. Use the **network-interfaces** command to identify the network interfaces that support the SCTP primary and secondary addresses.

   Network interfaces are identified by their name.

Enter a list of network interface names using parentheses as list brackets. The order of interface names is not significant.

```
ORACLE(realm-config)# network-interfaces (mo1 M10)
ORACLE(realm-config)#
```

4. Use **done**, **exit**, and **verify-config** to complete realm configuration.

```
ORACLE(realm-config)# done
ORACLE(media-manager)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Configuring Session Agents

After configuring the realm, you identify adjacent SIP servers who will be accessed via the SCTP protocol.

1. From superuser mode, use the following command sequence to access session-agent configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

2. Use the **select** command to access the target session-agent.
3. Use the **transport-method** parameter to select the layer 4 transport protocol.

   Select staticSCTP for SCTP transport

```
ORACLE(session-agent)# transport-method staticSCTP
ORACLE(session-agent)#
```

4. Set the **reuse-connections** parameter to none.

   Select staticSCTP for SCTP transport

```
ORACLE(session-agent)# reuse-connections none
ORACLE(session-agent)#
```

5. Use **done**, **exit**, and **verify-config** to complete session agent configuration.

```
ORACLE(session-agent)# done
ORACLE(session-router)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

6. Repeat Steps 1 through 5 as necessary to configure additional session agents who will be accessed via SCTP transport.

Setting SCTP Timers and Counters

Setting SCTP timers and counters is optional. All configurable timers and counters provide default values that conform to recommended values as specified in RFC 4960, Stream Control Transmission Protocol.

Management of Retransmission Timer, section 6.3 of RFC 4960 describes the calculation of a Retransmission Timeout (RTO) by the SCTP process. This calculation involves three SCTP protocol parameters: RTO.Initial, RTO.Min, and RTO.Max. Suggested SCTP Protocol Parameter Values section 15 of RFC 4960 lists recommended values for these parameters.

The following shows the equivalence of recommended values and ACLI defaults.

RTO.Initial = 3 seconds **sctp-rto-initial = 3000 ms (default value)**

RTO.Min = 1 second **sctp-rto-min = 1000 ms (default value)**

RTO.Max = 60 seconds **sctp-rto-max = 60000 ms (default value)**

Path Heartbeat, section 8.3 of RFC 4960 describes the calculation of a Heartbeat Interval by the SCTP process. This calculation involves the current calculated RTO and a single SCTP protocol parameter — HB.Interval.

The following shows the equivalence of recommended the value and ACLI default.

HB.Interval = 30 seconds **sctp-hb-interval = 3000 ms (default value)**

Acknowledgement on Reception of DATA Chunks, section 6.2 of RFC 4960 describes requirements for the timely processing and acknowledgement of DATA chunks. This section requires that received DATA chunks must be acknowledged within 500 milliseconds, and recommends that DATA chunks should be acknowledged with 200 milliseconds. The interval between DATA chunk reception and acknowledgement is specific by the ACLI **sctp-sack-timeout** parameter, which provides a default value of 200 milliseconds and a maximum value of 500 milliseconds.

Transmission of DATA Chunks, section 6.1 of RFC 4960 describes requirements for the transmission of DATA chunks. To avoid network congestion the RFC recommends a limitation on the volume of data transmitted at one time. The limitation is expressed in terms of DATA chunks, not in terms of SCTP packets.

The maximum number of DATA chunks that can be transmitted at one time is specified by the ACLI **sctp-max-burst** parameter, which provides a default value of 4 chunks, the limit recommended by the RFC.

Setting the RTO

An SCTP endpoint uses a retransmission timer to ensure data delivery in the absence of any feedback from its peer. RFC 4960 refers to the timer itself as T3-rtx and to the timer duration as RTO (retransmission timeout).

When an endpoint's peer is multi-homed, the endpoint calculates a separate RTO for each IP address affiliated with the peer. The calculation of RTO in SCTP is similar to the way TCP calculates its retransmission timer. RTO fluctuates over time in response to actual network conditions. To calculate the current RTO, an endpoint maintains two state variables per destination IP address — the SRTT (smoothed round-trip time) variable, and the RTTVAR (round-trip time variation) variable.

Use the following procedure to assign values used in RTO calculation.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

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

2. Use the **sctp-rto-initial** parameter to assign an initial timer duration.

   Allowable values are integers within the range 0 through 4294967295 that specify the initial duration in milliseconds. In the absence of an explicitly configured integer value, **sctp-rto-initial** defaults to 3000 milliseconds (3 seconds, the recommended default value from RFC 4960).

   As described in Section 6.3 of RFC 4960, the value specified by **sctp-rto-initial** is assigned to the SCTP protocol parameter RTO.Initial, which provides a default RTO until actual calculations have derived a fluctuating duration based on network usage. The value specified by the **sctp-rto-initia**l parameter seeds these calculations.

```
ORACLE(network-parameters)# sctp-rto-initial 3000
ORACLE(network-parameters)#
```

3. Use the **sctp-rto-min** and **sctp-rto-max** parameters to assign an RTO floor and ceiling.

   Allowable values are integers within the range 0 through 4294967295 that specify the minimum and maximum durations in milliseconds. In the absence of an explicitly configured integer value, **sctp-rto-min** defaults to 1000 ms (1 second, the recommended default value from RFC 4960), and **sctp-rto-max** defaults to 60000 ms (60 seconds, the recommended default value from RFC 4960.)

   As described in Section 6.3 of RFC 4960, the values specified by **sctp-rto-min** and **sctp-rto-max** are assigned to the SCTP protocol parameters, RTO.min and RTO.max that limit RTO calculations. If a calculated RTO duration is less than RTO.min, the parameter value is used instead of the calculated value; likewise, if a calculated RTO duration is greater than RTO.max, the parameter value is used instead of the calculated value.

```
ORACLE(network-parameters)# sctp-rto-min 1000
ORACLE(network-parameters)# sctp-rto-max 60000
ORACLE(network-parameters)#
```

4. Use **done**, **exit**, and **verify-config** to complete RTO configuration.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Setting the Heartbeat Interval

Both single-homed and multi-homed SCTP endpoints test the reachability of associates by sending periodic HEARTBEAT chunks to UNCONFIRMED or idle transport addresses.

Use the following procedure to assign values used in Heartbeat Interval calculation.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

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

2. Use the **sctp-hb-interval** parameter to assign an initial Heartbeat Interval duration.

   Allowable values are integers within the range 0 through 4294967295 that specify the initial Heartbeat Interval in milliseconds. In the absence of an explicitly configured integer value, **sctp-hb-interval** defaults to 30000 milliseconds (30 seconds, the recommended default value from RFC 4960).

   As described in Section 8.3 of RFC 4960, the value specified by **sctp-hb-interval** is assigned to the SCTP protocol parameter HB.Interval, which provides a default interval until actual calculations have derived a fluctuating interval based on network usage. The value specified by the **sctp-hb-interval** parameter is used during these calculations.

```
ORACLE(network-parameters)# sctp-hb-interval 30000
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete Heartbeat Interval configuration.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE #
```

Setting the SACK Delay Timer

An SCTP Selective Acknowledgement (SACK) is sent to the peer endpoint to acknowledge received DATA chunks and to inform the peer endpoint of gaps in the received subsequences of DATA chunks. Section 6.2 of RFC 4960 sets a specific requirement for a SACK Delay timer that specifies the maximum interval between the reception of an SCTP packet containing one or more DATA chunks and the transmission of a SACK to the packet originator.

Use the following procedure to set the SACK Delay timer.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

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

```
ORACLE(system)# network-parameters
ORACLE(network-parameters)#
```

2. Use the **sctp-sack-timeout** parameter to assign a value to the SACK Delay timer.

   Allowable values are integers within the range 0 through 500 which specify the maximum delay (in milliseconds) between reception of a SCTP packet containing one or more Data chunks and the transmission of a SACK to the packet source. The value 0 indicates that a SACK is generated immediately upon DATA chunk reception

   In the absence of an explicitly configured integer value, **sctp-sack-timeout** defaults to 200 ms (the recommended default value from RFC 4960).

```
ORACLE(network-parameters)# sctp-sack-timeout 200
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete configuration of the SACK Delay timer.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Limiting DATA Bursts

Section 6.1 of RFC 4960 describes the SCTP protocol parameter, Max.Burst, used to limit the number of DATA chunks that are transmitted at one time.

Use the following procedure to assign a value to the SCTP protocol parameter, Max.Burst.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

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

2. Use the **sctp-max-burst** parameter to assign a value to the SCTP protocol parameter, Max.Burst.

   Allowable values are integers within the range 0 through 4294967295 that specify the maximum number of DATA chunks that will be sent at one time. In the absence of an explicitly configured integer value, **sctp-max-burst** defaults to 4 (DATA chunks, the recommended default value from RFC 4960).

```
ORACLE(network-parameters)# sctp-max-burst 4
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete configuration of DATA burst limitations.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
```

```
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Setting Endpoint Failure Detection

As described in Monitoring, Failure Detection and Recovery, a single-homed SCTP endpoint maintains a count of the total number of consecutive failed (unacknowledged) retransmissions to its peer. Likewise, a multi-homed SCTP endpoint maintains a series of similar, dedicated counts for all of its destination transport addresses. If the value of these counts exceeds the limit indicated by the SCTP protocol parameter Association.Max.Retrans, the endpoint considers the peer unreachable and stops transmitting any additional data to it, causing the association to enter the CLOSED state.

The endpoint resets the counter when (1) a DATA chunk sent to that peer endpoint is acknowledged by a SACK, or (2) a HEARTBEAT ACK is received from the peer endpoint.

Use the following procedure to configure endpoint failure detection.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

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

2. Use the **sctp-assoc-max-retrans** to assign a value to the SCTP protocol parameter Association.Max.Retrans.

   Allowable values are integers within the range 0 through 4294967295 which specify the maximum number of transmission requests. In the absence of an explicitly configured integer value, s**ctp-assoc-max-retrans** defaults to 10 (transmission re-tries, the recommended default value from RFC 4960).

   ```
   ORACLE(network-parameters)# sctp-assoc-max-retrans 10
   ORACLE(network-parameters)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete endpoint failure detection configuration.

   ```
   ORACLE(network-parameters)# done
   ORACLE(system)# exit
   ORACLE(configure)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   -------------------------------------------------------------------------
   Verification successful! No errors nor warnings in the configuration
   ORACLE#
   ```

Setting Path Failure Detection

As described in Monitoring, Failure Detection and Recovery, when its peer endpoint is multi-homed, an SCTP endpoint maintains a count for each of the peer's destination transport addresses.

Each time the T3-rtx timer expires on any address, or when a HEARTBEAT sent to an idle address is not acknowledged within an RTO, the count for that specific address is incremented. If the value of a specific address count exceeds the SCTP protocol parameter Path.Max.Retrans, the endpoint marks that destination transport address as inactive.

The endpoint resets the counter when (1) a DATA chunk sent to that peer endpoint is acknowledged by a SACK, or (2) a HEARTBEAT ACK is received from the peer endpoint.

When the primary path is marked inactive (due to excessive retransmissions, for instance), the sender can automatically transmit new packets to an alternate destination address if one exists and is active. If more than one alternate address is active when the primary path is marked inactive, a single transport address is chosen and used as the new destination transport address.

Use the following procedure to configure path failure detection.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

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

2. Use the **sctp-path-max-retrans** parameter to assign a value to the SCTP protocol parameter Path.Max.Retrans.

   Allowable values are integers within the range 0 through 4294967295 that specify the maximum number of RTOs and unacknowledged HEARTBEATS. In the absence of an explicitly configured integer value, **sctp-path-max-retrans** defaults to 5 (RTO and/or HEARTBEAT errors per transport address, the recommended default value from RFC 4960).

   When configuring endpoint and path failure detection, ensure that the value of the **sctp-assoc-max-retrans** parameter is smaller than the sum of the **sctp-path-max-retrans** values for all the remote peer's destination addresses. Otherwise, all the destination addresses can become inactive (unable to receive traffic) while the endpoint still considers the peer endpoint reachable.

   ```
   ORACLE(network-parameters)# sctp-path-max-retrans 5
   ORACLE(network-parameters)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete path failure detection configuration.

   ```
   ORACLE(network-parameters)# done
   ORACLE(system)# exit
   ORACLE(configure)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   ----------------------------------------------------------------------
   ```

```
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Specifying the Delivery Mode

As described in Delivery Modes, SCTP support two delivery modes, ordered and unordered.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

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

2. Use the **sctp-send-mode** parameter to select the preferred delivery mode.

   Choose ordered or unordered.

   ```
   ORACLE(network-parameters)# sctp-send-mode unordered
   ORACLE(network-parameters)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete delivery mode configuration.

   ```
   ORACLE(network-parameters)# done
   ORACLE(system)# exit
   ORACLE(configure)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   ----------------------------------------------------------------------
   Verification successful! No errors nor warnings in the configuration
   ORACLE #
   ```

Example Configurations

The following ACLI command sequences summarize required SCTP port configuration, and the configuration of required supporting elements.

- PHY interfaces
- Network interfaces
- SIP ports
- realms
- session agents

Sequences show only configuration parameters essential for SCTP operations; other parameters can retain default values, or assigned other values specific to local network requirements.

Phy Interface Configuration

The first ACLI command sequence configures a **phy-interface** named m10, that will support an SCTP primary address; the second sequence configures a **phy-interface** named m01 that will support a secondary SCTP address.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# phy-interface
ORACLE(phy-interface)# operation-type media
ORACLE(phy-interface)# port 0
ORACLE(phy-interface)# slot 1
ORACLE(phy-interface)# name m10
ORACLE(phy-interface)#
...
...
...
ORACLE(phy-interface)#
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# phy-interface
ORACLE(phy-interface)# operation-type media
ORACLE(phy-interface)# port 1
ORACLE(phy-interface)# slot 0
ORACLE(phy-interface)# name m01
ORACLE(phy-interface)#
...
...
...
ORACLE(phy-interface)#
```

Network Interface Configuration

These ACLI command sequences configure two **network-interface**s. The first sequence configures a **network-interface** named m10, thus associating the **network-interface** with the **phy-interface** of the same name. The ACLI **ip-address** command assigns the IPv4 address 172.16.10.76 to the **network-interface**. In a similar fashion, the second command sequence associates the m01 network and **phy-interfaces**, and assigns an IPv4 address of 182.16.10.76.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-interface
ORACLE(network-interface)# name m10
ORACLE(network-interface)# ip-address 172.16.10.76
...
...
...
ORACLE(network-interface)#
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-interface
ORACLE(network-interface)# name m01
ORACLE(network-interface)# ip-address 182.16.10.76
```

```
...
...
...
ORACLE(network-interface)#
```

SIP Port Configuration

This ACLI command sequence configures a SIP port for SCTP operations. It specifies the use of SCTP as the transport layer protocol, and assigns the existing network interface address, 172.16.10.76, as the SCTP primary address. Additionally, it identifies three other existing network addresses (182.16.10.76, 192.16.10.76, and 196.15.32.108) as SCTP secondary addresses.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# sip-ports
ORACLE(sip-port)# address 172.16.10.76
ORACLE(sip-port)# transport-protocol sctp
ORACLE(sip-port)# multi-homed-addrs (182.16.10.76 192.16.10.76 196.15.32.108)
...
...
...
ORACLE(sip-port)#
```

Realm Configuration

These ACLI command sequences configure a realm for SCTP operations. The first ACLI sequence assigns a named realm, in this example core-172, to a SIP interface during the interface configuration process. The second sequence accesses the target realm and uses the **network-interfaces** command to associate the named SCTP network interfaces with the realm.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# realm-id core-172
...
...
...
ORACLE(sip-interface)#
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
identifier: core-172
1. core-172 ...
selection: 1
ORACLE(realm-config)# network-interfaces (m01 m10 ...)
...
...
...
ORACLE(ream-config)#
```

Session Agent Configuration

The final ACLI command sequence enables an SCTP-based transport connection between the Oracle Communications Session Border Controller and an adjacent network element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)# select
<hostname>: core-172S1
1. core-172S1 ...
selection: 1
ORACLE(session-agent)#
ORACLE(session-agent)# transport-method staticSCTP
ORACLE(session-agent)# reuse-connections none
...
...
...
ORACLE(session-agent)#
```

IPv6 Address Configuration

This section calls out the configurations and parameters for which you can enter IPv6 addresses. In this first IPv6 implementation, the complete range of system configurations and their parameters are available for IPv6 use.

The Oracle Communications Session Border Controller follows RFC 3513 its definition of IPv6 address representations. Quoting from that RFC, these are the two forms supported:

- The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. Examples:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

1080:0:0:0:8:800:200C:417A

Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.).

- Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. For example, the following addresses: 1080:0:0:0:8:800:200C:417A a unicast address FF01:0:0:0:0:0:0:101 a multicast address
  0:0:0:0:0:0:0:1 the loopback address

  0:0:0:0:0:0:0:0 the unspecified addresses

  may be represented as:

  1080::8:800:200C:417A a unicast address

  FF01::101 a multicast address

  ::1 the loopback address

  :: the unspecified addresses

> **Note:**
>
> For ACLI parameters that support only IPv4, there are many references to that version as the accepted value for a configuration parameter or other IPv4-specific languages. For IPv6 support, these references have been edited. For example, rather than providing help that refers specifically to IPv4 addresses when explaining what values are accepted in an ACLI configuration parameter, you will now see an <ipAddr> note.

Access Control

These are the IPv6-enabled parameters in the **access-control** configuration.

Parameter	Entry Format
source-address	<ip-address>[/<num-bits>][:<port>[/<port-bits>]]
destination-address	<ip-address>[/<num-bits>][:<port>[/<port-bits>]]

Host Route

These are the IPv6-enabled parameters in the **host-route** configuration.

Parameter	Entry Format
dest-network	<ipv4> | <ipv6>
netmask	<ipv4> | <ipv6>
gateway	<ipv4> | <ipv6>

Local Policy

These are the IPv6-enabled parameters in the **local-policy** configuration.

Parameter	Entry Format
from-address	<ipv4> | <ipv6> | POTS Number, E.164 Number, hostname, wildcard
to-address	<ipv4> | <ipv6> | POTS Number, E.164 Number, hostname, wildcard

Network Interface

These are the IPv6-enabled parameters in the **network-interface** configuration.

Parameter	Entry Format
hostname	<ipv4> | <ipv6> | hostname
ip-address	<ipv4> | <ipv6>
pri-utility-addr	<ipv4> | <ipv6>
sec-utility-addr	<ipv4> | <ipv6>
netmask	<ipv4> | <ipv6>
gateway	<ipv4> | <ipv6>
sec-gateway	<ipv4> | <ipv6>
dns-ip-primary	<ipv4> | <ipv6>

Parameter	Entry Format
dns-ip-backup1	<ipv4> | <ipv6>
dns-ip-backup2	<ipv4> | <ipv6>
add-hip-ip	<ipv4> | <ipv6>
remove-hip-ip	<ipv4> | <ipv6>
add-icmp-ip	<ipv4> | <ipv6>
remove-icmp-ip	<ipv4> | <ipv6>

ENUM Server

These are the IPv6-enabled parameters in the enum-config.

Parameter	Entry Format
enum-servers	[<ipv4> | <ipv6>]:port

Realm Configuration

These are the IPv6-enabled parameters in the **realm-config**.

Parameter	Entry Format
addr-prefix	[<ipv4> | <ipv6>]/prefix

Session Agent

These are the IPv6-enabled parameters in the **session-agent** configuration.

Parameter	Entry Format
hostname	<ipv4> | <ipv6>
ip-address	<ipv4> | <ipv6>

SIP Configuration

These are the IPv6-enabled parameters in the **session-config**.

Parameter	Entry Format
registrar-host	<ipv4> | <ipv6> | hostname | *

SIP Interface SIP Ports

These are the IPv6-enabled parameters in the **sip-interface>sip-ports** configuration.

Parameter	Entry Format
address	<ipv4> | <ipv6>

Steering Pool

These are the IPv6-enabled parameters in the **steering-pool** configuration.

Parameter	Entry Format
ip-address	<ipv4> | <ipv6>

System Configuration

These are the IPv6-enabled parameters in the **system-config**.

Parameter	Entry Format
default-v6-gateway	<ipv6>

Account Server

These are the IPv6-enabled parameters in the **account-server** configuration.

Parameter	Entry Format
hostname	<ip-address>[/<num-bits>][:<port>[/<port-bits>]]
dns-query-type	<none><A><AAAA>

> **Note:**
>
> When set to none, the system refers to the interface associated with the dns-realm setting in the account-config. If the interface has an IPv6 address, the system performs an AAAA query.

IPv6 Support for Management and Telemetry

Several management-oriented parameters on the Oracle Communications Session Border Controller may be configured with IPv6 addresses to be used within IPv6 networks.

The following parameters that are configured with IP addresses accept IPv6 addresses to be used within IPv6 address space.

You may configure the wancom0/eth0 physical interface in the bootparams with an IPv6 address and complementary IPv6 gateway via the following parameters:

- **bootparams**, **inet on ethernet**
- **bootparams**, **gateway inet**

You may configure a syslog server with an IPv6 destination address via the following parameter:

- **system**, **system-config**, **syslog-servers**, **address**

You may configure a system access list entry with an IPv6 source address and complementary IPv6 gateway.

- **system**, **system-access-list**, **source-address**
- **system**, **system-access-list**, **netmask**

You may configure a RADIUS server with an IPv6 destination address via the following parameter:

- **security**, **authentication**, **radius-servers**, **address**

IPv6 Default Gateway

In the system configuration, you configure a default gateway—a parameter that now has its own IPv6 equivalent.

To configure an IPv6 default gateway:

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

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

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

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

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

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

4. **default-v6-gateway**—Set the IPv6 default gateway for this Oracle Communications Session Border Controller. This is the IPv6 egress gateway for traffic without an explicit destination. The application of your Oracle Communications Session Border Controller determines the configuration of this parameter.
5. Save your work.

IPv6 Link Local Addresses

The Oracle Communications Session Border Controller supports IPv6 Link Local addresses configured for a network interface's gateway.

An IPv6 link local address is signified by its first hextet set to FE80:. Even if a network interface's first hextet is not FE80, but the gateway is, the Oracle Communications Session Border Controller will still function as expected.

show neighbor-table

The show neighbor-table command displays the IPv6 neighbor table and validates that there is an entry for the link local address, and the gateway uses that MAC address.

```
System# show neighbor-table
LINK LEVEL NEIGHBOR TABLE
Neighbor                              Linklayer Address  Netif Expire    S
```

```
Flags
300::100                                0:8:25:a1:ab:43      sp0 permanent ? R
871962224
400::100                                0:8:25:a1:ab:45      sp1 permanent ? R
871962516
fe80::bc02:a98f:f61e:20%sp0             be:2:ac:1e:0:20      sp0 4s        ? R
871962808
fe80::bc01:a98f:f61e:20%sp1             be:1:ac:1e:0:20      sp1 4s        ? R
871963100
-------------------------------------------------------------------------------
ICMPv6 Neighbor Table:
-------------------------------------------------------------------------------
------------
  entry: slot port vlan IP                                type       flag
pendBlk Hit MAC
-------------------------------------------------------------------------------
------------
  5    : 1    0    0    fe80::bc01:a98f:f61e:20/64     08-DYNAMIC 1
0       1   be:01:ac:1e:00:20
  4    : 1    0    0    0.0.0.0/64                     01-GATEWAY 0
0       1   be:01:ac:1e:00:20
  3    : 1    0    0    400::/64                       02-NETWORK 0
0       1   00:00:00:00:00:00
  2    : 0    0    0    fe80::bc02:a98f:f61e:20/64     08-DYNAMIC 1
0       1   be:02:ac:1e:00:20
  1    : 0    0    0    0.0.0.0/64                     01-GATEWAY 0
0       1   be:02:ac:1e:00:20
  0    : 0    0    0    300::/64                       02-NETWORK 0
0       1   00:00:00:00:00:00
-------------------------------------------------------------------------------
------------
```

Network Interfaces and IPv6

You set many IP addresses in the network interface, one of which is the specific IP address for that network interface and others that are related to different types of management traffic. This section outlines rules you must follow for these entries.

- For the **network-interface ip-address** parameter, you can set a single IP address. When you are working with an IPv6-enabled system, however, note that all other addresses related to that network-interface IP address must be of the same version.
- Heterogeneous address family configuration is prevented for the **dns-ip-primary**, **dns-ip-backup1**, and **dns-ip-backup2** parameters.
- For HIP addresses (**add-hip-ip**), you can use either IPv4 or IPv6 entries.
- For ICMP addresses (**add-icmp-ip**), you can use either IPv4 or IPv6 entries.

IPv6 Reassembly and Fragmentation Support

As it does for IPv4, the Oracle Communications Session Border Controller supports reassembly and fragmentation for large signaling packets when you enable IPV6 on your system.

The Oracle Communications Session Border Controller takes incoming fragments and stores them until it receives the first fragment containing a Layer 4 header. With that header information, the Oracle Communications Session Border Controller performs a look-up so it can forward the packets to its application layer. Then the packets are re-assembled at the applications layer. Media fragments, however, are not reassembled and are instead forwarded to the egress interface.

On the egress side, the Oracle Communications Session Border Controller takes large signaling messages and encodes it into fragment datagrams before it transmits them.

Note that large SIP INVITE messages should be sent over TCP. If you want to modify that behavior, you can use the SIP interface's option parameter **max-udp-length=xx** for each SIP interface where you expect to receive large INVITE packets.

Other than enabling IPv6 on your Oracle Communications Session Border Controller, there is no configuration for IPv6 reassembly and fragmentation support. It is enabled automatically.

Access Control List Support

The Oracle Communications Session Border Controller supports IPv6 for access control lists in two ways:

- For static access control lists that you configure in the **access-control** configuration, your entries can follow IPv6 form. Further, this configuration supports a prefix that enables wildcarding the source IP address.
- Dynamic ACLs are also supported; the Oracle Communications Session Border Controller will create ACLs for offending IPv6 endpoints.

Data Entry

When you set the **source-address** and **destination-address** parameters in the **access-control** configuration, you will use a slightly different format for IPv6 than for IPv4.

For the **source-address**, your IPv4 entry takes the following format: <ip-address>[/<num-bits>][:<port>[/<port-bits>]]. And for the **destination-address**, your IPv4 entry takes this format: <ip-address>[:<port>[/<port-bits>]].

Since the colon (:) in the IPv4 format leads to ambiguity in IPv6, your IPv6 entries for these settings must have the address encased in brackets ([]): [7777::11]/64:5000/14.

In addition, IPv6 entries are allowed up to 128 bits for their prefix lengths.

The following is an example access control configuration set up with IPv6 addresses.

```
ORACLE(access-control)# done
access-control
        realm-id                        net7777
        description
        source-address                  7777::11/64:5060/8
        destination-address             8888::11:5060/8
        application-protocol            SIP
        transport-protocol              ALL
        access                          deny
        average-rate-limit              0
        trust-level                     none
        minimum-reserved-bandwidth      0
        invalid-signal-threshold        10
```

```
maximum-signal-threshold       0
untrusted-signal-threshold     0
deny-period                    30
```

Homogeneous Realms

IPv6 is supported for realms and for nested realms, as long as the parent chain remains within the same address family. If you try to configure realms with mixed IPv4-IPv6 addressing, your system will issue an error message when you try to save your configuration. This check saves you time because you do not have to wait to run a configuration verification (using the ACLI **verify-config** command) to find possible errors.

Parent-Child Network Interface Mismatch

Your system will issue the following error message if parent-child realms are on different network interfaces that belong to different address families:

```
ERROR: realm-config [child] and parent [net8888] are on network interfaces
that belong to different address families
```

Address Prefix-Network Interface Mismatch

If the address family and the address-prefix you configure for the realm does not match the address family of its network interface, your system will issue the following error message:

```
ERROR: realm-config [child] address prefix and network interface [1:1:0]
belong to different address families
```

RADIUS Support for IPv6

The Oracle Communications Session Border Controller's RADIUS support includes:

- RADIUS CDR generation for SIPv6-SIPv6 and SIPv6-SIPv4 calls
- IPv6-based addresses in RADIUS CDR attributes

The sixteen-byte requirement for IPv6 addresses is supported, and there is a set of attributes with the type ipv6addr. Attributes 155-170 are reserved for the IPv6 addresses.

NAS addresses use the number 95 to specify the NAS-IPV6-Address attribute. And local CDRs now contain IPv6 addresses.

Supporting RADIUS VSAs

The following VSAs have been added to the Oracle RADIUS dictionary to support IPv6.

```
Acme-Flow-In-Src-IPv6_Addr_FS1_F        155     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS1_F        156     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS1_F       157     ipv6addr      Acme
Acme-Flow-Out-Dst-IPv6_Addr_FS1_F       158     ipv6addr      Acme
Acme-Flow-In-Src-IPv6_Addr_FS1_R        159     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS1_R        160     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS1_R       161     ipv6addr      Acme
```

```
Acme-Flow-Out-Dst-IPv6_Addr_FS1_R       162     ipv6addr      Acme
Acme-Flow-In-Src-IPv6_Addr_FS2_F        163     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS2_F        164     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS2_F       165     ipv6addr      Acme
Acme-Flow-Out-Dst-IPv6_Addr_FS2_F       166     ipv6addr      Acme
Acme-Flow-In-Src-IPv6_Addr_FS2_R        167     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS2_R        168     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS2_R       169     ipv6addr      Acme
Acme-Flow-Out-Dst-IPv6_Addr_FS2_R       170     ipv6addr      Acme
```

NTP Synchronization

This section provides information about how to set and monitor NTP on your Oracle Communications Session Border Controller.

When an NTP server is unreachable or when NTP service goes down, the Oracle Communications Session Border Controller generates traps for those conditions. Likewise, the Oracle Communications Session Border Controller clears those traps when the conditions have been rectified. The Oracle Communications Session Border Controller considers a configured NTP server to be unreachable when its reach number (whether or not the NTP server could be reached at the last polling interval; successful completion augments the number) is 0. You can see this value for a server when you use the ACLI **show ntp server** command.

- The traps for when a server is unreachable and then again reachable are: **apSysMgmtNTPServerUnreachableTrap** and **apSysMgmtNTPServerUnreachableClearTrap**
- The traps for when NTP service goes down and then again returns are: **apSysMgmtNTPServiceDownTrap** and **apSysMgmtNTPServiceDownClearTrap**

> **Note:**
>
> The Oracle Communications Session Border Controller does not support NTP service over wancom0 when that interface is configured for a VLAN.

Setting NTP Synchronization

When the SBC requires time-critical processing, you can set NTP for time synchronization. Setting NTP synchronizes both the hardware and the software clocks with the reference time from an NTP server that you specify. NTP is most useful for synchronizing multiple devices located on one network, or across many networks, to a reference time standard.

To guard against NTP server failure, NTP is restarted periodically to support the dynamic recovery of an NTP server.

Note that **ntp-sync** works only by way of the management interface and only on wancom0. Do not configure **ntp-sync** by way of the media interface or any other port.

To set NTP synchronization:

1. In the ACLI's configure terminal section, type **ntp-sync** and then press Enter to access the NTP configuration.

```
ORACLE# configure terminal
ORACLE(configure)# ntp-sync
ORACLE(ntp-config)#
```

2. To add an NTP server, type **add-server**, the Space bar, then the FQDN, IPv4, or IPv6 address of the server and then press the Enter key.

   For FQDN configuration, see FQDNs for Time Servers on the SBC below.

   For example, this entry adds the NTP server at the Massachusetts Institute of Technology in Cambridge, MA:

```
ORACLE(ntp-config)# add-server 18.26.4.105
```

3. To delete an NTP server, type **delete-server**, the Space bar, and the IPv4 or IPv6 address of the server you want to delete and then press the Enter key.

```
ORACLE(ntp-config)# del-server 18.26.4.105
```

FQDNs for Time Servers on the SBC

You can configure the SBC with an FQDN for establishing communications with NTP time servers. This feature supports FQDN resolution through a DNS query over wancom or media interfaces. Having received DNS resolution for the query, the SBC uses its standard selection process for DNS results to request time synchronization from one of multiple, redundant NTP servers.

The SBC includes a DNS client that it uses for FQDN resolution purposes within several contexts, including NTP server address resolution. You set the system to use FQDN resolution for NTP servers by configuring the **add-server** parameter in the **ntp-config** with an FQDN.

The SBC includes DNS configuration on **network-interface** elements to provide resolution services for any specific realm. For NTP, you can specify the realm you want to use to access DNS services within the **ntp-config**. The system can then use the **network-interface** configuration associated with that realm to make the DNS queries.

Other elementary **ntp-config** configuration detail includes:

- You cannot configure the **add-server** parameter with both IP addresses and an FQDN.
- You cannot configure **add-server** parameter with multiple FQDNs.
- A change to a **network-interface** always requires a reboot for the change to take effect. A change to the **ntp-config**, which impacts the **network-interface**, also requires a reboot for changes to take effect.

When configured with an FQDN, the SBC:

1. Triggers the time synchronization process either after a reboot or the system's periodic NTP daemon restart.

> **Note:**
>
> This is also true when configured with an IP address.

2. Issues a DNS request out the configured realm. This DNS SRV query uses the _ntp._udp prefix to specify the resolution type.
3. Receives the SRV response from the DNS server, which includes the associated A records of IP addresses, and may or may not include priority.
4. Provides its NTP client with the addresses it receives, either ordered by priority or in the same sequence as the DNS response.
5. Issues an NTP synchronization request to the NTP server(s).
6. Receives the NTP response.
7. Synchronizes time.

Important operational detail includes the ability of the SBC to:

- Retry NTP server resolution after periodic intervals if the SRV FQDN lookup resolution fails.
- Retrieve TTL timing for each NTP resolution from the DNS response and retry this connection if and when this timer expires.
- Update the new IP List if there are any IP changes in the DNS Response.
- Apply priority provided within the DNS Response to decide the order of IP addresses it attempts to contact.
- Contact IP addresses using the sequential order presented in the DNS records if there is no priority provided.
- When a user configures NTP with an FQDN within an HA deployment, the active SBC resolves it and synchronizes the resolved IP list with the standby through NTP redundancy. After it receives the resolved IP list from the active, the standby SBC performs NTP update synchronization with the timer servers independently.

Important configuration detail includes:

- You must configure the **dns-ip-primary**, **dns-ip-backup1** and **dns-domain** parameters on the realm's **network-interface**,
- You must configure the **DNS-realm** parameter when configured for FQDN in your **ntp-config**. This realm object must be attached to the **network-interface** with your DNS server configuration, which must be attached to the applicable **phy-interface**.
- If you want to use a media interface's realm for NTP SRV FQDN Resolution, you must configure that **network-interface** for DNS, and you must configure the **ntp-config** with that realm name.
- If you want the NTP SRV FQDN resolution to use wancom0, additional configuration detail includes:
  - If you want to reach DNS servers in the same subnet range as the wancom0 address, you must configure the **phy-interface** name to begin with the “wancom0” prefix and set the **operation-type** to **maintenance**.
    For example, the name "wancom0ntp" would be correct.
  - You must create and attach a wancom0 **network-interface** to a wancom0 **phy-interface**.
  - You must configure your wancom0 **network-interface** with the same IP addressing as your boot parameters and include DNS server configuration.

Configuration

You configure this functionality using the **add-server** parameter within the **ntp-config**. Required configuration includes setting the **add-server** parameter to a text name and the **realm-id** to the realm you want to use for DNS resolution.

```
ORACLE(configuration)#ntp-sync
ORACLE(ntp-config)#add-server example.ntp.com
ORACLE(ntp-config)#realm-id wancom0realm
```

You may find it useful to create a realm specifically for this NTP FQDN resolution. Realms exclusively for NTP resolution are supported over both wancom0 or media interfaces. The following steps apply to creating an NTP resolution specific realm over wancom0.

1. Create a new **physical-interface** using the text "wancom" as the prefix to its name, and set its **operation-type** type to **maintenance**.
2. Create a **network-interface** for this **physical-interface**.
   - Configure the **network-interface** with your DNS Server configuration.
   - Configure the **network-interface** with the same IP addressing values that you use within your boot parameters.
3. Create a **realm-config** and attach it to this **network-interface**.

Resolution Process

Regardless of the interface you use to perform FQDN resolution for your NTP servers, the SBC performs the same DNS procedures to get and use the resolutions.

The SBC uses your configuration to reach DNS servers sequentially. The SBC extracts server information from the first successful DNS response and drops any subsequent responses. Information extracted for NTP purposes includes:

- IP address(es) of NTP servers—One or more addresses, based on the responding server's data.
- Priority—Each IP address can include a priority, which the SBC uses to establish a connection attempt order. The SBC uses the sequence of the resolutions in the DNS response when addresses have the same or no priority.
- Calculated minimum TTL—Each IP address includes a time to live value.

The SBC establishes the minimum value of the timer and starts it. When the timer expires, the SBC sends a new SRV-query to refresh its NTP server list. When it receives the response, the SBC stores the DNS results and rebuilds the NTP list, sorted based on priority or response sequence.

The SBC behaviors above are dependent on the DNS response:

- Single IP address received—Priority is irrelevant and the SBC simply delivers the received address to the NTP daemon.
- Multiple IP addresses received—The lowest priority value is the highest priority server. For addresses presented with the same priority, the SBC uses the DNS server list's order as the order to attempt contact with servers.
- Error/No Response—If the SBC receives an error response or no response to the SRV-query, it starts an internal DNS retry timer before it attempts to contact the servers. Also, if

it finds the primary DNS Server is down, the SBC retries using your configured backup DNS Servers.

- TTL below 30 secs—If the SBC receives TTL that is less than 30 secs for any IP address, it uses 30 seconds as the TTL. This ensures that the system does not become overloaded by an incorrect configuration.

Configuring NTP Using an FQDN - Wancom

These instructions include the specific steps that apply to configuring a wancom interface as the source for synchronizing system time with an NTP server.

Although this is an ACLI procedure, you can perform this procedure using equivalent procedures with supported management interfaces.

You must have enabled the **sip-config**.

1. Configure an applicable **phy-interface**.

   Configure your **phy-interface**, **name** using the text “wancom” as its prefix. If not, the system throws a **verify-config** error.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# phy-interface
   ORACLE(phy-interface)# name wancom_ntp
   ORACLE(phy-interface)# operation-type Maintenance
   ```

   Retain the defaults for all other parameters.

2. Configure an applicable **network-interface**.

   > **Note:**
   >
   > In a normal network interface setup, the pri-utility-addr and sec-utility-addr parameters are configured. However, for a wancom interface, you must leave these parameters unconfigured.

   Create a **network-interface** for your **phy-interface**. Configure that interface with the same **ip-address**, **netmask** and **gateway** used in your system's equivalent boot parameters. The DNS Server IP’s/IP and domain name must reachable from this network.

   ```
   ORACLE(system)# network-interface
   ORACLE(network-interface)# name wancom_ntp
   ORACLE(network-interface)# ip-address 10.196.179.2
   ORACLE(network-interface)# netmask 255.255.128.0
   ORACLE(network-interface)# gateway 10.196.128.1
   ORACLE(network-interface)# dns-ip-primary 10.196.177.83
   ORACLE(network-interface)# dns-domain ntp.com
   ```

   > **Note:**
   >
   > If your **network-interface** values are not the same as your system's boot parameters, you lose SSH connectivity.

3. Configure an applicable **realm**.

   Create a wancom **realm** and attach the **network-interface**.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# identifier wancom_realm
ORACLE(realm-config)# network-interfaces wancom_ntp:0.4
```

4. Create an NTP Configuration.

   Create an **ntp-config** using an FQDN and attach the realm.

```
ORACLE(configure)# ntp-sync
ORACLE(ntp-config)# add-server example.ntp.com
ORACLE(ntp-config)# dns-realm wancom_realm
```

Configuring NTP Using an FQDN - Media Interfaces

These instructions include the specific steps that apply to configuring a media interface as the source for synchronizing system time with an NTP server.

You can use an existing realm if you have configured its **network-interface** with DNS parameters. Follow these steps to create a media realm dedicated to NTP services.

You must have enabled the **sip-config**.

1. Configure an applicable **phy-interface**.

   Leave the undocumented parameters at their defaults.

```
ORACLE(configure)#system-config
ORACLE(system-config)#phy-interface
ORACLE(phy-interface)#name M00
ORACLE(phy-interface)#operation-type Media
```

2. Configure an applicable **network-interface** and attach it to your **phy-interface**.

   Create a **network-interface** for your **phy-interface**. The DNS Server IP's/IP and domain name must reachable from this media interface subnet.

```
ORACLE(configure)#system-config
ORACLE(system-config)#network-interface
ORACLE(network-interface)#name M00
ORACLE(network-interface)#sub-port-id 33.4
ORACLE(network-interface)#ip-address 192.168.203.10
ORACLE(network-interface)#netmask 255.255.0.0
ORACLE(network-interface)#dns-ip-primary 192.168.203.1
ORACLE(network-interface)#dns-domain ntp.com
```

3. Configure your realm.

   Create a **realm-config** and attach the **network-interface**.

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

```
ORACLE(realm-config)#identifier ntp_access
ORACLE(realm-config)#network-interfaces M00:33.4
```

> **Note:**
>
> If you configure an FQDN, such as example.ntp.com, from the **add-server** parameter, the system adds the prefix _ntp._udp.example.ntp.com to the DNS request. You must also ensure that the DNS database includes the _ntp_udp prefix.

Run the command below to verify access to DNS services.

```
ORACLE# show dns query access SRV _ntp._udp.example.ntp.com
DNS Result:
Query Name -->SRV:_ntp._udp.example.ntp.com
Answers -->10.196.177.83: 5060/UDP Hl= 100
```

4. Configure an NTP configuration.

   Create an **ntp-config** using an FQDN and attach the realm.

```
ORACLE# configure terminal
ORACLE(configure)# ntp-sync
ORACLE(ntp-config)# add-server example.ntp.com
ORACLE(ntp-config)# dns-realm ntp_access
```

   Run the commands below to verify NTP synchronization and access to the DNS server the SBC selected.

```
ORACLE# show ntp status
NTP synchronized to server at: 10.196.177.83

ORACLE# show ntp server
NTP Status Tue Apr 19 10:31:34 GMT 2022MS server st poll reach LastRx
LastSample
        <LastOffset>[<ActualOffset>]+/-<Error>--
^ti 10.196.177.83 4 2 377 4 -95us[ -109us] +/- 67ms
^- 10.196.177.181 4 2 377 4 -95us[ -109us] +/- 67ms +85us[ +71us] +/- 93ms
```

Authenticated NTP

The Oracle Communications Session Border Controller can authenticate NTP server requests using MD5. The configured MD5 keys are encrypted and obscured in the ACLI. You configure an authenticated NTP server with its IP address, authentication key, and the key ID. Corresponding key and key IDs are provided by the NTP server administrator.

To configure an authenticated NTP server:

1. Access the ntp-config configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# ntp-sync
ACMEPACKET(ntp-config)#
```

2. Type select.

```
ORACLE(ntp-config)# select
```

3. Access the auth-servers configuration element

```
ORACLE(ntp-config)# auth-servers
ORACLE(auth-servers)#
```

4. ip-address — Enter the IPv4 or IPv6 address of the NTP server that supports authentication.
5. key-id — Enter the key ID of the key you enter in the next step. This value's range is 1 - 999999999.
6. key — Enter the key used to secure the NTP requests. The key is a string 1 - 31 characters in length.
7. Type **done** to save your work.
8. Type **exit** to return to the previous configuration level.
9. Type **done** to save the parent configuration element.

Monitoring NTP from the ACLI

NTP server information that you can view with the **show ntp server** command tell you about the quality of the time being used in terms of offset and delays measurements. You can also see the maximum error bounds.

When you use this command, information for all configured servers is displayed. Data appears in columns that are defined in the table below:

Display Column	Definition
server	Lists the NTP servers configured on the Oracle Communications Session Border Controller by IP address. Entries are accompanied by characters: Plus sign (+)—Symmetric active server Dash (-)—Symmetric passive server Equal sign (=)—Remote server being polled in client mode Caret (^)—Server is broadcasting to this address Tilde (~)—Remote peer is sending broadcast to * Asterisk (*)—The peer to which the server is synchronizing
st	Stratum level—Calculated from the number of computers in the NTP hierarchy to the time reference. The time reference has a fixed value of 0, and all subsequent computers in the hierarchy are n+1.
poll	Maximum interval between successive polling messages sent to the remote host, measured in seconds.

Display Column	Definition
reach	Measurement of successful queries to this server; the value is an 8-bit shift register. A new server starts at 0, and its reach augments for every successful query by shifting one in from the right: 0, 1, 3, 7, 17, 37, 77, 177, 377. A value of 377 means that there have been eight successful queries.
delay	Amount of time a reply packet takes to return to the server (in milliseconds) in response.
offset	Time difference (in milliseconds) between the client's clock and the server's.
disp	Difference between two offset samples; error-bound estimate for measuring service quality.

View Statistics

To view statistics for NTP servers:

- At the command line, type **show ntp server** and press Enter.

```
ORACLE# show ntp server
NTP Status                                    FRI APR 11:09:50 UTC 2007
 server                 st  poll  reach   delay    offset     disp
---------------------- --  ----  ------  -------  --------   ---------
*64.46.24.66             3    64     377  0.00018  0.000329   0.00255
=61.26.45.88             3    64     377  0.00017  0.002122   0.00342
```

You can the see the status of NTP on your system by using the **show ntp status** command. Depending on the status of NTP on your system, one of the following messages will appear:

- NTP not configured
- NTP Daemon synchronized to server at [the IP address of the specific server]
- NTP synchronization in process
- NTP down, all configured servers are unreachable

View Status

To view the status of NTP on your Oracle Communications Session Border Controller:

- At the command line, type **show ntp status** and press Enter.

```
ORACLE# show ntp status
```

HTTP Connection Management

By default, the SBC limits system impact caused by HTTP client behavior using the **httpclient-max-total-conn** and **httpclient-max-cpu-load** parameters in the **system-config**. These parameters allow you to change the number of TCP connections and the amount of CPU resources consumed by traffic between the SBC and all types of HTTP servers.

Use the following **system-config** parameters to adjust or disable management of the number of active HTTP clients by the SBC:

- **httpclient-max-total-conn**—Specifies the maximum number of TCP connections that the **http-client** allows open simultaneously.
  When this traffic exceeds this value, the SBC and the **http-client** begin to discard new http/https requests. When used TCP connections falls below this value, the SBC resumes accepting HTTP client TCP connections.

  Valid Values:

  – 0—Disables the function
  – Range—0 - 2147483647
  – Default—500

  You cannot configure**http-client** in real time. You must reboot the system whenever you make a change.
- **httpclient-max-cpu-load**—Specifies the maximum percentage of CPU consumed by HTTP traffic during STIR/SHAKEN operations.
  When CPU resource utilization exceeds this value, the SBC, the **http-client** begins to discard new http/https requests. When CPU utilization falls below this value, the SBC resumes accepting this traffic.

  Valid Values:

  – Range—30% - 90%
  – Default—70%

  You can configure **httpclient-max-cpu-load** in real time.
- **httpclient-cache-size-multiplier**—Specifies the multiplier used to calculate the size of the HTTP client connection cache.
  The system maintains an HTTP connection cache pool. This is a collection of previously used connections that the system keeps alive, instead of closing after use, so that subsequent transfers targeting the same host name can use them instead of creating a new connection. The size of the HTTP connection cache is based on the number of these live connections.

  The system calculates this cache size using the formula:

  client connection cache = (number of pending transactions * httpclient-cache-size-multiplier)

  Valid Values:

  – Default: 16
  – Values: 4 - 50

  You cannot configure the **http-client** in real time. You must reboot the system whenever you make a change.
- **http-clearDead-conn-timer**—Specifies the number of seconds the system waits before it closes connections to HTTP servers that are in the half closed state.
  The system maintains connections between itself and HTTP servers that is equal to the number of connections in the cache pool multiplied by the **httpclient-cache-size-multiplier** value when, for example, STIR traffic stops. The system puts these connections in half-closed state after the STIR server tries to close the connections using FIN messages. By default, these connections remain in half closed state indefinitely until STIR traffic starts again. You can set the **http-clearDead-conn-timer** parameter to a timer value, after which the system clears these connections regardless of STIR traffic status. The value specifies the regular interval the system uses to clear these half-closed STIR server connections.

  Valid Values:

- Default: 0 (disabled)
- Values: 300 - 86400

You use the **Http/s calls Dropped** field in the **show sipd errors** command to monitor dropped HTTP traffic.

Configure HTTP Connection Management

To prevent system issues caused by HTTP client traffic:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   If you are adding support for this feature to a preexisting configuration, you must select (using the ACLI **select** command) the single instance **system-config** element.

4. **httpclient-max-total-conn**—Set this parameter to specify the maximum number of TCP connections that the **http-client** allows open simultaneously or disable the function. You cannot configure **httpclient-max-total-conn** in real time. You must reboot the system whenever you make a change.

   Valid Values:

   - 0—Disables the function
   - Range—0 - 2147483647
   - Default—500

   ```
   ORACLE(system-config)# httpclient-max-total-conn 1000
   ```

5. **httpclient-max-cpu-load**—Set this parameter to specify the maximum percentage of CPU consumed by HTTP traffic during STIR/SHAKEN operations. You cannot configure **httpclient-max-cpu-load** in real time.

   Valid Values:

   - Range—30% - 90%
   - Default—70%

   ```
   ORACLE(system-config)# httpclient-max-cpu-load 60
   ```

6. **httpclient-cache-size-multiplier**—Specifies the multiplier used to calculate the size of the HTTP client connection cache. The system maintains an HTTP connection cache pool. The size of the HTTP connection cache is the number of pending transactions, multiplied by this **httpclient-cache-size-multiplier**.

Valid Values:

- Default: 16
- Values: 4 - 50

```
ORACLE(system-config)# httpclient-cache-size-multiplier 20
```

7. **http-clearDead-conn-timer**—Specifies the number of seconds the system waits before it closes connections to HTTP servers that are in the half closed state.

   Valid Values:

   - Default: 0 (disabled
   - Values: 300 - 84600

```
ORACLE(system-config)# http-clearDead-conn-timer 1000
```

Telephony Fraud Protection

You can use the Oracle Communications Session Border Controller (SBC) to protect against fraudulent calls by enabling Telephony Fraud Protection and creating lists of phone numbers to block, allow, redirect, and rate limit calls. The lists reside together in a single source-file that you create and manage. The source-file can contain any combination of the list types and it can reside on either the SBC or in Session Delivery Manager (SDM) because you can manage Telephony Fraud Protection from either one. The following information explains using Telephony Fraud Protection on the SBC. See the *Oracle Communications Session Element Manager User Guide for the Enterprise Edge and Core Plug-in* for managing Telephony Fraud Protection from SDM.

> **Note:**
>
> The Session Router does not support Telephony Fraud Protection.

Fraud Protection List Types and Uses

The SBC supports the following types of lists for protecting against fraudulent calls.

Blocklist—Use the blocklist to specify a fraudulent call based on the destination phone number or URI. You can add a known fraudulent destination to the blocklist by prefix or by fixed number. When the SBC receives a call to an entry on the blocklist, the system rejects the call according to the SIP response code that you specify. When the system determines a match and blocks a call, the default response is "403 Forbidden." You can set another SIP response code from the standard list of responses defined in RFC3261 by way of the **Local Response Map** configuration and the local error **Fraud Protection Reject Call** setting.

Allowlist—Use the allowlist to manage any exception to the blocklist. Suppose you choose to block a prefix such as +49 555 123 by way of the blocklist. This action also blocks calls to individual numbers starting with this prefix, such as +49 555 123 666. If you add a prefix or individual number to the allowlist, the system allows calls to the specified prefix and number. Continuing with the example, if you add +49 555 123 6 to the allowlist, the system allows calls to +49 555 123 666, which was blocked by the blocklist entry of +49 555 123.

Redirect List—Use the redirect list to send a fraudulent call to an Interactive Voice Response (IVR) system, or to a different route. For example, you can intercept and redirect a call going to

a revenue-share fraud target in a foreign country to an end point that defeats the fraud. Or, you might want to redirect subscribers dialing a particular number and URI to an announcement to make them aware that an account is compromised and tell them what they should do.

Rate Limit List—Use rate limiting to limit the loss of money, performance, and availability that an attack might cause. While local ordinances may not allow you to completely block or suppress communication, you may want to reduce the impact of a disruption with rate limiting until a network engineer can analyze an attack and plan remediation. For example, you might want time to find the origin of an attack or to add attackers to a blocklist. Note that rate limiting may not function immediately after a High Availability switch over because the newly active system must re-calculate the call rate before it can apply rate limiting.

Configuration

The process for using Telephony Fraud Protection includes the following steps:

1. Enable Telephony Fraud Protection
2. Specify the source of fraud protection management
3. Create the file that contains the list of phone numbers to manage
4. Activate the fraud protection file

See "Configure Telephony Fraud Protection", "Fraud Protection XML Source File Example," and "Telephony Fraud Protection File Activation."

> **Note:**
>
> See the following topics in the Release Notes for important information about "Fraud Protection File Rollback Compatibility" and "Fraud Protection File Upgrade Compatibility."

You can enable and manage Telephony Fraud Protection from the ACLI command line.

Telephony Fraud Protection is included in the base license.

Administration

When you configure the SBC to manage Telephony Fraud Protection, the system applies the following behavior:

- An Administrator with privileges can Refresh, Add, and Upload an unselected file, and Edit, Download, and Delete a selected file.
- An Administrator with no privileges can only view the fraud protection file.

To view fraud protection data:

- From the ACLI, use the show commands to view fraud protection statistics. See "Telephony Fraud Protection Show Commands."

> **Note:**
>
> The Telephony Fraud Protection feature does not affect emergency calls or block any calls while you are loading entries.

High Availability

Telephony Fraud Protection supports High Availability (HA).

- After an HA switch over, use the **synchronize file <filename>** command to copy the fraud protection file to the standby.
- Note that after a switch over, rate limiting may not take effect immediately because the new Active system needs time to recalculate the call rate before it can apply rate limiting.

Whenever you refresh the telephony fraud protection file from the ACLI with the **notify fped refresh** command, this updates the runtime table by reloading the entries in the file specified in the fraud-protection configuration, only for the active SBC. To update the FPE runtime table on the standby SBC , run the **synchronize file <filename>** command on the active SBC and then run the **notify fped refresh** command on the standby SBC.

Telephony Fraud Protection Management from SDM

If you prefer to manage Telephony Fraud Protection from the Enterprise Fraud Manager in SDM, rather than from the SBC, store the fraud protection list in a file named **sbc_fpe_entries.xml** (case sensitive) in SDM. You can edit the file in SDM, which will notify the SBC afterward to download the file to its **/code/fpe** directory. When the SBC is part of an HA pair, the Active partner automatically pushes the updated file to the Standby partner. In the event of an unsuccessful download, the system raises an SNMP alarm. Should the connection to SDM ever go down, the system also raises an SNMP alarm and sends a trap. When the connection gets re-established, the alarm and trap clear, and the SBC sends a RESYNC command to SDM.

Unsupported Functions

Telephony Fraud Protection for the SBC does not support the following:

- IPv6
- H.323
- InterWorking Function (IWF)
- Comm Monitor
- Enterprise Operations Monitor

Telephony Fraud Protection Target Matching Rules

When matching a call to an entry on a telephony fraud protection list, the Oracle Communications Session Border Controller (SBC) performs the matching only on the ingress leg of the initial INVITE. If ingress realm is defined as *, then the realm takes precedence over all other entries. In the initial INVITE, the SBC uses the From, To, and User-Agent headers for matching. Because you can place a phone number on multiple types of fraud prevention lists in the same source file, the SBC uses the following evaluation hierarchy to determine which number takes precedence:

1. Longest match—The most specific entry takes precedence. For example, when 555-123-4000 is block listed and 555-123-* is allow listed, the system blocks the call from 555-123-4000 because it is the longest match.
2. Destination—When the system detects matches in both the SIP **From** header and the SIP **To** header, the match for the **To** header takes precedence.
3. URI—When the system detects matches in both the **USER** and **Host** parts of a SIP URI, the match for the **USER** part takes precedence.

4. SIP User-Agent header—Lowest priority. When nothing else matches, and there is a match for the User-Agent field, the SBC acts as instructed.
5. Multiple instances—When the system detects multiple instances of the same match length, or when the target resides in multiple lists, the system uses the following order of precedence:
   1. Allowlist—Entries on the allowlist take precedence with no restrictions. For example, when 555-123-4567 is on both the blocklist and the allowlist, the system allows this call because the number is on the allowlist.
   2. Blocklist
   3. Redirect
   4. Rate limiting

> **Note:**
>
> The telephony fraud protection feature does not affect emergency calls.

The telephony fraud protection feature uses source or destination IP, source or destination name or phone number, and caller user-agent to identify a caller. The system enforces the following rules for formatting entries on a fraud protection list:

Hostname

Format: Enter the exact IP address or FQDN.

User name

Format: Enter the exact user name. For example: joe.user or joe_user.

User-Agent-Header

The User-Agent header text in the INVITE message from the first call leg. This text usually contains the brand and firmware version of the SIP device making the call. For example, sipcli/v1.8, Asterisk PBX 1.6.026-FONCORE-r78.

Format: Enter the exact text.

Phone Number

Format: Enter the exact number or a partial number using the following characters to increase the scope of the matches.

Asterisk *	Use to indicate prefix matching, but only at the end of the pattern. For example, use 555* not *555. Do not use * in any other patterns, for example, in brackets [ ], parentheses ( ), or with an x.
Square Brackets [ ]	Use to enclose ranges in a pattern. Syntax: [min-max]. For example: 555 [0000-9999]. The system considers 8[1-20]9 and 8[01-20]9 to contain the same number of characters because the leading 0 is implied. The system strictly enforces this pattern with respect to the range and the number of characters, as follows: • 8019 matches

	• 819 does not match • 8119 matches
Character x	Use as a wild card a the end of a dial pattern to mean 0-9. For example: 555xxx means match a number starting with 555 followed by 3 digits from 0-9.
Parentheses ( )	Use to enclose optional digits in a pattern. For example: 555xx(xxxx) means match a number starting with 555 plus a minimum of 2 digits, and optionally up to 4 more digits.

Telephony Fraud Protection File Activation

After you create, edit, or upload the telephony fraud protection file, you must activate the file before the Oracle Communications Session Border Controller (SBC) can use it as the source of the fraud protection lists. The system recognizes only one file at a time as the active file.

The first time you configure the SBC to manage fraud protection, the system activates the file when you save and activate the configuration. After the initial configuration, the system does not automatically refresh the fraud protection file when you save and activate other configuration changes on the SBC. You must upload a new file or edit the existing file and activate it to update the file. The exception occurs when you specify a new file name in the fraud protection configuration and coincidentally make changes to other configurations, and then save and activate all of the changes at the same time.

After the initial configuration, use the following methods to activate the fraud protection file.

- New File—After you create or upload a new file, go to Fraud Protection configuration, enter the name of the new file, and click Save. The system prompts for activation upon a successful Save. Note that you can decline the inline activation and manually activate the file later. For example, you might want to edit an uploaded file before activation.
- Overwrite File—When you upload a file with the same name as the existing file, the system prompts for activation upon upload.
- Edit File—When you upload the edited file, the system prompts for activation.
- Refresh File—When you want to use the ACLI to refresh the fraud protection file, send the file to the SBC and use the `notify fped refresh` command. The name of the file that you refresh must match the name of the file specified in the configuration.

Telephony Fraud Protection Data Types and Formats

Use the information in the following tables when you create or edit a fraud protection list in the Add Fraud Protection Entry and Modify Fraud Protection Entry dialogs.

Data Type Descriptions

The following table describes the data types listed in the **Type** drop-down list.

from-hostname	The hostname from the SIP FROM header.
from-phone-number	The phone number from the SIP FROM header
from-username	The user name from the SIP FROM header.
to-hostname	The hostname from the SIP TO header.

to-phone-number	The phone number from the SIP TO header.
to-username	The user name from the SIP TO header.
user-agent-header	The SIP User-Agent header.

Match Value Formats

The following table describes the formats required for the data types.

hostname	Enter the exact IP address or FQDN.
username	Enter the exact user name. For example: joe.user or joe_user.
user-agent-header	Enter the exact text match to the SIP User-Agent header. For example: equipment vendor information.
phone-number	You can use the following characters for phone-number: • Asterisk *. Use to indicate prefix matching, but only at the end of the pattern. For example, use 555* not *555. Do not use * in any other patterns, for example, in brackets [ ], parentheses ( ), or with an x. • Brackets [ ]. Use to enclose ranges in a pattern. Syntax: [min-max]. For example: 555 [0000-9999]. • Parentheses. ( ) Use to enclose optional digits in a pattern. For example: 555xx(xxxx) means 555 with between 2 and 4 following digits. • Character x. Use as a wildcard a the end of a dial pattern to mean 0-9. For example: 555xxx means a number starting with 555 followed by 3 digits.

Caution:

The use of encoding characters is especially susceptible to creating overlapping dial pattern matches that can result in unexpected behavior.

Configure Telephony Fraud Protection

The telephony fraud protection feature requires configuration, which you can perform from the Oracle Communications Session Border Controller (SBC) ACLI by way of the **fraud-protection** configuration element under System.

- Add or upload at least one telephony fraud protection file to the SBC.
- Note the name of the fraud protection file that you want to use.

Use this procedure to enable telephony fraud protection on the SBC. You must specify the fraud protection file name and activate the configuration. You cannot specify multiple fraud protection files because the system recognizes only one file as the active source file.

1. Access the **fraud-protection** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(session-router)# fraud-protection
```

2. Type **select**, and press ENTER.
3. Type **show**, and press ENTER.
4. Do the following:

mode	Select one of the following modes: • **local**—Use the SBC as the source of the fraud protection file. • **comm-monitor**—Not currently supported. • **disabled**—Default.
file name	Enter the name of the fraud protection file. Syntax: /code/fpe/<filename>
options	Add fraud protection options. (Not supported in some releases.)
allow-remote-call-terminate	Not currently supported.

5. Save and activate the configuration.

Refresh the Telephony Fraud Protection File

You can refresh the telephony fraud protection file from the ACLI with the **notify fped refresh** command. This command updates the runtime table by reloading the entries in the file specified in the fraud-protection configuration.

- SFTP the updated file to the SBC.
- Confirm that the name of the updated file matches the name of the file specified in the configuration.

Use the following procedure apply updates to the telephony fraud protection file.

1. Log on to the ACLI.
2. Type **notify fped refresh**, and press ENTER.

   The system confirms a successful refresh.

Telephony Fraud Protection ACLI Show Commands

The Oracle Communications Session Border Controller (SBC) supports viewing and refreshing telephony fraud protection statistics by way of ACLI commands. The displayed data is read-only.

The following ACLI commands provide displays of telephony fraud protection statistics.

`show-fraud-protection <list type> <matches-only>`—Use this command to display all entries or only entries on a particular fraud prevention list, and optionally, to show only the entries on the specified list that incurred a match. Use one of the following variables for <list type>:

- all—displays all entries
- blocklist—displays only the blocklist matches
- allowlist—displays only the allowlist matches
- redirect—displays only the redirect matches

- ratelimit—displays only the rate limit matches

Command Examples:

- `show-fraud-protection all`—displays all blocklist, redirect, allowlist, and rate limit entries.
- `show-fraud-protection all matches-only`—displays only the matches for blocklist, redirect, allowlist, and rate limit entries.
- `show-fraud-protection blocklist`—displays only the blocklist, showing all entries.
- `show-fraud-protection blocklist matches-only`—displays only the matches for blocklist entries.

Display Examples

```
show fraud-protection all
17:31:09-109
-------------------------------------------------------------------------------------------------------------------
List Type   To/From     Match Value                          Ingress Realm                No. of Hits
                                                                                  Recent      Total  PerMax
-------------------------------------------------------------------------------------------------------------------
BLOCKLIST   To          1809*                                peer                      0          0       0
BLOCKLIST   To          22478300501                          access                    0          0       0
BLOCKLIST   From        172.38.10.0/24                       enterpriseco              0          0       0
BLOCKLIST   From        192.168.39.0/24                      peer                      0          0       0
BLOCKLIST   From        roberto.veras                        peer                      0          0       0
RATE_LIMIT  To          20.20.20.20                          boston.com                0          0       0
RATE_LIMIT  From        18092059090                          peer                      0          0       0
RATE_LIMIT  From        peter.paker                          nyrealm                   0          0       0
REDIRECT    To          john.doe                             domain                    0          0       0
REDIRECT    From        10.10.10.10                          nyrealm                   0          0       0
REDIRECT    From        18092059080                          peer                      0          0       0
ALLOWLIST   To          1978973[0000-9999]                   peer                      0          0       0
ALLOWLIST   To          22478300501                          access                    0          0       0
ALLOWLIST   From        172.38.10.0/24                       service_provider          0          0       0

                    Total hits: 0
                 Total entries: 14
      Total displayed  entries: 14
                     File name: my_entries.xml
         Last file upload time: 2015-07-22 17:28:08
```

BLOCKLIST

`show-fraud-protection`—Use to display all entries with matches-only

`show fraud-protection stats`—Use to display Recent, Total, and Period Maximum statistics for the fraud protection lists: For example: STATS

```
show fraud-protection stats


Fraud Protection Engine Stats       ---- Lifetime ----
                                  Recent      Total  PerMax
Blocklisted Calls                      0          0       0
Allowlisted Calls                      0          0       0
RateLimited Calls                      0          0       0
Redirected Calls                       0          0       0
Blocklist Rejected Calls               0          0       0
RateLimit Rejected Calls               0          0       0
```

The following ACLI commands refresh displays of fraud protection entries.

`notify fped refresh`—Use to update the fraud protection lists table after you make changes. If for some reason the refresh command is unsuccessful and cannot update the list with new data, the system preserves the existing data.

`notify fped reset-stats`—Use to reset the fraud protection statistics counter to zero, for example, to begin a new data collection period.

Telephony Fraud Protection verify-config

When you run the verify-config command for Telephony Fraud Protection, the system verifies the following:

- When you set the Fraud Protection mode to local, verify-config confirms that the file specified in the Fraud Protection configuration exists in the /code/fpe directory. If the specified file is not compatible with the release version or is otherwise invalid, the system displays an error message.

Fraud Protection XML Source File Example

When you enable the Oracle Communications Session Border Controller (SBC) to protect against fraudulent calls, you must create lists of phone numbers or IP addresses to block, allow, redirect, and rate limit calls. The lists reside together in a single file that you specify as the source file in the fraud protection configuration. The source file can contain any combination of the list types, but the system limits the size of the fraud protection file to 100,000 total entries.

The following example shows an XML file created as the source file for fraud protection. The file includes a section for each type of list, which includes call-blocklist, call-allow list, call-limit, and call-redirect. The example shows the coding for adding the source-ip, destination-phone, realm, calls-per-second for rate limiting, and the target for a redirected call.

```
<?xml version='1.0' standalone='yes'?>
<oracleSbcFraudProtectionApi version="1.0">
    <call-allowlist>
         <userEntry>
             <to-phone-number>1234567[0000-9999]</to-phone-number>
             <realm>Core</realm>
         </userEntry>
         <userEntry>
             <to-phone-number>12345678901</to-phone-number>
             <realm>DefaultSP</realm>
           </userEntry>
           <userEntry>
               <from-hostname>123.45.67.8/90</from-hostname>
               <realm>*</realm>
           </userEntry>
    </call-allowlist>
    <call-blocklist>
         <userEntry>
             <to-hostname>12345678901</to-hostname>
             <realm>*</realm>
         </userEntry>
         <userEntry>
             <from-hostname>123.45.67.8/90</from-
hostname>
             <realm>*</realm>
         </userEntry>
    </call-blocklist>
    <call-redirect>
         <userEntry>
```

```
            <from-hostname>19877654321</from-hostname>
            <realm>Core</realm>
            <target>sip:support@phonesystem.com</target>
        </userEntry>
    </call-redirect>
    <call-rate-limit>
        <userEntry>
            <from-hostname>19877654321</from-
hostname>
            <realm>DefaultSP</realm>
            <calls-per-second>5</calls-per-second>
            <max-active-calls>0</max-active-calls>
        </userEntry>
    </call-rate-limit>
```

Note that the SBC enforces an order of precedence among the lists. See "Telephony Fraud Protection Target Matching Rules."

> **Note:**
>
> If you rollback to a previous version of the software, you must edit this file by changing Allowlist to Whitelist and Blocklist to Blacklist. When you return to the SCz9.1.0 or higher version of the software, revert to Allowlist and Blocklist.

4

Realms and Nested Realms

This chapter explains how to configure realms and nested realms, and specialized media-related features.

A realm is a logical definition of a network or groups of networks made up in part by devices that provide real-time communication sessions comprised of signaling messages and possibly media flows. These network devices might be call agents, softswitches, SIP proxies, H.323 gatekeepers, IP PBXs, etc., that are statically defined by IPv4 addresses. These network devices might also be IPv4 endpoints: SIP phones, IADs, MAs, media gateways, etc., that are defined by an IPv4 address prefix.

Realms support bandwidth-based call admission control and QoS marking for media. They are the basis for defining egress and ingress traffic to the Oracle Communications Session Border Controller—which supports the Oracle Communications Session Border Controller's topology hiding capabilities.

This chapter also explains how to configure media ports (steering pools). A steering pool exists within a realm and contains a range of ports that have a common address (for example, a target IPv4 address). The range of ports contained in the steering pool are used to steer media flows from one realm, through the Oracle Communications Session Border Controller, to another.

Finally, in this chapter you can learn about TOS/DiffServ functionality for realm-based packet marking by media type.

Overview

Realms are a logical distinction representing routes (or groups of routes) reachable by the Oracle Communications Session Border Controller and what kinds of resources and special functions apply to those routes. Realms are used as a basis for determining ingress and egress associations to network interfaces, which can reside in different VPNs. The ingress realm is determined by the signaling interface on which traffic arrives. The egress realm is determined by the following:

- Routing policy—Where the egress realm is determined in the session agent configuration or external address of a SIP-NAT
- Realm-bridging—As applied in the SIP-NAT configuration and H.323 stack configurations
- Third-party routing/redirect (i.e., SIP redirect or H.323 LCF)

Realms also provide configuration support for denial of service (DoS)/access control list (ACL) functionality.

Realms can also be nested in order to form nested realm groups. Nested realms consist of separate realms that are arranged within a hierarchy to support network architectures that have separate backbone networks and VPNs for signaling and media. This chapter provides detailed information about nested realms after showing you how to configure realms on your Oracle Communications Session Border Controller.

About Realms and Network Interfaces

All realms reference network interfaces on the Oracle Communications Session Border Controller. This reference is made when you configure a list of network interfaces in the realm configuration.

You configure a network interface to specify logical network interfaces that correspond existing phy-interfaces on the Oracle Communications Session Border Controller. Configuring multiple network interfaces on a single phy-interface creates a channelized phy-interface, a VLAN. VLANs, in turn, allow you to reuse address space, segment traffic, and maximize bandwidth.

In order to reach the realms you configure, you need to assign them network interfaces. The values you set for the name and port in the network interface you select then indicate where the realm can be reached.

About the SIP Home Realm

The realm configuration is also used to establish what is referred to as the SIP home realm. This is the realm where the Oracle Communications Session Border Controller's SIP proxy sits.

In peering configurations, the SIP home realm is the internal network of the SIP proxy. In backbone access configurations, the SIP home realm typically interfaces with the backbone connected network. In additions, the SIP home realm is usually exposed to the Internet in an HNT configuration.

Although you configure a SIP home realm in the realm configuration, it is specified as the home realm in the main SIP configuration by the home realm identifier parameter. Specifying the SIP home realm means that the Oracle Communications Session Border Controller's SIP proxy can be addressed directly by connected entities, but other connected network signaling receives layer 3 NAT treatment before reaching the internal SIP proxy.

About Realms and Other Functions

Realms are referenced by other configurations in order to support this functionality across the protocols the Oracle Communications Session Border Controller supports and to make routing decisions. Other configurations' parameters that point to realms are:

- SIP configuration: home realm identifier, egress realm identifier
- SIP-NAT configuration: realm identifier
- H.323 stack configuration: realm identifier
- Session agent configuration: realm identifier
- Media manager: home realm identifier
- Steering ports: realm identifier
- Static flow: in realm identifier, out realm identifier

Realms

Realm configuration is divided into the following functional areas, and the steps for configuring each are set out in this chapter: identity and IP address prefix, realm interfaces, realm service profiles, QoS measurement, QoS marking, address translation profiles, and DNS server configuration.

Before You Configure

Before you configure realms, you want to establish the phy and network interfaces with which the realm will be associated.

- Configure a phy-interface to define the physical characteristics of the signaling line.
- Configure a network-interface to define the network in which this realm is participating and optionally to create VLANs.

If you wish to use QoS, you should also determine if your Oracle Communications Session Border Controller is QoS enabled.

Remember that you will also use this realm in other configurations to accomplish the following:

- Set a signaling port or ports at which the Oracle Communications Session Border Controller listens for signaling messages.
- Configure sessions agents to point to ingress and egress signaling devices located in this realm in order to apply constraint for admission control.
- Configure session agents for defining trusted sources for accepting signaling messages.

Configure realm-config

To access the realm configuration parameters in the ACLI:

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

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

2. Press **Enter**.

   The system prompt changes to let you know that you can begin configuring individual parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

Identity and IP Address Prefix

The first parameters you configure for a realm are its name (a unique identifier) and an IP address prefix and subnet mask.

The IP address and subnet mask establish a set of matching criteria for the realm, and distinguishes between realms that you assign to the same network interface.

To configure a realm's identity and IP address prefix in the ACLI:

1. **identifier**—Enter the name of the realm. This parameter uniquely identifies the realm. You will use this parameter in other configurations when asked for a realm identifier value.
2. **addr-prefix**—Enter the IPv4 or IPv6 address and subnet mask combination to set the criteria the Oracle Communications Session Border Controller uses to match packets sent or received on the network interface associated with this realm. This matching determines the realm, and subsequently what resources are used for that traffic. This setting determines whether the realm is an IPv4 or IPv6 realm.

This parameter must be entered in the correct format where the IPv4 or IPv6 address comes first and is separated by a slash (/) from the subnet mask value. For example, 172.16.0.0/24.

The default for this parameter is **0.0.0.0**. When you leave this parameter set to the default, all addresses match.

Realm Interfaces

The realm points to one network interface on the Oracle Communications Session Border Controller.

> **Note:**
>
> Only one network-interface can be assigned to a single realm-config object, except for Local multi-homing SCTP deployments.

To assign interfaces to a realm:

- **network-interfaces**—Enter the physical and network interface(s) that you want this realm to reference. These are the network interfaces though which this realm can be reached by ingress traffic, and through which this traffic exits the system as egress traffic.

  Enter the name and port in the correct format where the name of the interface comes first and is separated by a colon (:) from the port number. For example, f10:0.

  The parameters you set for the network interfaces must be unique.

  Enter multiple network interfaces for this list by typing an open parenthesis, entering each field value separated by a Space, typing a closed parenthesis, and then pressing Enter.

  ```
  ORACLE(realm-config)# network-interfaces fe1:0
  ```

  You must explicitly configure a realm's network interface as either IPv4 or IPv6 when the applicable interface is either dual-stack or IPv6. You do this by appending the realm's network-interface with a **.4** or a **.6**, as shown below.

  ```
  ORACLE(realm-config)# network-interfaces fe1:0.6
  ```

  For single-stack interface configurations that do not specify this format, the Oracle Communications Session Border Controller assumes an IPv4 interface. Dual stack interface configurations fail if this IP version family suffix is not specified.

Realm Service Profile

The parameters you configure to establish the realm service profile determine how bandwidth resources are used and how media is treated in relation to the realm. Bandwidth constraints set for realm service profiles support the Oracle Communications Session Border Controller (SBC) admission control feature.

Peer-to-peer media between endpoints can be treated in one of three different ways:

- Media can be directed between sources and destinations within this realm on this specific SBC. Media travels through the SBC rather than straight between the endpoints.

- Media can be directed through the SBC between endpoints that are in different realms, but share the same subnet.
- For SIP only, media can be released between multiple SBCs.
  To enable SIP distributed media release, you must set the appropriate parameter in the realm configuration. You must also set the SIP options parameter to media-release with the appropriate header name and header parameter information. This option defines how the SBC encodes IPv4 address and port information for media streams described by, for example, SDP.

> **Note:**
>
> The **nat-traversal** parameter can establish an important media handling behavior. If you set **nat-traversal** on a **sip-interface** to **always**, this setting supersedes any multi-media configuration that would otherwise release the media. Instead, the SBC recognizes when a flow's leg is behind a NAT during the signaling, and ignores any configuration that would release the media. The SBC then sets up the end to end media flow in MBCD and performs its HNT function for that flow.

To configure realm service profile:

1. **max-bandwidth**—Enter the total bandwidth budget in kilobits per second for all flows to/from the realm defined in this element. The default is **0** which allows for unlimited bandwidth. The valid range is:
   - Minimum—0
   - Maximum—4294967295
2. **mm-in-realm**—Enable this parameter to treat media within this realm on this SBC. The default is **disabled**. Valid values are:
   - enabled | disabled
3. **mm-in-network**—Enable this parameter to treat media within realms that have the same subnet mask on this SBC. The default is **enabled**. Valid values are:
   - enabled | disabled
4. **msm-release**—Enable or disable the inclusion of multi-system (multiple SBCs) media release information in the SIP signaling request sent into the realm identified by this realm-config element. If this field is set to enabled, another SBC is allowed to decode the encoded SIP signaling request message data sent from a SIP endpoint to another SIP endpoint in the same network to restore the original SDP and subsequently allow the media to flow directly between those two SIP endpoints in the same network serviced by multiple SBCs. If this field is disabled, the media and signaling will pass through both SBCs. Remember that for this feature to work, you must also set the options parameter in the SIP configuration accordingly. The default is **disabled**. Valid values are:
   - enabled | disabled

QoS Measurement

This chapter provides detailed information about when to configure the **qos-enable** parameter. If you are not using QoS or a QoS-capable Oracle Communications Session Border Controller, then you can leave this parameter set to disabled (default).

QoS Marking

QoS marking allows you to apply a set of TOS/DiffServ mechanisms that enable you to provide better service for selected networks

You can configure a realm to perform realm-based packet marking by media type, either audio/voice or video.

The realm configuration references a set of media policies that you configure in the media policy configuration. Within these policies, you can establish TOS/DiffServ values that define an individual type (or class) of service, and then apply them on a per-realm basis. In the media profiles, you can also specify:

- One or more audio media types for SIP and/or H.323
- One or more video types for SIP and/or H.323
- Both audio and video media types for SIP and/or H.323
  To establish what media policies to use per realm in the ACLI:

- **media-policy**—Enter the name (unique identifier) of the media policy you want to apply in the realm. When the Oracle Communications Session Border Controller first sets up a SIP or H.323 media session, it identifies the egress realm of each flow and then determines the media-policy element to apply to the flow. This parameter must correspond to a valid name entry in a media policy element. If you leave this parameter empty, then QoS marking for media will not be performed for this realm.

Address Translation Profiles

If you are not using this feature, you can leave the **in-translationid** and **out-translationid** parameters blank.

Realm Support of DNS Servers

You can configure DNS functionality on a per-realm basis. Configuring DNS servers for your realms means that you can have multiple DNS servers in connected networks. In addition, this allows you to specify which DNS server to use for a given realm because the DNS might actually be in a different realm with a different network interface.

This feature is available for SIP only.

To configure realm-specific DNS in the ACLI:

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

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

2. **dns-realm**—Enter the name of the network interface that is configured for the DNS service you want to apply in this realm. If you do not configure this parameter, then the realm will use the DNS information configured in its associated network interface.

Note:

Note: If DNS is not configured on the associated **network-interface** of that realm, then following other configured realms shall be used to send out the query. These are listed in order of their preference.

a. **dns-realm** of that realm.

b. **dns-realm** of **egress-realm-id** under **sip-config**, if configured. If not configured, **egress-realm-id** would be used.

c. **dns-realm** of **home-realm-id** under **sip-config**, if configured. If not configured, **home-realm-id** would be used.

This is only applicable for SIP.

DoS ACL Configuration

If you are not using this functionality, you can leave the parameters at their default values: **average-rate-limit**, **peak-rate-limit**, **maximum-burst-size**, **access-control-trust-level**, **invalid-signal-threshold**, and **maximum-signal-threshold**.

Enabling RTP-RTCP UDP Checksum Generation

The Oracle Communications Session Border Controller can generate a UDP checksum for RTP/ RTCP packets on a per-realm basis. This feature is useful in cases where devices performing network address translation (NAT) do not pass through packets with a zero checksum from the public Internet. These packets do not make it through the NAT, even if they have the correct to and from IP address and UDP port information. The Oracle Communications Session Border Controller calculates a checksum for these packets and thereby enables them to traverse a NAT successfully.

If a checksum is already present when the traffic arrives at the hardware, the system simply relays it.

Hiding Media Updates

The Real-Time Transport Protocol uses timestamps, sequence numbers, and synchronization source (SSRC) endpoint identifiers to identify and maintain media streams between endpoints. Unexpected changes to these can cause some VoIP terminals to drop calls. You can configure the Oracle Communications Session Border Controller (SBC) to Hide Media Update (HMU) changes from endpoints and prevent the associated media flows from failing on a per-realm basis.

The HMU function monitors SSRC, timestamp, and sequence number data within RTP media traffic. The SBC stores this data within the context of individual NAT flows, and manages the flows internally with H.248. Changes to SSRC and sequence number trigger HMU logic. When triggered, the SBC refers to the SSRC, sequence number and timestamp stored in the flow information, calculates the preferred values for all three, calculates a new header checksum, and writes them to the egress media streams. The SBC keeps track of these events and provides you with the ability to review these changes.

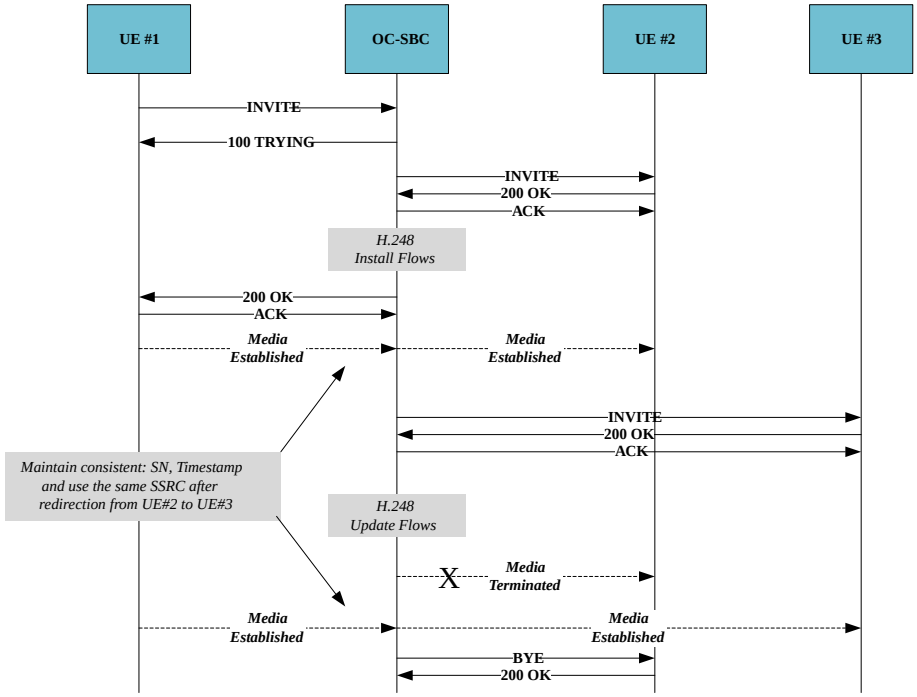
The preferred values the SBC writes include:

- The original SSRC
- A new sequence number, calculated from the last sequence number

- A new timestamp, calculated from the new received timestamp and the timestamp delta

When any change to SSRC occurs, the SBC can recognize the issue and, if configured, invoke the HMU logic. Refer to the section on HMU Support for RTP to SRTP Interworking for an explanation about the use of HMU for monitoring sequence numbers.

A common cause for SSRC changes is call transfer. The diagram below depicts a transfer from UE #2 to US #3. The call transfer creates an SSRC change that the administrator has determined may not be accepted by UE #3. By configuring HMU on the egress realm, the administrator prevents these changes from causing the call to fail.



The HMU function operates on traffic at the egress realm. The SBC evaluates SSRC changes when the traffic reaches the egress interface and corrects RTP data before it egresses the SBC. You may determine that the SBC needs to hide media changes from one realm only. If you are not configuring bi-directional HMU monitoring, refer to the following table to understand where to apply your HMU configuration.

Ingress Realm HMU Enabled	Egress Realm HMU Enabled	Result
No	No	HMU not used
No	Yes	HMU applied to response RTP
Yes	No	HMU not applied to response RTP
Yes	Yes	HMU applied to response RTP

Although the feature is RTC, the system only performs HMU on new calls after configuration. In addition, HMU supports HA. The SBC maintains all established calls across a failover. Failover may cause sessions that have HMU active to have a jump in SSRC, timestamps and sequence number. This may trigger the HMU logic and generate RTP data rewrites for flows.

The SBC maintains HMU status messages whenever received RTP packets trigger the HMU logic. You can verify how many HMU transitions have occurred on interfaces using the **show**

media host stats command, and the HMU index per media flow using the **show nat by-index** command.

> **Note:**
>
> HMU is not supported for RTCP or SRTCP packets. Regardless of HMU configuration, the SBC supports only up to 7 SSRC changes per SRTP session. Also, if HMU is disabled, the SBC supports only up to 7 SSRC changes per SRTP session for RTP and RTCP packets.

HMU State Machine

The SBC implements a state machine to manage HMU behavior that includes 5 states:

- INIT—For the first packet of a particular stream, The SBC stores sequence number, SSRC and timestamp in an HMU translation table identified by the index derived from the flow id.
- LEARN—For the next packet of a particular stream, HMU checks if there is change of SSRC. If the SSRC has not changed, the SBC keep the state machine in LEARN state. If there is a change, the SBC changes the state to WAIT.
- WAIT—For the next packet of the stream, HMU check if there is change on SSRC, OR if change in sequence number is not within the permissible limit, it moves the state to TRANS state. It saves the last ingress sequence number and also egress sequence number.
- TRANS—For the next packet of the stream, HMU checks if the changes observed in WAIT state are permanent; in that case it moves the state to HIDE and starts hiding the changed ingress information.
- HIDE—From here onwards, for the RTP packets received, the SBC calculates the sequence number from the latched last egress sequence number before forwarding packets.

> **Note:**
>
> The HMU state machine requires at least two consecutive packets with the same SSRC value before it begins to hide the SSRC value. This ensures that a pattern is set and there is consistency to this SSRC value, which tells the SBC it can be used for hiding other SSRC values. Also, in the case of a REINVITE, the SBC may create new flows. In these new flows, the HMU state machine restarts from its INIT state.

HMU Configuration

You can configure the Oracle Communications Session Border Controller (SBC) to Hide Media Update (HMU) changes from endpoints on a per-realm basis to prevent unsuccessful media flows due to media updates.

Use the following procedure to enable hide-media-update on a realm. By default, hide-media-update is disabled.

> **Note:**
>
> Enable hide-media-update on a realm, only. Do not enable hide-media-update in the media-sec-policy configuration.

1. In Superuser mode, use the following command sequence to access the realm-config configuration:

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

   When configuring a preexisting realm, use **select** to choose the realm you want to edit.

2. **hide-egress-media-update**—Set this parameter to **enabled** to enable HMU functionality on this realm.

   ```
   ORACLE(realm-config)# hide-egress-media-update enabled
   ```

3. Use done and exit to complete the configuration.

Aggregate Session Constraints Per Realm

You can set session constraints for the Oracle Communications Session Border Controller's global SIP configuration, specified session agents, and specified SIP interfaces. This forces users who have a large group of remote agents to create a large number of session agents and SIP interfaces.

With this feature implemented, however, you can group remote agents into one or more realms on which to apply session constraints.

To enable sessions constraints on a per realm basis:

- **constraint-name**—Enter the name of the constraint you want to use for this realm. You set up in the session-constraints confiuration.

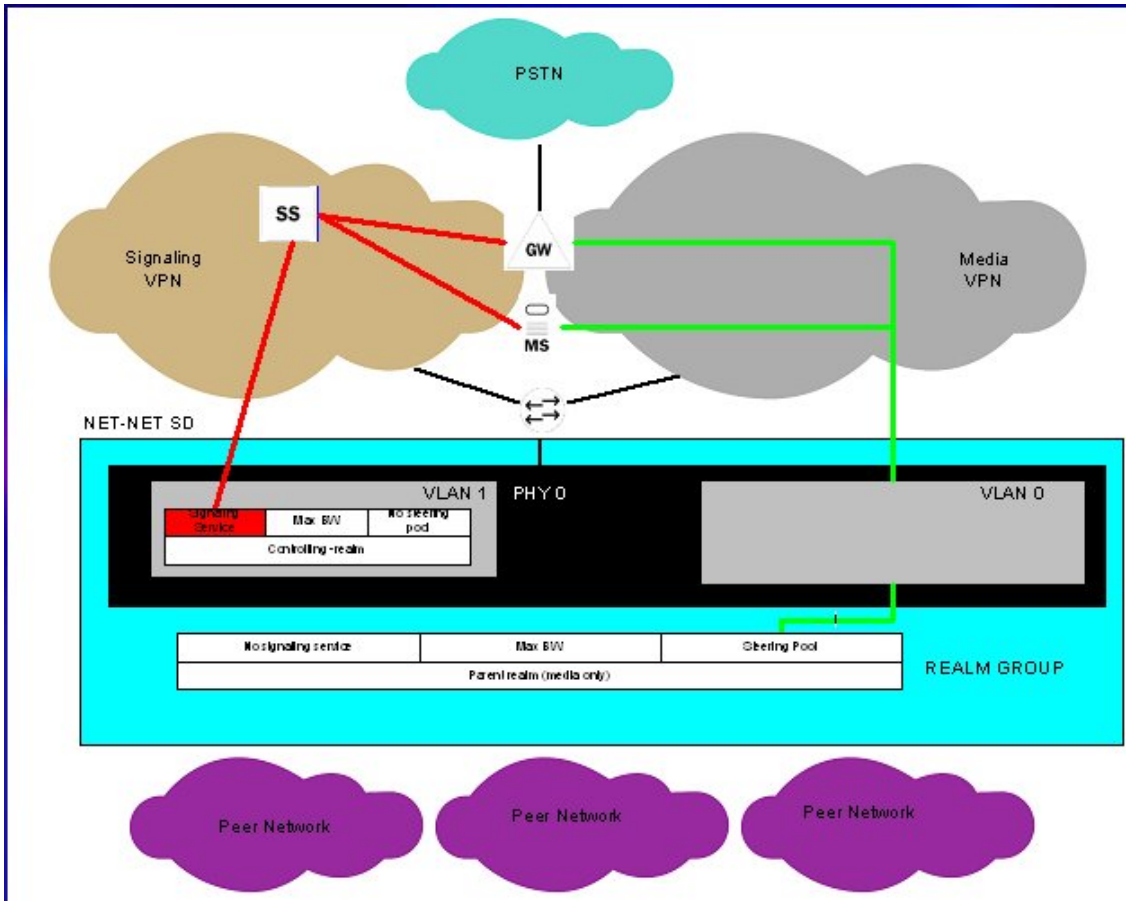
Admission Control Configuration

You can set admission control based on bandwidth for each realm by setting the **max-bandwidth** parameter for the realm configuration. Details about admission control are covered in this guide's *Admission Control and QoS* chapter.

Nested Realms

Configuring nested realms allows you to create backbone VPN separation for signaling and media. This means that you can put signaling and media on separate network interfaces, that the signaling and media VPN can have different address spaces, and that the parent realm has one media-only sub-realm.

The following figure shows the network architecture.



In addition, you can achieve enhanced scalability by using a shared service interface. A single service address is shared across many customers/peers, customer specific policies for bandwidth use and access control are preserved, and you can achieve fine-grained policy control.

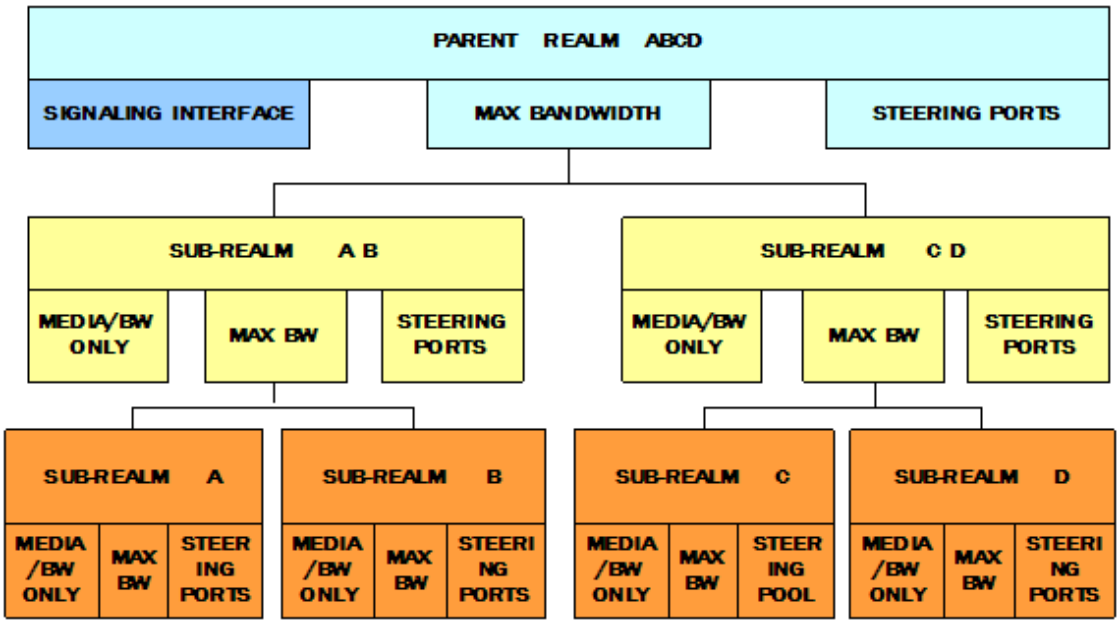
These benefits are achieved when you configure these types of realms:

- Realm group—A hierarchical nesting of realms identified by the name of the highest order realm.
- Controlling realm—A realms for which a signaling interface is configured. For example, you might configure these signaling interfaces in the following configurations: SIP-NAT, SIP port or H.323 stack. Typically, this is the highest order realm for the parent realm in a realm group.
- Parent realm—A realm that has one or more child realms. A parent realm might also be the child realm of another realm group.
- Child realm—A realm that is associated with a single higher order parent realm. A child might also be the parent realm of another realm group. Child realms inherit all signaling and steering ports from higher order realms.
- Media-only realm—A realm for which there is no configured signaling interface directly associated. Media-only realms are nested within higher order realms.

As these definitions suggest, parent and child realms can be constructed so that there are multiple nesting levels. Lower order realms inherit the traits of the realms above them, including: signaling service interfaces, session translation tables, and steering pools.

Since realms inherit the traits of the realms above them in the hierarchy, you will probably want to map what realms should be parents and children before you start configuring them. These

relationships are constructed through one parameter in the realm configuration that identifies the parent realm for the configuration. If you specify a parent realm, then the realm you are configuring becomes a child realm subject to the configured parameters you have established for that parent. And since parent realms can themselves be children of other realm, it is important that you construct these relationships with care.



Configuring Nested Realms

When you are configuring nested realms, you can separate signaling and media by setting realm parameters in the SIP interface configuration, the H.323 stack configuration, and the steering ports configuration.

- The realm identifier you set in the SIP interface configuration labels the associated realm for signaling.
- The realm identifier you set in the H.323 stack configuration labels the associated realm for signaling.
- The realm identifier you set in the steering ports configuration labels the associated realm for media.

Constructing a hierarchy of nested realms requires that you note which realms you want to handle signaling, and which you want to handle media.

In the SIP port configuration for the SIP interface and in the H.323 stack configuration, you will find an allow anonymous parameter that allows you to set certain access control measures. The table below outlines what each parameter means.

Allow Anonymous Parameter	Description
all	All anonymous connections allowed.
agents-only	Connections only allowed from configured session agents.
realm-prefix	Connections only allowed from addresses with the realm's address prefix and configured session agents.

Allow Anonymous Parameter	Description
registered	Connections allowed only from session agents and registered endpoints. (For SIP only, a REGISTER is allowed for any endpoint.)
register-prefix	Connections allowed only from session agent and registered endpoints. (For SIP only, a REGISTER is allowed for session agents and a matching realm prefix.)

Parent and Child Realm Configuration

To configure nested realms, you need to set parameters in the realm configuration.

To configure parent and child realms:

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

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

2. **parent-realm**—Enter the identifier of the realm you want to name as the parent. Configuring this parameter makes the realm you are currently configuring as the child of the parent you name. As such, the child realm is subject to the configured parameters for the parent.

Required Signaling Service Parameters

To configure nested realms, you need to set parameters in the realm configuration and in the configurations for the signaling protocols you want to use.

To configure H.323 stack parameters for nested realms:

1. Access the **h323 > h323-stacks** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# h323
ORACLE(h323)# h323-stack
ORACLE(h323-stack)# select
```

2. **allow-anonymous**—Enter the admission control of anonymous connections accepted and processed by this H.323 stack. The default is **all**. The valid values are:
   - all—Allow all anonymous connections
   - agents-only—Only requests from session agents allowed
   - realm-prefix—Session agents and address matching relam prefix

Aggregate Session Constraints Nested Realms

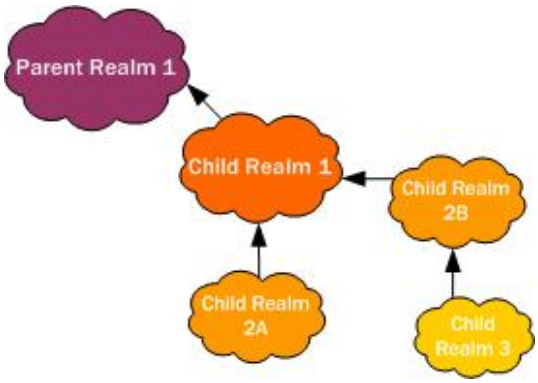
In addition to setting session constraints per realm for SIP and H.323 sessions, you can also enable the Oracle Communications Session Border Controller to apply session constraints across nested realms. When you set up session constraints for a realm, those constraints

apply only to the realm for which they are configured without consideration for its relationship either as parent or child to any other realms.

You can also, however, enable the Oracle Communications Session Border Controller to take nested realms into consideration when applying constraints. For example, if a call enters on a realm that has no constraints but its parent does, then the constraints for the parent are applied. This parameter is global and so applies to all realms on the system. For the specific realm the call uses and for all of its parents, the Oracle Communications Session Border Controller increments the counters upon successful completion of an inbound or outbound call.

In the following example, you can see one parent realm and its multiple nested, child realms. Now consider applying these realm constraints:

- Parent Realm 1—55 active sessions
- Child Realm 1—45 active sessions
- Child Realm 2A—30 active sessions
- Child Realm 2B—90 active sessions
- Child Realm 3—20 active sessions



Given the realm constraints outlined above, consider these examples of how global session constraints for realms. For example, a call enters the Oracle Communications Session Border Controller on Child Realm 2B, which has an unmet 90-session constraint set. Therefore, the Oracle Communications Session Border Controller allows the call based on Child Realm 2B. But the call also has to be within the constraints set for Child Realm 1 and Parent Realm 1. If the call fails to fall within the constraints for either of these two realms, then the Oracle Communications Session Border Controller rejects the call.

Impact to Other Session Constraints and Emergency Calls

You can set up session constraints in different places in your Oracle Communications Session Border Controller configuration. Since session agents and SIP interfaces also take session constraints, it is import to remember the order in which the Oracle Communications Session Border Controller applies them:

1. Session agent session constraints
2. Realm session constraints (including parent realms)
3. SIP interface session constraints

   Emergency and priority calls for each of these is exempt from session constraints. That is, any call coming into the Oracle Communications Session Border Controller marked priority is processed.

Session Constraints Configuration

You enabled use of session constraints for nested realms across the entire system by setting the **nested-realms-stats** parameter in the session router configuration to **enabled**.

1. Access the **session-router-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-router
ORACLE(session-router-config)#
```

2. **nested-realms-stats**—Change this parameter from **disabled** (default) to **enabled** if you want the Oracle Communications Session Border Controller to apply session constraints across all nested realms (realms that are children to other realms)
3. Save and activate your configuration.

Realm-Based Packet Marking

The Oracle Communications Session Border Controller supports TOS/DiffServ functions that allow you to

- Set up realm-based packet marking by media type, either audio-voice or video
- Set up realm-based packet marking for signaling, either SIP or H.323

Upstream devices use these markings to classify traffic in order to determine the priority level of treatment it will receive.

By default, the SBC does not pass DSCP codes in ingress packets to egress packets. You must configure a **media-policy** with desired TOS changes and affix those policies to the realms on which you want to define egress types of service. Without a**media-policy**, the SBC includes the default DSCP code, CS0 (Hex 0x00), as the DSCP code to all egress media packets.

TOS Passthrough Configuration

As stated above, the SBC does not passthrough received DSCP values transparently. If this is the desired behavior, no config change is required. This is the default behavior. Packets sent by SBC show DSCP value 0x00.

If passthrough support is desired, you can enable the **sip-config** option called **use-recvd-dscp-marking** which enables passthrough support. With this option enabled, the SBC passes the DSCP value which was received through to egress. To enable this option in **sip-config**, set the option as shown below.

```
ORACLE(sip-config)#options +use-recvd-dscp-marking
```

About TOS DiffServ

TOS and DiffServ are two different mechanisms used to achieve QoS in enterprise and service provider networks; they are two different ways of marking traffic to indicate its priority to upstream devices in the network.

For more information about TOS (packet) marking, refer to:

- IETF RFC 1349 (http://www.ietf.org/rfc/rfc1349.txt)

For more information about DiffServ, refer to:

- IETF RFC 2474 (http://www.ietf.org/rfc/rfc2474.txt)
- IETF RFC 2475 (http://www.ietf.org/rfc/rfc2475.txt).

ToS Byte

The TOS byte format is as follows:

The TOS byte is broken down into three components:

- **Precedence**—The most used component of the TOS byte, the precedence component is defined by three bits. There are eight possible precedence values ranging from 000 (decimal 0) through 111 (decimal 7). Generally, a precedence value of 000 refers to the lowest priority traffic, and a precedence value of 111 refers to the highest priority traffic.
- **TOS**—The TOS component is defined by four bits, although these bits are rarely used.
- **MBZ**—The must be zero (MBZ) component of the TOS byte is never used.

DiffServ Byte

Given that the TOS byte was rarely used, the IETF redefined it and in doing so created the DiffServ byte.

The DiffServ byte format is as follows:

The DiffServ codepoint value is six bits long, compared to the three-bit-long TOS byte's precedence component. Given the increased bit length, DiffServ codepoints can range from 000000 (decimal 0) to 111111 (decimal 63).

> **Note:**
>
> By default, DiffServ codepoint mappings map exactly to the precedence component priorities of the original TOS byte specification.

Packet Marking for Media

You can set the TOS/DiffServ values that define an individual type or class of service for a given realm. In addition, you can specify:

- One or more audio media types for SIP and/or H.323

- One or more video media types for SIP and/or H.323
- Both audio and video media types for SIP and/or H.323

For all incoming SIP and H.23 requests, the media type is determined by negotiation or by preferred codec. SIP media types are determined by the SDP, and H.323 media types are determined by the media specification transmitted during call setup.

Configuring Packet Marking by Media Type

This section describes how to set up the media policy configuration that you need for this feature, and then how to apply it to a realm.

These are the ACLI parameters that you set for the media policy:

```
name         media policy name
tos-settings  list of TOS settings
```

> **Note:**
>
> The **media-policy**, **tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

This is the ACLI parameter that you set for the realm:

```
media-policy default media policy name
```

Packet Marking Configuration

To set up a media policy configuration to mark audio-voice or video packets:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ```

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

   ```
   ORACLE(media-manager)# media-policy
   ORACLE(media-policy)#
   ```

   From this point, you can configure media policy parameters. To view all configuration parameters for media profiles, enter a **?** at the system prompt.

4. Type **media-policy** and press Enter.

   ```
   ORACLE(media-manager)# media-policy
   ```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

5. **name**—Create a reference name for this policy and press Enter.
6. Type **tos-settings** and press Enter.

```
ORACLE(media-policy)# tos-settings
```

7. **media-type**—Enter the media type that you want to use for this group of TOS settings. You can enter any of the IANA-defined media types for this value: audio, example, image, message, model, multipart, text, and video. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-type message
```

8. **media-sub-type**—Enter the media sub-type you want to use for the media type. This value can be any of the sub-types that IANA defines for a specific media type. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-sub-type sip
```

9. **media-attributes**—Enter the media attribute that will match in the SDP. This parameter is a list, so you can enter more than one value. The values are case-sensitive and can be up to 255 characters in length. This parameter has no default.

   If you enter more than one media attribute value in the list, then you must enclose your entry in quotation marks ().

```
ORACLE(tos-settings)# media-attributes sendonly sendrecv
```

10. **tos-value**—Enter the TOS value you want applied for matching traffic. This value is a decimal or hexidecimal value. The valid range is:
    - 0x00 to 0xFF.

```
ORACLE(tos-settings)# tos-value 0xF0
```

11. Save and activate your configuration.

Applying a Media Policy to a Realm

To apply a media policy to a realm:

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

```
ORACLE# configure terminal
```

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

```
ORACLE(configure)# media-manager
```

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

```
ORACLE(media-manager)# realm
ORACLE(realm)#
```

4. **media-policy**—Enter the unique name of the media policy you want to apply to this realm.

Signaling Packet Marking Configuration

ToS marking for signaling requires you to configure a media policy and set the name of the media policy in the appropriate realm configuration.

This section shows you how to configure packet marking for signaling.

Configuring a Media Policy for Signaling Packet Marking

To set up a media policy configuration to mark signaling packets:

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

```
ORACLE# configure terminal
```

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

```
ORACLE(configure)# media-manager
```

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

```
ORACLE(media-manager)# media-policy
ORACLE(media-policy)#
```

From this point, you can configure media policy parameters. To view all media policy configuration parameters, enter a **?** at the system prompt.

4. Type **media-policy** and press Enter.

```
ORACLE(media-manager)# media-policy
```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

5. **name**—Create a reference name for this policy and press Enter.
6. Type **tos-settings** and press Enter.

```
ORACLE(media-policy)# tos-settings
```

> **Note:**
>
> The **media-policy**, **tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

7. **media-type**—Enter the media type that you want to use for this group of TOS settings. You can enter any of the IANA-defined media types for this value: audio, example, image, message, model, multipart, text, and video. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-type message
```

8. **media-sub-type**—Enter the media sub-type you want to use for the media type. This value can be any of the sub-types that IANA defines for a specific media type. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-sub-type sip
```

9. **media-attributes**—Enter the media attribute that will match in the SDP. This parameter is a list, so you can enter more than one value. The values are case-sensitive and can be up to 255 characters in length. This parameter has no default.

   If you enter more than one media attribute value in the list, then you must enclose your entry in quotation marks ().

```
ORACLE(tos-settings)# media-attributes sendonly sendrecv
```

10. **tos-value**—Enter the TOS value you want applied for matching traffic. This value is a decimal or hexidecimal value. The valid range is:

    - 0x00 to 0xFF.

```
ORACLE(tos-settings)# tos-value 0xF0
```

11. Save and activate your configuration.

Applying a Media Policy to a Realm

To apply a media policy to a realm:

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

```
ORACLE# configure terminal
```

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

```
ORACLE(configure)# media-manager
```

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

```
ORACLE(media-manager)# realm
ORACLE(realm)#
```

4. **media-policy**—Enter the unique name of the media policy you want to apply to this realm.

Using Class Profile for Packet Marking

Class profile provides an additional means of ToS marking, but only for limited circumstances. Use class-profile only if you are marking ToS on traffic destined for a specific To address, and

when media-policy is not used on the same realm. Using media-policy for ToS marking is, by far, more common.

To configure a class profile, you prepare your desired media policy, create the class profile referencing the media policy and the To address, and set the name of the class profile in the appropriate realm configuration.

Class Profile and Class Policy Configuration

This section shows you how to configure packet marking using a class profile.

To configure the class profile and class policy:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

   ```
   ORACLE(session-router)# class-profile
   ORACLE(class-profile)#
   ```

4. Type **policy** and press Enter to begin configuring the class policy.

   ```
   ORACLE(class-profile)# policy
   ```

   From this point, you can configure class policy parameters. To view all class policy configuration parameters, enter a **?** at the system prompt.

5. **profile-name**—Enter the unique name of the class policy. When you apply a class profile to a realm configuration, you use this value.
6. **to-address**—Enter a list of addresses to match to incoming traffic for marking. You can use E.164 addresses, a host domain address, or use an asterisk (*) to set all host domain addresses.
7. **media-policy**—Enter the name of the media policy you want to apply to this class policy.

Applying a Class Policy to a Realm

To apply a class policy to a realm:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ```

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

```
ORACLE(media-manager)# realm
ORACLE(realm)#
```

4. **class-profile**—Enter the name if the class profile to apply to this realm. This is the name you set in the **profile-name** parameter of the class-policy configuration.

Differentiated Services for DNS and ENUM

The Oracle Communications Session Border Controller can mark DNS/ENUM packets with a configurable differentiated services code point (DSCP).

Certain service providers mandate support for Differentiated Services (DS) on all traffic streams exiting any network element. This mandate requires that network elements function as a DS marker — that is as a device that sets the Distributed Services Codepoint (DSCP) in the Differentiated Services field of the IP header.

Previous software releases provided the capabilities to mark standard SIP and Real-Time Protocol (RTP) messages. Release S-CX6.4.0M3 adds the capability to mark DNS and ENUM queries.

For basic information about DS, refer to:

- The *ACLI Configuration Guide* Realm-Based Packet Marking topic in the Realms and Nested Realms chapter, which provides information on currently supported DS marking of SIP and RTP packets
- *IETF RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers* (http://www.ietf.org/rfc/rfc2474.txt)
- *IETF RFC 2475, An Architecture for Differentiated Services* (http://www.ietf.org/rfc/rfc2475.txt).

DS provides a mechanism to define and deliver multiple and unique service classifications that can be offered by a service provider. Specific service classifications are identified by a DSCP, essentially a numeric index. The DSCP maps to a per-hop-behavior (PHB) that defines an associated service class. PHBs are generally defined in terms of call admission controls, packet drop criteria, and queue admission algorithms. In theory, DS supports 64 distinct classifications. In practice, however, network offerings generally consist of a much smaller suite, which typically includes:

- Default PHB - required best effort service — defined in RFC 2474
- Expedited Forwarding (EF) PHB - low-loss, low-latency — defined in RFC 3246, An Expedited Forwarding PHB
- Assured Forwarding (AF) PHB - assured delivery within subscriber limits — defined in *RFC 2597, Assured Forwarding PHB Group*, and *RFC 3260, New Terminology and Clarifications for Diffserv*
- Class Selector PHBs - maintain compatibility with previous TOS (type of service) precedence usage — defined in RFC 2474

Marking of DNS and ENUM queries requires the creation of a Differentiated Services media policy, and the assignment of such a policy to a specific realm.

Differentiated Services for DNS and ENUM Configuration

To create a Differentiated Services media policy:

1. From Superuser mode, use the following ACLI command sequence to move to media-policy configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-policy
ORACLE(media-policy)#
```

2. Use the required **name** parameter to provide a unique identifier for this media-policy instance.

```
ORACLE(media-policy)# name diffServeDnsEnum
ORACLE(media-policy)#
```

3. Use the tos-settings command to move to tos-settings configuration mode.

```
ORACLE(media-policy)# tos-settings
ORACLE(tos-settings)#
```

> **Note:**
>
> The **media-policy**, **tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

4. Use the required media-type parameter to identify the packet-type subject to Diffentiated Services marking.

   For DNS/ENUM packets, use message/dns.

```
ORACLE(tos-settings)# media-type message/dns
ORACLE(tos-settings)#
```

5. Use the required tos-value parameter to specify the 6-bit DSCP to be included in the Differentiated Services field of the IP header.

   Specify the DSCP as an integer (within the range from 0 to 63). The DSCP can be expressed in either decimal or hexadecimal format.

```
ORACLE(tos-settings)# tos-value 0x30
ORACLE(tos-settings)#
```

6. Other displayed parameters, media-attributes and media-sub-type, can be safely ignored.
7. Use the done and exit commands to complete Differentiated Services media policy configuration and return to media-policy configuration-mode.

```
ORACLE(tos-settings)# done
tos-settings
        media-type                  message/dns
        media-sub-type
```

```
        tos-value                0x30
        media-attributes
ORACLE(tos-settings)# exit
ORACLE(media-policy)#
```

8. If necessary, repeat steps 2 through 7 to create additional Differentiated Services media policies.
9. Assign this media policy to the target realm via the **media-policy** parameter in a **realm-config** object.

SIP-SDP DCSP Marking ToS Bit Manipulation

Used to indicate priority and type of requested service to devices in the network, type of service (TOS) information is included as a set of four-bit flags in the IP header. Each bit has a different purpose, and only one bit at a time can be set: There can be no combinations. Available network services are:

- Minimum delay—Used when latency is most important
- Maximum throughput—Used when the volume of transmitted data in any period of time is important
- Maximum reliability—Used when it is important to assure that data arrives at its destination without requiring retransmission
- Minimum cost—Used when it is most important to minimize data transmission costs

The Oracle Communications Session Border Controller's support for type of service (TOS allows you to base classification on the media type as well as the media subtype. In prior releases, you can configure the Oracle Communications Session Border Controller to mark TOS bits on outgoing packets using a media policy. Supported media types include audio, video, application, data, image, text, and message; supported protocol types are H.225, H.245, and SIP. Note that, although H.225 and H.245 are not part of any IANA types, they are special cases (special subtypes) of message for the Oracle Communications Session Border Controller. When these criteria are met for an outgoing packet, the Oracle Communications Session Border Controller applies the TOS settings to the IP header. The augmented application of TOS takes matching on media type or protocol and expands it to match on media type, media-sub-type, and media attributes.

The new flexibility of this feature resolves issues when, for example, a customer needs to differentiate between TV-phone and video streaming. While both TV-phone and video streaming have the attribute "media=video," TV-phone streaming has "direction=sendrcv" prioritized at a high level and video has direction=sendonly or recvonly with middle level priority. The Oracle Communications Session Border Controller can provide the appropriate marking required to differentiate the types of traffic.

In the media policy, the **tos-values** parameter accepts values that allow you to create any media type combination allowed by IANA standards. This is a dynamic process because theOracle Communications Session Border Controller generates matching criteria directly from messages.

The new configuration takes a media type value of any of these: audio, example, image, message, model, multipart, text, and video. It also takes a media sub-type of any value specified for the media type by IANA; however, support for T.38 must be entered exactly as **t.38** (rather than t38). Using these values, the Oracle Communications Session Border Controller creates a value Based on a combination of these values, the Oracle Communications Session Border Controller applies TOS settings.

You also configure the TOS value to be applied, and the media attributes you want to match.

You can have multiple groups of TOS settings for a media policy.

ToS Bit Manipulation Configuration

This section provides instructions for how to configure TOS bit manipulation on your Oracle Communications Session Border Controller.

To configure TOS bit manipulation:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-policy** and press Enter.

   ```
   ORACLE(media-manager)# media-policy
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

4. **name**—Create a reference name for this policy and press Enter.
5. Type **tos-settings** and press Enter.

   ```
   ORACLE(media-policy)# tos-settings
   ```

   > **Note:**
   >
   > The **media-policy**, **tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

6. **media-type**—Enter the media type that you want to use for this group of TOS settings. You can enter any of the IANA-defined media types for this value: audio, example, image, message, model, multipart, text, and video. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

   ```
   ORACLE(tos-settings)# media-type message
   ```

7. **media-sub-type**—Enter the media sub-type you want to use for the media type. This value can be any of the sub-types that IANA defines for a specific media type. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

   ```
   ORACLE(tos-settings)# media-sub-type sip
   ```

8. **media-attributes**—Enter the media attribute that will match in the SDP. This parameter is a list, so you can enter more than one value. The values are case-sensitive and can be up to 255 characters in length. This parameter has no default.

If you enter more than one media attribute value in the list, then you must enclose your entry in quotation marks ().

```
ORACLE(tos-settings)# media-attributes sendonly sendrecv
```

9. **tos-value**—Enter the TOS value you want applied for matching traffic. This value is a decimal or hexidecimal value. The valid range is:
   - 0x00 to 0xFF.

   ```
   ORACLE(tos-settings)# tos-value 0xF0
   ```

10. Save and activate your configuration.

DSCP Marking for MSRP and Media Over TCP

The Oracle Communications Session Border Controller supports Differentiated Services Code Point (DSCP) marking of MSRP and Media over TCP traffic. This feature may be used for MSRP traffic in both B2BUA and non-B2BUA modes.

In order to configure the Oracle Communications Session Border Controller to mark MSRP or media over TCP packets with a value in the IP header's DSCP field, create a **media-manager**, **media-policy**, **tos-settings** configuration element and set the **media-type** parameter to **message**. This has the effect of marking all traffic described by prior SDP with m=message with this DSCP value.

Note that setting **media-type** to **message** can potentially mark a large range of traffic. For other common traffic to remain marked differently than marked MSRP, you need to create additional **tos-settings** configuration elements with valid **media-sub-type** configurations. The following values may be provisioned in the **media-sub-type** parameter to mark these other types of traffic differently (with an accompanying **tos-value**).

- sip
- li
- dns

PAI Header and FQDN Manipulation

You can configure the SBC to manipulate the content of egress messages using realm parameters instead of HMR. By setting these parameters, you cause the SBC to perform these manipulations on specific SIP methods that egress the realm.
Realm-based content manipulation that applies to surrogate agent operation includes:

- P-Asserted Identity (PAI) content
- Fully Qualified Domain Name (FQDN) content

PAI Header Manipulation

You can configure the **realm-config** element with PAI manipulation behavior that applies to the realm's egress traffic.
You set the **P-Asserted-Identity** parameter in conjunction with the **P-Asserted-Identity-For** parameter to insert PAI headers into that realm's egress traffic. You specify the methods that you want to include by configuring the **P-Asserted-Identity-For** parameter for any or all of the following methods:

- INVITE
- ACK
- BYE
- REGISTER

You enable the behavior by configuring the **P-Asserted-Identity** parameter with the PAI value you want to insert. If you do not set both of these parameters, the SBC does not insert the headers. This manipulation deals with the selected headers for egress INVITE and REGISTER flows, and does not interfere with PRACK INVITE flows.

If the originating message does not include any PAI headers, the SBC inserts the headers per configuration. If PAI headers are already present, inserted by an upstream device, the SBC:

- Adds any **P-Asserted-Identity** parameter headers to the top of the header list.
- Replaces any existing sip:PAI headers with the sip:PAI headers you configure with the **P-Asserted-Identity** parameter.
- Replaces any existing tel:PAI headers with the tel:PAI headers you configure with the **P-Asserted-Identity** parameter.
- The SBC retains any existing tel:PAI headers or sip:PAI headers at the bottom of the PAI list that are not replaced by **P-Asserted-Identity** tel:PAI or sip:PAI values, as above.

Example configuration syntax includes:

```
ORACLE(realm-config)# P-Asserted-Identity sip:MyPai


ORACLE(realm-config)# P-Asserted-Identity-For (INVITE BYE)
```

Examples

Example 1—When you configure the **P-Asserted-Identity** as a TEL URI and there is a tel URI PAI header in the incoming message, the SBC replaces the incoming TEL URI with the PAI header configured in the **P-Asserted-Identity** parameter in the egress messages specified in the **P-Asserted-Identity-For** parameter.

- Incoming PAI: tel:56789
- P-Asserted-Identity configured: tel:12345
- P-Asserted-Identity-For: INVITE
- Output: Egress PAI: tel:12345

Example 2—When you configure the **P-Asserted-Identity** parameter as a TEL URI and there is a SIP URI in the incoming message, the SBC inserts the SIP URI with the PAI header configured in the **p-asserted-identity** parameter in the egress messages specified in the **P-Asserted-Identity-For** parameter.

- Incoming PAI headers: sip:123@oracle.com
- P-Asserted-Identity configured: tel:12345
- P-Asserted-Identity-For: INVITE
- Output Egress PAI: tel:12345 sip:123@oracle.com

FQDN Hostname Manipulation

Additional configuration allows you to modify the hostname used in SBC response headers. Some deployments need you to specify a hostname value in specific headers. You can configure the SBC to make these changes using two **realm-config** parameters. You can configure the system to apply your hostname to the following SIP headers:

- FROM
- TO
- CONTACT
- RUI

You specify a desired hostname value by configuring in the **fqdn-hostname** parameter with the desired hostname. Furthermore, you specify the headers you want to change using the **fqdn-hostname-in-header** parameter. If either of these parameters are empty, the SBC does not set this hostname in any headers.

Example settings for these parameter include:

```
ORACLEORACLE(realm-config)# fqdn-hostname MyHostName


ORACLEORACLE(realm-config)# fqdn-hostname-in-header FROM, TO
```

Configure Manipulation Attributes on a Realm

In the SBC ACLI, you can access manipulation parameters applicable to surrogate agent operation using the path **media-manager**, **realm-config**.

To configure targeted manipulation parameter on the SBC:

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

   ```
   ORACLE# configure terminal
   ```

2. Type media-manager and press Enter to access the media manager-related objects.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type realm-config and press Enter.

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

4. Create or select the realm-config on which the softswitch resides.
5. P-Asserted-Identity — Enter the string you want to use to set the identity within PAI headers for traffic egressing this realm.

   ```
   ORACLE(realm-config)# P-Asserted-Identity MyPAI
   ```

6. P-Asserted-Identity-For — List the methods for which the SBC includes a PAI header using the PAI identity you set in this realm's p-asserted-identity parameter. Separate multiple values with a comma.
   - INVITE
   - BYE
   - ACK
   - REGISTER

   ```
   (realm-config)# P-Asserted-Identity-For (INVITE BYE)
   ```

7. fqdn-hostname — Enter the hostname you want to in include in the selected headers for this realm's egress traffic.

   ```
   ORACLE(realm-config)# fqdn-hostname MyHostName
   ```

8. fqdn-hostname-in-header — List the headers for which the SBC inlcudes the hostname that you configured in the fqdn-host-name parameter. Separate multiple values with a comma.
   - FROM
   - TO
   - CONTACT
   - RURI

   ```
   (realm-config)# fqdn-hostname-in-header FROM, TO
   ```

9. Type done to save changes to this realm-config.
10. Save and activate your configuration.

Configure the applicable **local-policy**. This configuration includes setting the **auth-user-lookup** parameter in the applicable **local-policy-attribute** with the same value as the **auth-user-lookup** above.

SDP Alternate Connectivity

The Oracle Communications Session Border Controller can create an egress-side SDP offer containing both IPv4 and IPv6 media addresses via a mechanism which allows multiple IP addresses, of different address families (i.e., IPv4 & IPv6) in the same SDP offer. Our implementation is based on the RFC draft "draft-boucadair-mmusic-altc-09".

Each realm on the Oracle Communications Session Border Controller can be configured with an alternate family realm on which to receive media in the alt family realm parameter in the realm config. As deployed, one realm will be IPv4, and the alternate will be IPv6. The Oracle Communications Session Border Controller creates the outbound INVITE with IPv4 and IPv6 addresses to accept the media, each in an a=altc: line and each in its own realm. The IP addresses inserted into the a=altc: line are from the egress realm’s and alt-realm-family realm’s steering pools. Observe in the image how the red lines indicate the complementary, alternate realms.

You can configure the order in which the a=altc: lines appear in the SDP in the pref-address-type parameter in the realm-config. This parameter can be set to

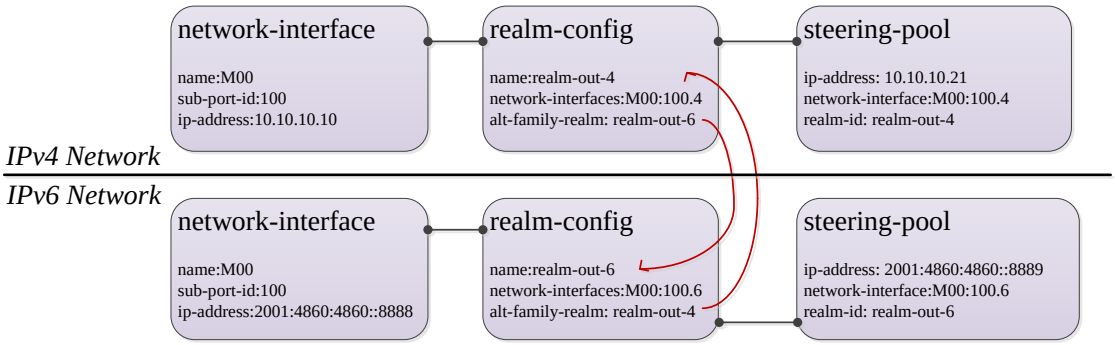
- IPv4 - SDP contains the IPv4 address first

- IPv6 - SDP contains the IPv6 address first
- NONE - SDP contains the native address family of the egress realm first

In the 200OK to the INVITE, the callee chooses either the IPv6 or IPv4 address to use for the call's media transport between itself and Oracle Communications Session Border Controller. After the Oracle Communications Session Border Controller receives the 200OK, the chosen flow is installed, and the unused socket is discarded.

For two realms from different address families to share the same phy-interface and vlan, you use a .4 or .6 tag in the network-interface reference. When IPv4 and IPv6 realms share the same network-interface and VLAN, you identify them by realm name and network-interface configured as:

- IPv4 - <phy-interface>:<vlan>.4
- IPv6 - <phy-interface>:<vlan>.6



If the INVITE's egress realm is IPv6, pref-address-type = NONE, the outbound SDP has these a=altc: lines:

```
a=altc:1 IPv6 2001:4860:4860::8889 20001
a=altc:2 IPv4 10.10.10.21 20001
```

If the INVITE's egress realm is IPv6, pref-address-type = IPv4, the outbound SDP has these a=altc: lines:

```
a=altc:1 IPv4 10.10.10.21 20001
a=altc:2 IPv6 2001:4860:4860::8889 20001
```

SDP Alternate connectivity supports B2B and hairpin call scenarios. SDP Alternate connectivity also supports singleterm, B2B, and hairpin call scenarios.

When providing SDP alternate connectivity for SRTP traffic, in the security policy configuration element, the network-interface parameter's value must be configured with a .4 or .6 suffix to indicate IPv4 or IPv6 network, respectively.

SDP Alternate Connectivity Configuration

To configure SDP alternate connectivity:

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

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

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **alt-realm-family** — Enter the realm name of the alternate realm, from which to use an IP address in the other address family. If this parameter is within an IPv4 realm configuration, you will enter an IPv6 realm name.
4. **pref-address-type** — Set the order in which the a=altc: lines suggest preference. Valid values are:
   - none — address family type of egress realm signaling
   - ipv4 — IPv4 realm/address first
   - ipv6 — IPv6 realm/address first
5. Type **done** to save your configuration.

Steering Pools

Steering pools define sets of ports that are used for steering media flows through the Oracle Communications Session Border Controller. These selected ports are used to modify the SDP to cause receiving session agents to direct their media toward this system. Media can be sent along the best quality path using these addresses and ports instead of traversing the shortest path or the BGP-4 path.

For example, when the Oracle Communications Session Border Controller is communicating with a SIP device in a specific realm defined by a steering pool, it uses the IP address and port number from the steering pool's range of ports to direct the media. The port the Oracle Communications Session Border Controller chooses to use is identified in the SDP part of the message.

> **Note:**
>
> The values entered in the steering pool are used when the system provides NAT, PAT, and VLAN translation.

Configuration Overview

To plan steering pool ranges, take into account the total sessions available on the box, determine how many ports these sessions will use per media stream, and assign that number of ports to all of the steering pools on your Oracle Communications Session Border Controller.

For example, if your Oracle Communications Session Border Controller can accommodate 500 sessions and each session typically uses 2 ports, you would assign 1000 ports to each steering pool. This strategy provides for a maximum number of ports for potential use, without using extra resources on ports your Oracle Communications Session Border Controller will never use.

The following lists the steering pool parameters you must configure:

- IP address—IP address of the steering pool.
- start port—Port number that begins the range of ports available to the steering pool. You must define this port to enable the system to perform media steering and NAT operations.
- end port—Port number that ends the range of ports available to the steering pool. You must define this port to enable the system to perform media steering and NAT operations.
- realm id—Identifies the steering pool's realm. The steering pool is restricted to only the flows that originate from this realm.

> **Note:**
>
> The combination of entries for IP address, start port, and realm ID must be unique in each steering pool. You cannot use the same values for multiple steering pools.

Each bidirectional media stream in a session uses two steering ports, one in each realm (with the exception of audio/video calls that consume four ports). You can configure the start and end port values to provide admission control. If all of the ports in all of the steering pools defined for a given realm are in use, no additional flows/sessions can be established to/from the realm of the steering pool.

> **Note:**
>
> If your signaling interface and **steering-pool** use the same IP address over TCP, you must ensure the **steering-pool** port range does not include the signaling interface's port number. If it does, the system cannot properly process signaling traffic on that interface.

Allocation Strategies for Steering Pools

You can configure the SBC with three types of steering pools to allocate network ports for specific types of network traffic. These pool types include audio/video, MSRP and mixed media types. Establishing these pool types provides more efficient use of media ports. The SBC provides you with a means of monitoring port usage by type to troubleshoot and refine these configurations.

By default, the SBC does not allocate steering pool ports based on media type. These default allocations can establish scenarios, including sequential allocation and port release, wherein port usage is inefficient. By configuring or monitoring your realms for traffic type use, you can plan for and adjust configuration based on expected port usage:

- Audio/Video sessions—Consume 4 ports, including two audio ports supporting the two traffic directions and two additional ports for RTP and RTCP
- MSRP sessions—Consume 2 ports supporting the two traffic directions

- Hairpin Audio sessions—Consume 8 ports, including four audio flows supporting the two traffic directions, two for RTP, and two for RTCP

Per recommendation in RFC 3550, the SBC allocates RTP and RTCP ports for audio calls sequentially. This behavior can reduce the number of ports you have configured for a **steering-pool**.

For example, if you have configured 100 ports on a realm and it receives 100 MSRP calls, the SBC allocates the 100 ports for the MSRP call flows. When the users terminate these sessions, those terminations happen randomly, leaving multiple non-sequential ports open.

If the users terminate 20 random sessions, the number of ports that become available are not likely to support 5 A/V calls. This is because the SBC may not be able to find open, sequential ports to support RTP and RTCP. This can cause calls to fail, with the system reporting no ports available as the reason.

This feature provides you with a means of establishing multiple **steering-pool** objects for a realm that use your configured allocation strategy as a means of determining pool affinity based on traffic type. The SBC recognizes the traffic type and chooses the appropriate **steering-pool** based on **port-allocation-strategy** and current pool capacity:

- **mixed**—(Default) Supports all types of sessions
- **single**—Recommended for MSRP sessions
- **pair**—Recommended for audio and video sessions

The SBC tracks pool utilization and, assuming the **port-allocation-strategy** is appropriate, allocates traffic preferred for an empty pool to use ports from a pool that is not empty rather than reject the call. In addition, the system supports the use of steering pool ports from a parent realm by flows in a child realm. This adds an additional option for establishing allocation flexibility.

Consider a realm with three steering pools, including a **mixed**, a **single** and a **pair**. System port allocation behavior would include:

- The SBC directs MSRP sessions to the **single** pool. If the **single** pool runs out of ports, the SBC directs MSRP sessions to the **mixed** pool. If both of these pools are exhausted, the SBC drops new MSRP sessions and increments the 'no ports' counter.
- The SBC directs audio sessions to the **pair** pool. If the **pair** runs out of ports, the SBC directs audio sessions to the **mixed** pool. If both of these pools are exhausted, the SBC drops new audio sessions and increments the 'no ports' counter.

Notice that the SBC restricts traffic types when using the **single** and **pair** allocation strategies. The **mixed** pool, however, supports both traffic types.

To mitigate against the lack of consecutive ports described above, you could configure two **steering-pool** objects for same realm, one set to **pair** and the other to **single**. This configuration also restricts port utilization to specific pools by not including any **steering-pool** set to **mixed**.

Ultimately, you evaluate and monitor your deployment's needs for pool allocation and configure pools to suit those needs.

Configuration

You configure this functionality using the **port-allocation-strategy** parameter within **steering-pool** elements. The default is **mixed**.

```
ORACLE(steering-pool)#port-allocation-strategy single
```

Important configuration detail includes:

- Do not overlap port ranges configured on the same interface regardless of allocation strategy. The system throws a **verify-config** error identifying any overlapping **steering-pool** port range configuration, but it does not prevent you from saving and activating the configuration.
- The **port-allocation-strategy** parameter is real time configurable, allowing you to change an allocation strategy during operation. Established calls using this pool's strategy persist until the users terminate them. Ensuing calls use your updated strategy.
- The SBC selects ports from a child realm's pools when sessions start on that child realm. If a child realm's ports are exhausted, the SBC can allocate ports from the parent realm as long as the strategy is appropriate.

Steering Pool Allocation Examples

When determining the best port allocation strategies for your deployment, consider the examples in this section.

Basic Example

This example presents a realm with 3 **steering-pool** elements configured to the following strategies:

- SP1—**mixed**
- SP2—**single**
- SP3— **pair**

When MSRP calls arrive at this realm, the system allocates ports from SP2. If SP2 becomes exhausted, the system allocates ports for new MSRP calls from SP1. If both SP2 and SP1 become exhausted, the system rejects MSRP calls to this realm.

Similarly, when audio calls arrive at this realm, the system allocates ports from SP3. If SP3 becomes exhausted, the system allocates ports for new audio calls from SP1. If both SP3 and SP1 become exhausted, the system rejects audio calls to this realm.

Configuration Change Example

When a call is established with a steering pool port, configuration changes do not affect that call. After a configuration change, however, the system uses that port's updated strategy for ensuing allocations.

This example presents a realm with 3 **steering-pool** elements configured to the following strategies:

- SP1—**mixed**
- SP2—**single**
- SP3— **mixed**

> **Note:**
>
> Port allocation to pools of the same type depends on the port range available. When it begins allocating ports from a pool, the system chooses the lowest port available. But calls terminate randomly, so the system picks the topmost port available in a pool on ensuing calls, which may not be the lowest port number available.

When MSRP calls arrive at this realm, the system allocates ports from SP2. If SP2 becomes exhausted, the system allocates ports for new MSRP calls from SP1 and SP3. If both SP1 and SP3 become exhausted, the system rejects MSRP calls to this realm and increments the no ports counter.

Similarly, when audio calls arrive at this realm, the system allocates ports from SP1 and SP3. If SP3 becomes exhausted, the system allocates ports for new audio calls from SP1. If both SP3 and SP1 become exhausted, the system rejects audio calls to this realm.

If you change the **port-allocation-strategy** of SP2 from **single** to **pair**, then perform the **save-config** and **activate config** commands, the system begins to allocate audio calls to SP2. With this new configuration, the system is able to assign audio calls to all three pools. If SP1, SP2 and SP3 become exhausted, the system rejects audio calls to this realm.

If MSRP calls arrive at this realm, the system allocates ports for these calls using SP1 and SP3. If SP1 and SP3 become exhausted, the system rejects MSRP calls to this realm

Parent/Child Realm Allocation Example 1

Note that the following example presents a typical port utilization scenario. The difference here is that the system is using the parent realm's ports. In this example, either the child realm has no steering pools, or all of the child realm's applicable ports are in use.

Case 1—Assume the parent realm has 3 steering-pools:

- SP1—**mixed**
- SP2—**single**
- SP3— **pair**

In this case, an MSRP call arriving at the child realm uses the parent realm's SP2. If SP2 runs out of ports, the system tries to use SP1. Audio calls to the child realm use the parent realm's SP3. If SP3 runs out of ports, the system again tries to use SP1.

Case 2—In this case, all of the parent realm pools are **mixed**. Here, both MSRP and audio calls to the child realm can use SP1, SP2 and SP3.

Parent/Child Realm Allocation Example 2

Similar to Parent/Child Realm Allocation Example 1, the system here uses a child realm's steering pool ports first, then begins to use the parent realm's ports when the child realm's ports are in use. Utilization detail still uses the mixed, single, pair strategy rules.

- Case 1—Assume a child realm has:
  - SP4—**mixed**
  - SP5—**mixed**

  If you have configured 200 ports, the system uses all of those ports before using parent realm ports.
  - Sub Case 1—Assume the parent realm has:

        * SP1—**mixed**
        * SP2—**single**
        * SP3—**pair**

        The system receives MSRP calls to the child realm, which then uses SP2 in the parent realm. If parent realm SP2 becomes exhausted, the system starts using SP1. Audio calls to the child realm use the parent realm's SP3 and then SP1 when SP3 is exhausted.

    - Sub Case 2—Assume all of the parent realm's pool strategies are **mixed**. In this case, all MSRP and audio calls to the child realm can use SP1, SP2 and SP3.

- Case 2—In this case, assume the parent realm uses the same allocation for Case 1, Sub-case 1 and the child realm has:
    - SP4—**mixed** - 100 ports
    - SP5—**single** - 80 ports
    - SP6—**pair** - 60 ports

    Consider the results when there are 180 MSRP calls to the child realm. The system handles the first 80 using SP5 ports and the next 100 using SP4 ports. When MSRP call number 181 arrives, the system uses the parents realm's SPs, not the child realm's SPs. Similarly, when all 60 **pair** ports are used, the system uses any available **mixed** ports from SP4, and then uses the parent realm SPs.

    The following cases assume the child realm's ports are all used.

    - Sub Case 1—Incoming MSRP calls to child realm use SP2, in the parent realm, for port allocation. If SP2 ports are used, the system uses SP1. Audio calls to child realm use SP3, in the parent realm, for port allocation. If SP3 ports are used, the system uses SP1.
    - Sub Case 2—If the parent realm's pools are all **mixed**, all calls to child realm can use SP1, SP2 and SP3.
    - Sub Case 3—If the parent realm's pools are all **single**, the system terminates all audio calls to the child realm, and indicates that no ports are available.
    - Sub Case 4—If the parent realm's pools are all **pair**, the system terminates all MSRP calls to the child realm, and indicates that no ports are available.

Monitoring Port Allocation

You can troubleshoot your port allocation configuration and report on port usage, including the number of audio, MSRP calls and ports assigned using following commands:

- show sipd
- show mbcd
- show mbcd realms
- show mbcd realms <identifier>
- show mbcd realms <identifier> detailed

You can see status on **steering-pool** ports using the **show mbcd realm <identifier> detailed** command. On a per-realm basis, this command displays used and free ports for each steering pool whether configured as **mixed**, **single** or **pair**. This command also displays the 'no ports'

counter. This command allows you to monitor port utilization on a realm and adjust your allocation configuration based on your traffic.

```
ORACLE#show mbcd realm Realm172 detailed
                -- Period --       -- Lifetime --
               Active High Total  Total PerMax High
Ports Used        0    0     0      0     0     0
Free Ports        0    0     0      0     0     0
No Ports Avail    -    -     0      0     0     -
Ingress Band      OK   OK    0      0     0    OK
Egress Band       OK   OK    0      0     0    OK
Ingr Pri Band     OK   OK    0      0     0    OK
Egr Pri Band      OK   OK    0      0     0    OK
BW Allocations    0    0     0      0     0     0
Band Not Avail    -    -     0      0     0     -
Icmp Sent         -    -     0      0     0     -
Icmp Received     -    -     0      0     0     -
Total Bandwidth=0K

                -- Period --       -- Lifetime --
               Active High Total  Total PerMax High
-- MIXED PORTS --
Ports Used        0    0     0      0     0     0
Free Ports        0    0     0      0     0     0
-- SINGLE PORTS --
Ports Used        0    0     0      0     0     0
Free Ports        0    0     0      0     0     0
-- PAIRED PORTS --
Ports Used        0    0     0      0     0     0
Free Ports        0    0     0      0     0     0
```

Steering Pool Configuration

To configure a steering pool:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ```

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

   ```
   ORACLE(media-manager)# steering-pool
   ORACLE(steering-pool)#
   ```

4. **ip-address**—Enter the target IP address of the steering pool in IP address format. For example:

   ```
   192.168.0.11
   ```

5. **start-port**—Enter the start port value that begins the range of ports available to this steering pool. The default is **0**. The valid range is:
   - Minimum—1
   - Maximum—65535
     You must enter a valid port number or the steering pool will not function properly.
6. **end-port**—Enter the end port value that ends the range of ports available to this steering pool. The default is **0**. The valid range is:
   - Minimum—1
   - Maximum—65535
     You must enter a valid port number or the steering pool will not function properly.
7. **realm-id**—Enter the realm ID to identify the steering pool's realm, following the name format. The value you enter here must correspond to the value you entered as the identifier (name of the realm) when you configured the realm. For example:

   ```
   peer-1
   ```

   This steering pool is restricted to flows that originate from this realm.
8. **network-interface** —Enter the name of network interface this steering pool directs its media toward. A valid value for this parameter must match a configured name parameter in the network-interface configuration element. This parameter applies only to realms with multiple interfaces.
9. **port-allocation-strategy** —Select the appropriate strategy for this steering pool based on media type support in this realm. You can create multiple steering pools using different strategies to support multiple media types, per your deployment. Settings include:
   - mixed—(Default)
   - pair—The system only allocates these ports for calls that require multiple ports
   - single—The system only allocates these ports for calls that require a single port

   The following example shows the configuration of a steering pool.

   ```
   steering-pool
           ip-address                  192.168.0.11
           start-port                  20000
           end-port                    21000
           realm-id                    peer-1
           network-interface
           port-allocation-strategy    mixed
           last-modified-date          2005-03-04 00:35:22
   ```

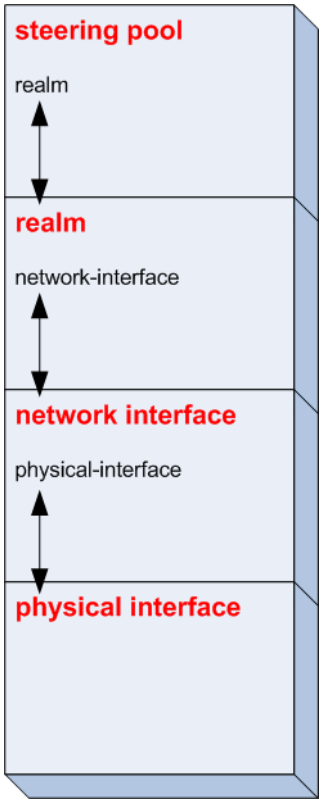
Multiple Interface Realms

The multi-interface realm feature lets you group multiple network interfaces to aggregate their bandwidth for media flows. In effect, this feature lets you use the total throughput of the available phy-interfaces on your Oracle Communications Session Border Controller for a single realm. Multi-interface realms are implemented by creating multiple steering pools, each on an individual network interface, that all reference a single realm.

> **Note:**
>
> Labels that read 'physical interface' in the diagrams below should be understood to reference the **phy-interface** configuration element.

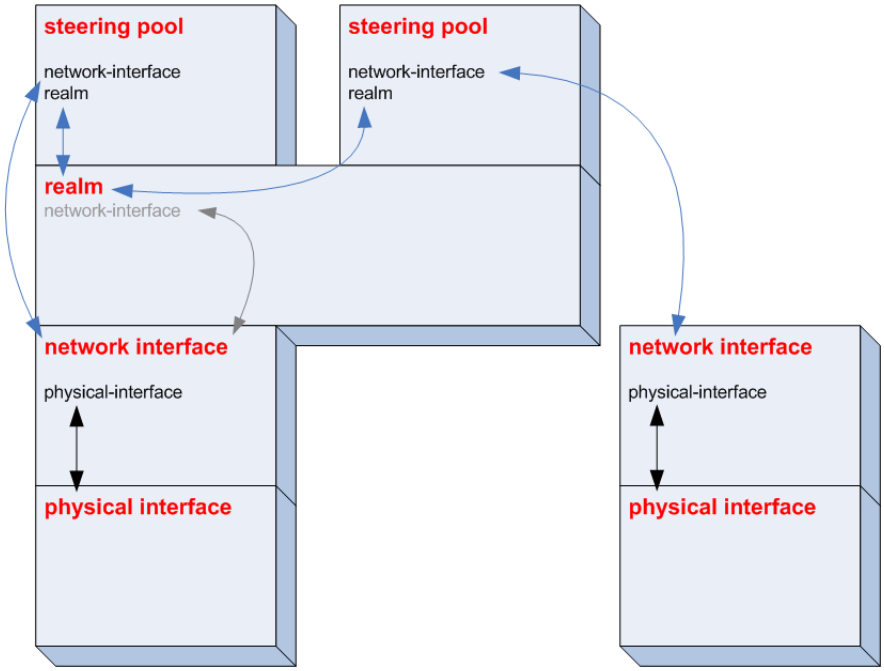
Of course, you can not to use this feature and configure your Oracle Communications Session Border Controller to create a standard one-realm to one-network interface configuration.

Without using multiple interface realms, the basic hierarchical configuration of the Oracle Communications Session Border Controller from the phy-interface through the media steering pool looks like this:

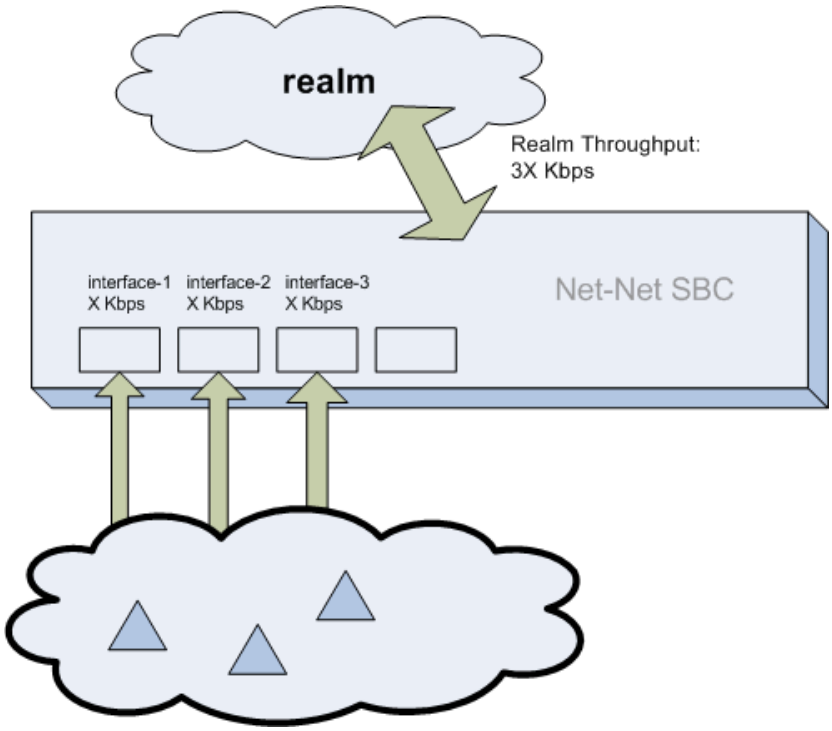


In this model, one (non-channelized) network interface exists on a phy-interface. One realm exists on one network interface. One or more steering pools can exist on one realm. Within each higher level configuration element exists a parameter that references a lower level configuration element in the Oracle Communications Session Border Controller's logical network model.

The multi-interface realm feature directs media traffic entering and exiting multiple network interfaces in and out of a single realm. Since all the steering pools belong to the same realm, their assigned network interfaces all feed into the same realm as well. The following diagram shows the relationship in the new logical model:



The advantage of using multi-interface realms is the ability to aggregate the bandwidth available to multiple network interfaces for a larger-than-previously-available total bandwidth for a realm. In the illustration below, three phy-interfaces each have X Kbps of bandwidth. The total bandwidth available to the realm with multiple network interfaces is now 3X the bandwidth. (In practical usage, interface-1 only contributes X - VoIP Signaling to the total media bandwidth available into the realm.)



Steering Pool Port Allocation

Every steering pool you create includes its own range of ports for media flows. The total number of ports in all the steering pools that feed into one realm are available for calls in and out of the realm.

Steering pool ports for a given realm are assigned to media flows sequentially. When the first call enters the Oracle Communications Session Border Controller after start-up, it is assigned the first ports on the first steering pool that you configured. New calls are assigned to ports sequentially in the first steering pool. When all ports from the first steering pool are exhausted, the Oracle Communications Session Border Controller uses ports from the next configured steering pool. This continues until the last port on the last configured steering pool is used.

After the final port is used for the first time, the next port chosen is the one first returned as empty from the full list of ports in all the steering pools. As media flows are terminated, the ports they used are returned to the realm's full steering pool. In this way, after initially exhausting all ports, the realm takes new, returned, ports from the pool in a least last used manner.

When a call enters the Oracle Communications Session Border Controller, the signaling application allocates a port from all of the eligible steering pools that will be used for the call. Once a port is chosen, the Oracle Communications Session Border Controller checks if the steering pool that the port is from has a defined network interface. If it does, the call is set up on the corresponding network interface. If a network interface is not defined for that steering pool, the network interface defined for the realm is used.

Network Interface Configuration

This section explains how to configure your Oracle Communications Session Border Controller to use multiple interface realms.

You must first configure multiple phy-interfaces and multiple network interfaces on your Oracle Communications Session Border Controller.

To configure the realm configuration for multi-interface realms.

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-manager path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes.

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

   From this point, you can configure a realm that will span multiple network interfaces.

4. **network-interfaces**—Enter the name of the network interface where the signaling traffic for this realm will be received.

Creating Steering Pools for Multiple Interface Realms

To configure steering pools for multi-interface realms:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-manager path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **steering-pool** and press Enter. The system prompt changes.

   ```
   ORACLE(media-manager)# steering-pool
   ORACLE(steering-pool)#
   ```

   From this point, you can configure steering pools which collectively bridge the multiple network interfaces they are connected to.

4. **ip-address**—Enter the IP address of the first steering pool on the first network interface.

   This IP address must correspond to an IP address within the subnet of a network interface you have already configured.

   This IP can not exist on a network interface other than the one you configure in the **network-interface** parameter.

5. **start-port**—Enter the beginning port number of the port range for this steering pool. The default is **0**. The valid range is:

   - Minimum—0
   - Maximum—65535

6. **end-port**—Enter the ending port number of the port range for this steering pool. The default is **0**. The valid range is:

   - Minimum—0
   - Maximum—65535

7. **realm-id**—Enter the name of the realm which this steering pool directs its media traffic toward.

8. **network-interface**—Enter the name of the network interface you want this steering pool to direct its media toward. This parameter will match a **name** parameter in the network-interface configuration element. If you do not configure this parameter, you can only assign a realm to a single network interface, as the behavior was in all Oracle Communications Session Border Controller Software releases pre- 2.1.

9. Create additional steering pools on this and on other network interfaces as needed. Remember to type **done** when you are finished configuring each new steering pool.

Media over TCP

The Oracle Communications Session Border Controller now supports RFC 4145 (TCP-Based Media Transport in the SDP), also called TCP Bearer support. Media over TCP can be used to support applications that use TCP for bearer path transport.

RFC 4145 adds two new attributes, setup and connection, to SDP messages. The setup attribute indicates which end of the TCP connection should initiate the connection. The connection attribute indicates whether an existing TCP connection should be used or if a new TCP connection should be setup during re-negotiation. RFC 4145 follows the offer/answer model specified in RFC3264. An example of the SDP offer message from the end point 192.0.2.2 as per RFC4145 is as given below:

```
m=image 54111 TCP t38
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
```

This offer message indicates the availability of t38 fax session at port 54111 which runs over TCP. Oracle Communications Session Border Controller does not take an active part in the application-layer communication between each endpoint.

The Oracle Communications Session Border Controller provides the means to set up the end-to-end TCP flow by creating the TCP/IP path based on the information learned in the SDP offer/answer process.

TCP Bearer Conditions

The following conditions are applicable to the Oracle Communications Session Border Controller's support of RFC 4145.

1. The Oracle Communications Session Border Controller can not provide media-over-TCP for HNT scenarios (endpoints behind a NAT).
2. When media is released into the network, the TCP packets do not traverse the Oracle Communications Session Border Controller because nNo TCP bearer connection is created.
3. The Oracle Communications Session Border Controller does not inspect the setup and connection attributes in the SDP message since the TCP packets transparently pass through the Oracle Communications Session Border Controller. These SDP attributes are forwarded to the other endpoint. It is the other endpoint's responsibility to act accordingly.
4. After the Oracle Communications Session Border Controllerreceives a SYN packet, it acts as a pure pass through for that TCP connection and ignores all further TCP handshake messages including FIN and RST. The flow will only be torn down in the following instances:
   - The TCP initial guard timer, TCP subsequent guard timer, or the TCP flow time limit timer expire for that flow.
   - The whole SIP session is torn down.

TCP Port Selection

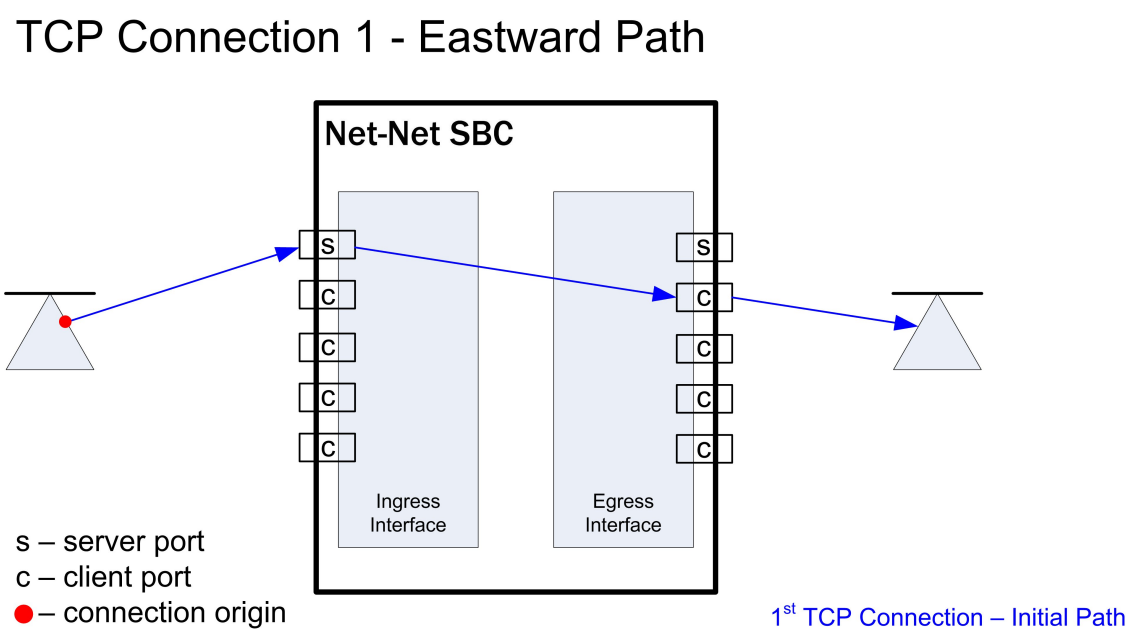
When a call is first set up, the Oracle Communications Session Border Controller inspects the SDP message's m-line to see if any media will be transported via TCP. If the SDP message indicates that some content will use TCP, the Oracle Communications Session Border Controller allocates a configured number of steering ports for the media-over-TCP traffic. These TCP media ports are taken from the each realm's steering pool.

Each endpoint can initiate up to four end-to-end TCP flows between itself and the other endpoint. The Oracle Communications Session Border Controller assigns one port to receive the initial TCP packet (server port), and one to four ports assigned to send TCP traffic (client

ports) to the receiving side of the TCP flow. The number of TCP flows for each call is configured globally.

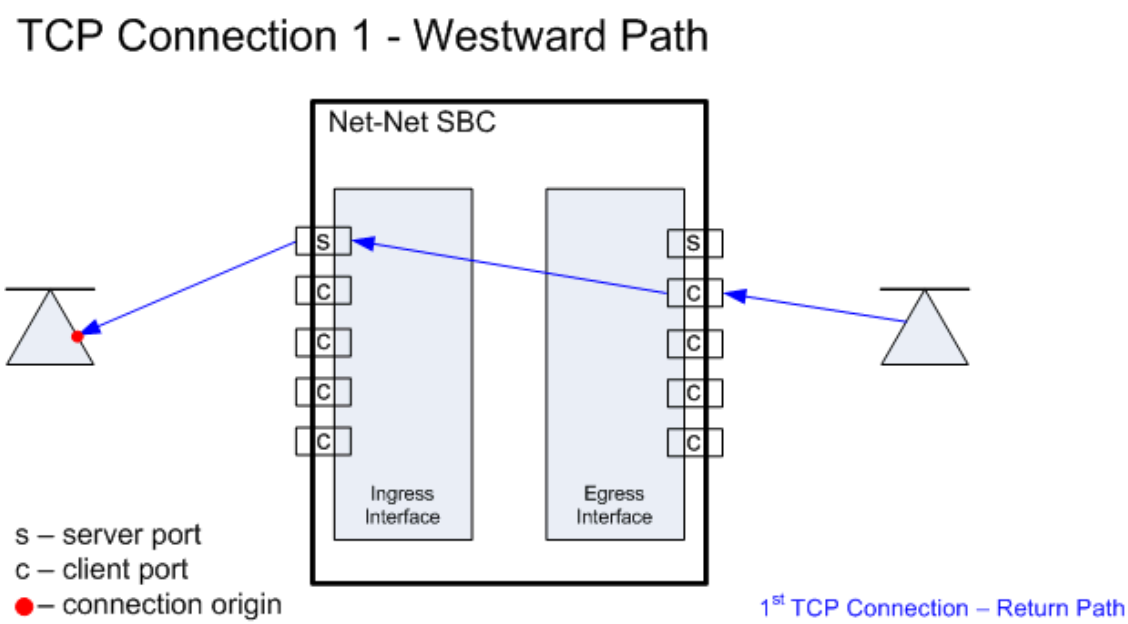
In order to configure the Oracle Communications Session Border Controller to facilitate and support this process, you need to specify the number of ports per side of the call that can transport discrete TCP flows. You can configure one to four ports/flows. For configuration purposes, the Oracle Communications Session Border Controller counts this number as inclusive of the server port. Therefore if you want the Oracle Communications Session Border Controller to provide a maximum of one end-to-end TCP flow, you have to configure two TCP ports; one to receive, and one to send. The receiving port (server) is reused to set up every flow, but the sending port (client) is discrete per flow. For example: for 2 flows in each direction, set the configuration to 3 TCP ports per flow; for 3 flows in each direction, set the configuration to 4 TCP ports per flow, etc.

The server port is used for initiating a new TCP connection. An endpoint sends the first packet to a server port on the ingress interface. The packet is forwarded out of the Oracle Communications Session Border Controller through a client port on the egress interface toward an endpoint:

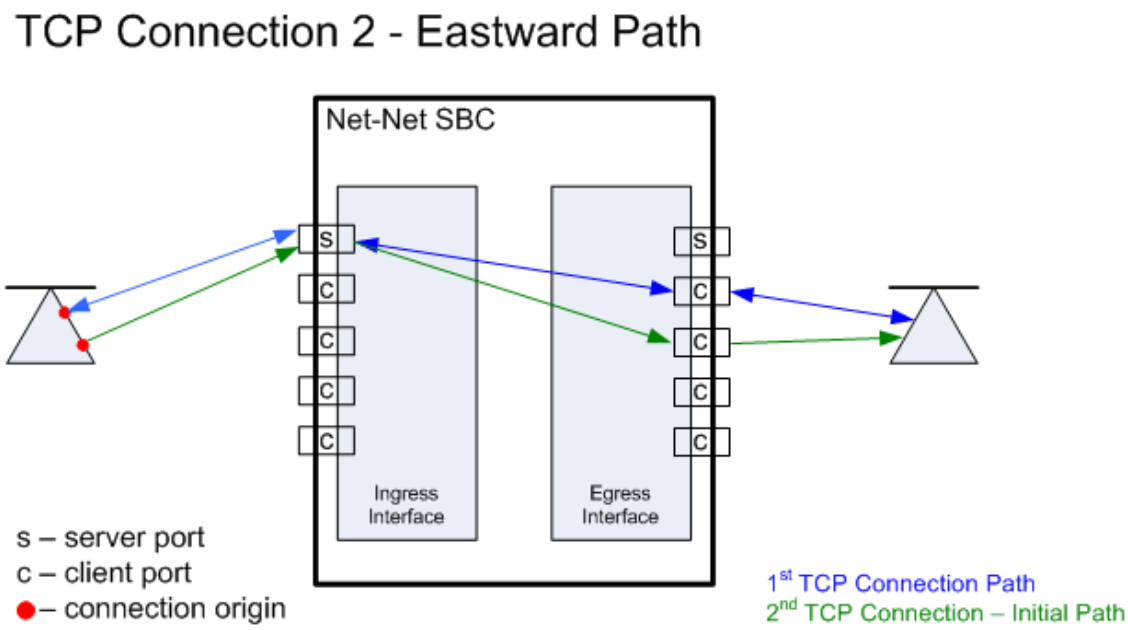


1st TCP Connection – Initial Path

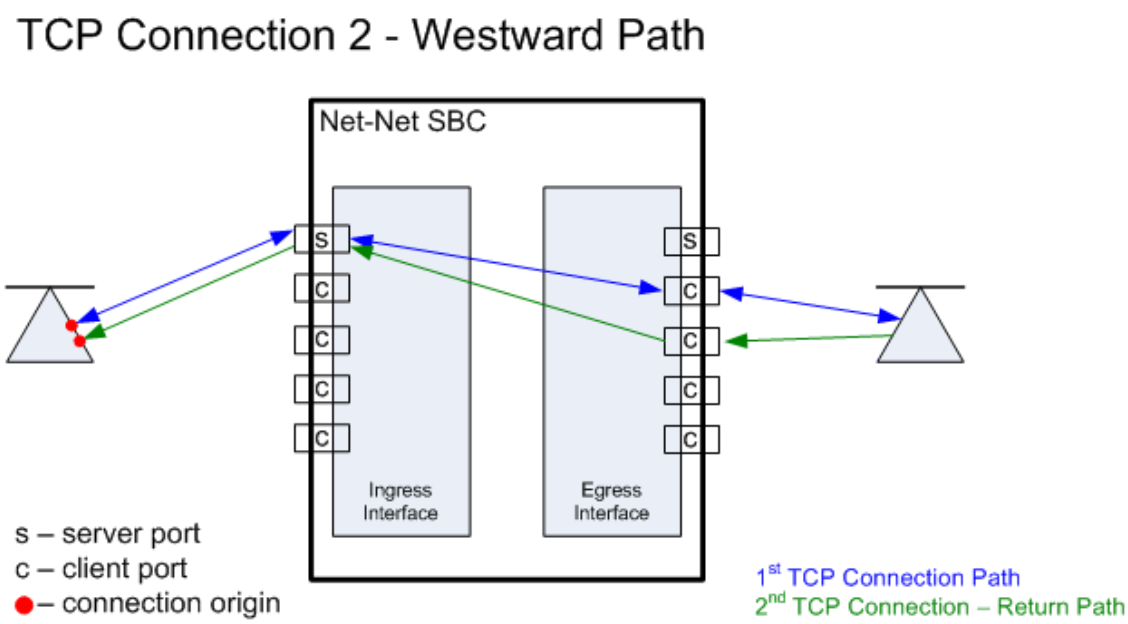
The endpoint responds back to the client port on the egress interface. This message traverses the Oracle Communications Session Border Controller and is forwarded out of the server port on the ingress interface where the initial packet was sent. The remainder of the TCP flow uses the server and client port pair as a tunnel through the Oracle Communications Session Border Controller:



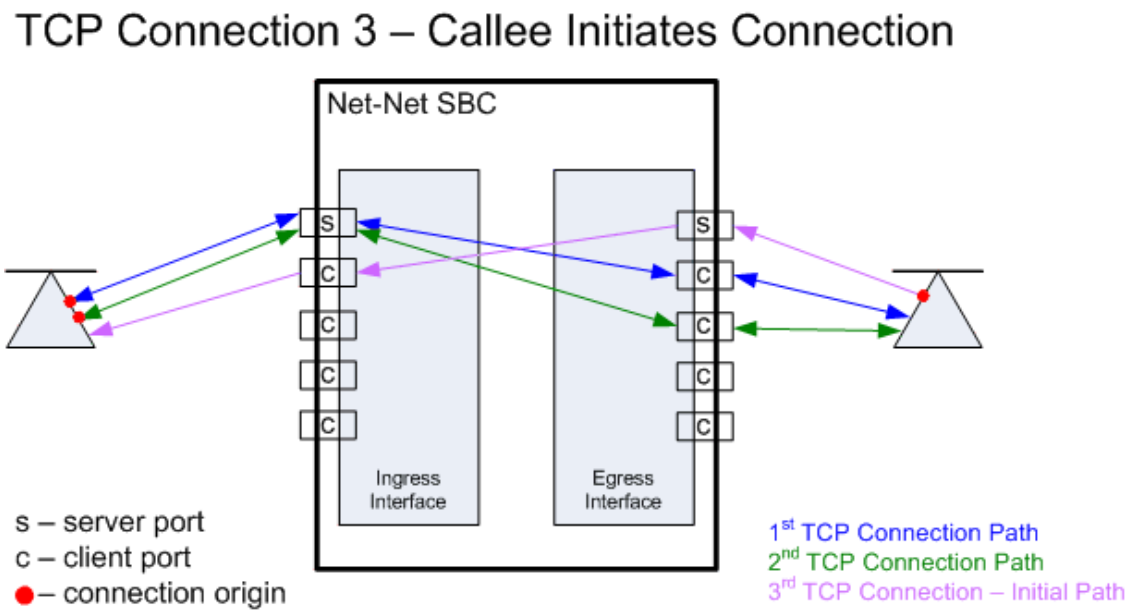
When the second TCP connection is set up in the same direction as in the first example, the first packet is still received on the server port of the ingress interface. The next unused client port is chosen for the packet to exit the Oracle Communications Session Border Controller:



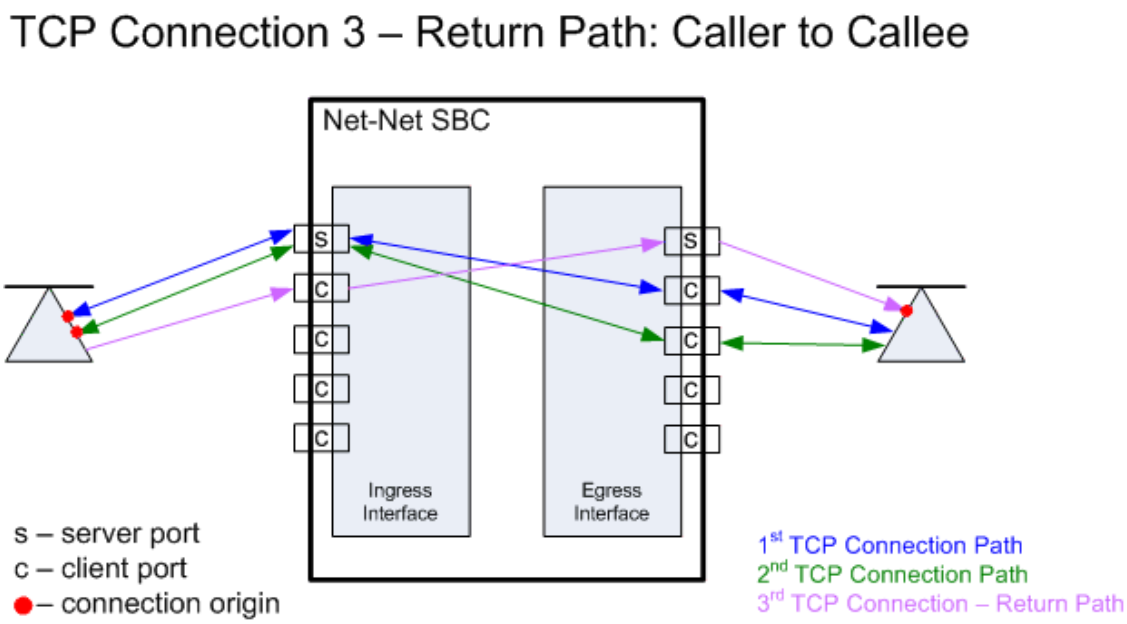
The response takes the same path back to the caller. The remainder of the second TCP connection uses this established path:



When the callee initiates a TCP connection, it must send its initial traffic to the server port on its Oracle Communications Session Border Controller ingress interface. The packet is forwarded out of the first free client port on the egress side of this TCP connection toward the caller.



The caller's response takes the same path back to the callee that initiated this TCP connection. The remainder of the third TCP connection uses this established path.

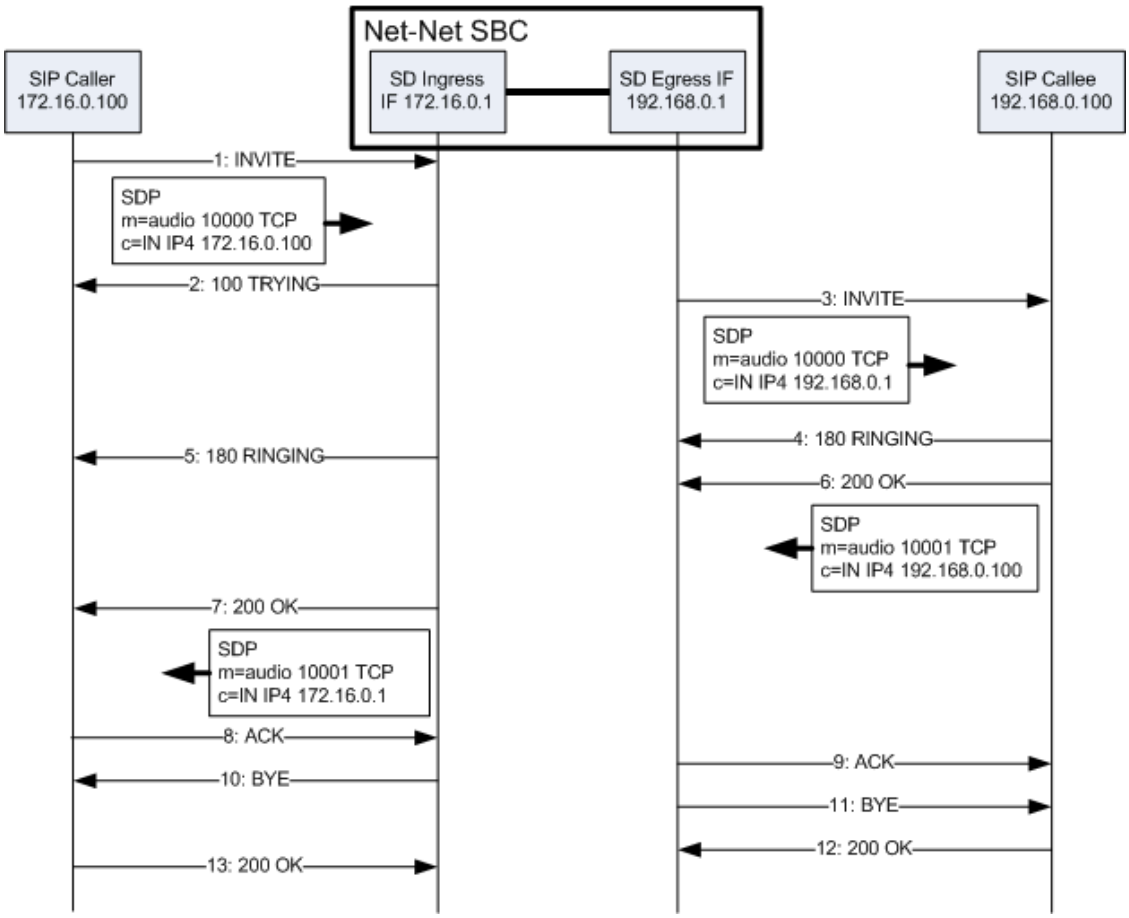


The Oracle Communications Session Border Controller can support a total of eight media-over-TCP connections per call. A maximum of 4 connections are supported as initiated from each side of the call.

SDP Offer Example

The following abbreviated call flow diagram sets up a media-over-TCP flow. Observe that the caller listens for audio over TCP on 172.16.0.10:10000, as described in the SDP offer (1). The Oracle Communications Session Border Controller re-writes the m and c lines in the SDP offer to reflect that it is listening for audio over TCP on its egress interface at 192.168.0.1:10000 (3). The Oracle Communications Session Border Controller then forwards the SIP invite to the callee.

The SIP callee responds with an SDP answer in a 200 OK message. The callee indicates it is listening for the audio over TCP media on 192.168.0.10:10001 (6). The Oracle Communications Session Border Controller re-writes the m and c lines in the SDP answer to reflect that it is listening for audio over TCP on the call's ingress interface at 172.16.0.1:10001 (7). The Oracle Communications Session Border Controller then forwards the SIP invite to the caller.



All interfaces involved with the end-to-end TCP flow have now established their listening IP address and port pairs.

Timers

The Oracle Communications Session Border Controller has three guard timers that ensure a TCP media flow does not remain connected longer than configured. You can set each of these from 0 (disabled) to 999999999 in seconds.

- TCP initial guard timer — Sets the maximum time in seconds allowed to elapse between the initial SYN packet and the next packet in this flow.
- TCP subsequent guard timer — Sets the maximum time in seconds allowed to elapse between all subsequent sequential TCP packets.
- TCP flow time limit — Sets the maximum time that a single TCP flow can last. This does not refer to the entire call.

TCP Port Configuration

To configure media over TCP:

1. Access the **media-manager-config** configuration element.

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

```
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

2. **tcp-number-of-ports-per-flow**—Enter the number of ports, inclusive of the server port, to use for media over TCP. The total number of supported flows is this value minus one. The default is **2**. The valid range is:
   - Minimum—2
   - Maximum—4

   ```
   ORACLE(media-manager-config)# tcp-number-of-ports-per-flow 5
   ```

3. **tcp-initial-guard-timer**—Enter the maximum time in seconds allowed to elapse between the initial SYN packet and the next packet in a media-over-TCP flow. The default is **300**. The valid range is:
   - Minimum—0
   - Maximum—999999999

   ```
   ORACLE(media-manager-config)# tcp-initial-guard-timer 300
   ```

4. **tcp-subsq-guard-timer**—Enter the maximum time in seconds allowed to elapse between all subsequent sequential media-over-TPC packets. The default is **300**.
   - Minimum—0
   - Maximum—999999999

   ```
   ORACLE(media-manager-config)# tcp-subsq-guard-timer 300
   ```

5. **tcp-flow-time-limit**—Enter the maximum time in seconds that a media-over-TCP flow can last. The default is **86400**. The valid range is:
   - Minimum—0
   - Maximum—999999999

   ```
   ORACLE(media-manager-config)# tcp-flow-time-limit 86400
   ```

Restricted Media Latching

The restricted media latching feature lets the Oracle Communications Session Border Controller latch only to media from a known source IP address, in order to learn and latch the dynamic UDP port number. The restricting IP address's origin can be either the SDP information or the SIP message's Layer 3 (L3) IP address, depending on the configuration.

About Latching

Latching is when the Oracle Communications Session Border Controller listens for the first RTP packet from any source address/port for the destination address/port of the Oracle Communications Session Border Controller. The destination address/port is allocated dynamically and sent in the SDP. After it receives a RTP packet for that allocated destination address/port, the Oracle Communications Session Border Controller only allows subsequent RTP packets from that same source address/port for that particular Oracle Communications Session Border Controller destination address/port. Latching does not imply that the latched

source address/port is used for the destination of the reverse direction RTP packet flow (it does not imply the Oracle Communications Session Border Controller will perform symmetric RTP).

Restricted Latching

The Oracle Communications Session Border Controller restricts latching of RTP/RTCP media for all calls within a realm. It latches to media based on one of the following:

- SDP: the IP address and address range based on the received SDP c= connect address line in the offer and answer.
- Layer 3: the IP address and address range based on the received L3 IP address of the offer or answer. This option is for access registered HNT endpoints. If the L3 IP address is locally known and cached by the Oracle Communications Session Border Controller as the public SIP contact address, that information could be used instead of waiting for a response. The Oracle Communications Session Border Controller might use the L3 IP address restriction method for all calls regardless of whether the endpoint is behind a NAT or not, for the same realms.

Symmetric Latching

A mode where a device's source address/ports for the RTP/RTCP it sends to the Oracle Communications Session Border Controller (SBC) that are latched, are then used for the destination of RTP/RTCP sent to the device.

After allocating the media session in SIP, the SBC sets the restriction mode and the restriction mask for the calling side as well as for the called side. It sets the source address and address prefix bits in the flow. It also parses and loads the source flow address into the MIBOCO messages. After receiving the calling SDP, the SBC sets the source address (address and address prefix) in the appropriate flow (the flow going from calling side to the called side). After receiving the SDP from the called side, the SBC sets the source address in the flow going from the called side to the calling side.

The SBC uses either the address provided in the SDP or the layer 3 signaling address for latching. You also configure the SBC to enable latching so that when it receives the source flow address, it sets the address and prefix in the NAT flow. When the NAT entry is installed, all the values are set correctly. In addition, sipd sends the information for both the incoming and outgoing flows. After receiving SDP from the called side sipd, the SBC sends information for both flows to the MBCD so that the correct NAT entries are installed.

Enabling restricted latching may make the SBC wait for a SIP/SDP response before latching, if the answerer is in a restricted latching realm. This is necessary because the SBC does not usually know what to restrict latching to until the media endpoint is reached. The only exception could be when the endpoint's contact/IP is cached.

Relationship to Symmetric Latching

The current forced HNT symmetric latching feature lets the Oracle Communications Session Border Controller assume devices are behind NATs, regardless of their signaled IP/SIP/SDP layer addresses. The Oracle Communications Session Border Controller latches on any received RTP destined for the specific IP address/port of the Oracle Communications Session Border Controller for the call, and uses the latched source address/port for the reverse flow destination information.

If both restricted latching and symmetric latching are enabled, the Oracle Communications Session Border Controller only latches if the source matches the restriction, and the reverse

flow will only go to the address/port latched to, and thus the reverse flow will only go to an address of the same restriction.

- Symmetric latching is enabled.
  If symmetric latching is enabled, the Oracle Communications Session Border Controller sends the media in the opposite direction to the same IP and port, after it latches to the source address of the media packet.
- Symmetric latching is disabled.
  If symmetric latching is disabled, the Oracle Communications Session Border Controller only latches the incoming source. The destination of the media in the reverse direction is controlled by the SDP address.

Example 1

A typical example is when the Oracle Communications Session Border Controller performs HNT and non-HNT registration access for endpoints. Possibly the SDP might not be correct, specifically if the device is behind a NAT. Therefore the Oracle Communications Session Border Controller needs to learn the address for which to restrict the media latching, based on the L3 IP address. If the endpoint is not behind a NAT, then the SDP could be used instead if preferred. However, one can make some assumptions that access-type cases will require registration caching, and the cached fixed contact (the public FW address) could be used instead of waiting for any SDP response.

Example 2

Another example is when a VoIP service is provided using symmetric-latching. A B2BUA/proxy sits between HNT endpoints and the Oracle Communications Session Border Controller, and calls do not appear to be behind NATs from the Oracle Communications Session Border Controller's perspective. The Oracle Communications Session Border Controller's primary role, other than securing softswitches and media gateways, is to provide symmetric latching so that HNT media will work from the endpoints.

To ensure the Oracle Communications Session Border Controller's latching mechanism is restricted to the media from the endpoints when the SIP Via and Contact headers are the B2BUA/proxy addresses and not the endpoints', the endpoint's real (public) IP address in the SDP of the offer/answer is used. The B2BUA/proxy corrects the c= line of SDP to that of the endpoints' public FW address.

The Oracle Communications Session Border Controller would then restrict the latching to the address in the SDP of the offer from the access realm (for inbound calls) or the SDP answer (for outbound calls).

Restricted Latching using Address and Port

You can configure the system to latch all media flows within a realm to both the externally provided address and port when you set the **restricted-latching** mode to **sdp-ip-port**. When configured to this setting, the system latches to media based on the IP Address received in the SDP c= connect address line, and the port in the mline in the offer and answer. This differs from standard latching in that the port is left unassigned by the SBC. This feature allows the SBC to better support multiple RTP streams from different ports using the same IP address, such as within forking scenarios.

When configuring latching to **sdp-ip-port**, the SBC supports:

- Latching to IP and port within early media scenarios and call establishment phases

- Re-latching when an SDP update received in a 200 OK is different than the one received in a provisional response
- Re-latching to media based on IP and port within reINVITE/UPDATE scenarios, when the media is updated after call establishment including:
  - Updating the latching after the 200 OK when the media is updated by reINVITE/ UPDATE scenarios
  - Updating the latching within call transfer scenarios involving forking.
- Maintaining latching established during early media stages when the 200 OK does not include SDP
- Maintaining latching established by the most recent provisional response when the 200 OK does not include SDP

> **Note:**
>
> If you downgrade to a version that does not support this feature, the system changes the **restricted-latching** setting to **none**. Upgrades and downgrades to system with feature compatibility retain your setting.

Related Configuration

If you have enabled **rtcp-mux** in conjunction with this feature, the system installs un-collapsed flows for RTP and RTCP that include IP and port in both the east and west realms. This is true if you have enabled **rtcp-mux** and this feature on one or both of these realms. In these cases, the system installs these uncollapsed flows on both realms and supports only the following two port assignments for RTCP:

- RTP port
- RTP port + 1

The table below presents the way the SBC assigns and handles RTCP differently, based on **restricted-latching** configuration and RTCP input.

restricted-latching Configurations	**RTCP sent from UAC mline (RTP) even port to SBC RTP port (even port)**	**RTCP from UAC mline port (RTP) to SBC RTP+1 port (odd port)**	**RTCP from UAC mline port+1 (UAC odd port) to SBC RTP port (even port)**	**RTCP from UAC mline port+1 (odd port) SBC RTP+1 port (odd port)**
UAC realm— **sdp** UAS realm— **disabled** Media is sent from UAC	RTCP is forwarded to RTP port of the UAS via SBC egress even port	RTCP is forwarded to RTP+1 port towards UAS from egress odd (RTP+1) port of SBC	RTCP is forwarded to RTP even port of UAS from egress even SBC port	RTCP is forwarded to RTP+1 port towards UAS from SBC odd port (RTP+1)
UAC realm— **sdp-ip-port** UAS realm— **disabled** Collapsed flows installed	RTCP is forwarded to RTP port of UAS via SBC egress even port	RTCP is forwarded to RTP+1 port towards UAS from egress odd (RTP+1) port of SBC	RTCP is forwarded to RTP even port of UAS from egress even SBC port	RTCP is forwarded to RTP+1 port towards UAS from SBC odd port (RTP+1)

In addition, when you enable this feature the SBC installs fully qualified NAT flows. This effectively disables latching on the IP and port within RTP media. As a result, you should not

enable this feature in conjunction with features that require explicit RTP packet based latching. For example, this feature effectively overrides dynamic-latching.

Reporting on the Feature

For UAC to UAS call when the feature is enabled on both the realms, the **show nat in-tabular** command displays the NAT flow similar to the output below. Note the assigment of port 0 in the first output below, In this case, **restricted-latching** is set to **sdp**.

```
ORACLE# show nat in-tabular
Index   Prot    Intf:Vlan  Src IP:Port                  Dst IP:Port
-------------------------------------------------------------------------------
2       udp     I=0/0:300  192.168.204.100:0       192.168.204.124:10000
                O=0/0:100  172.16.204.124:10000    172.16.204.100:11000
3       udp     I=0/0:100  172.16.204.100:0        172.16.204.124:10000
                O=0/0:300  192.168.204.124:10000   192.168.204.100:10000
```

Note the assigment of port 10000 instead of 0 in the second outut below, In this case, **restricted-latching** is set to **sdp-ip-port**.

```
ORACLE# show nat in-tabular
Index Prot    Intf:Vlan   Src IP:Port               Dst IP:Port
-------------------------------------------------------------------
2       udp   I=0/0:300  192.168.204.100:10000  192.168.204.124:10000
              O=0/0:100  172.16.204.124:10000   172.16.204.100:11000
3       udp   I=0/0:100  172.16.204.100:11000   172.16.204.124:10000
              O=0/0:300  192.168.204.124:10000  192.168.204.100:10000
```

The SBC installs these fully qualified flows on offer-answer completion, before it receives the RTP.

Call Flows Supporting Restricted Latching to Address and Port

A key problem resolved by latching on IP and port is the handling of multiple streams created by a single external device, such as an Enhanced Communications Network Application Server (ECN AS) forking an INVITE. Without this feature, the system implements restricted latching such that the latching is dependent on the external device, which may adversely impact which RTP stream (or streams) get played back to the caller.

Parallel Ringing before Answer

Consider a scenario wherein an ECN AS uses parallel forking to route a call via the SBC to several users. The ECN AS hides the multiple early dialogs from the SBC and forwards only one of them. There are, however, multiple RTP streams created from the Media Gateway (MGW) towards the SBC, one for each branch. The best way to setup such a call would result in the caller receiving only the RTP in the 18x SDP response from the ECN AS.

When you configure **restricted-latching** to **sdp-ip-port**, the SBC can also handle SDP changes, including RTP target and source. If the initial SDP becomes invalid within this forking scenario, the ECN AS would update the SBC with subsequent SDP. This update could come in an UPDATE with SDP or a new 18x with SDP, depending on the environment's support for 100rel/PRACK. The SBC would handle these changes, including changing the callee.

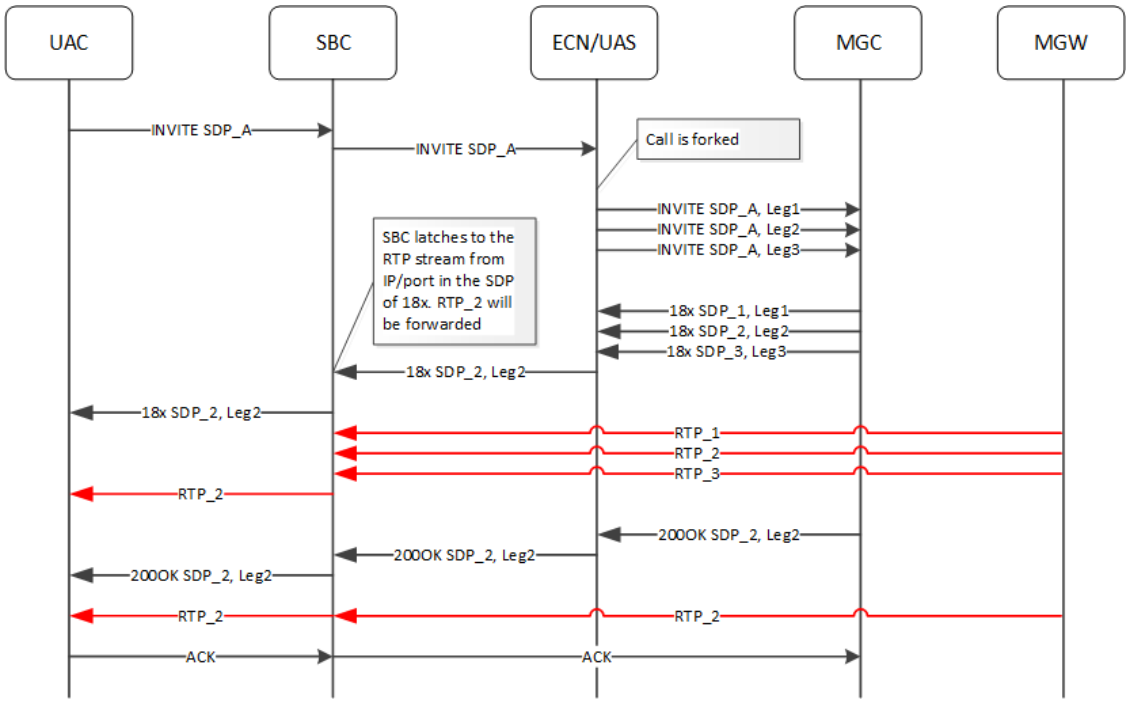
When you set **restricted-latching** to **sdp**, the system successfully blocks RTP streams from different addresses. But if there are multiple RTP streams coming from different ports on a single IP address, the system admits all of these. This can result in the calling party hearing a

mixed early media stream. When set to **sdp-ip-port**, however, the SBC latches based on the IP and port in the SDP from the ECN AS, thereby limiting the audible RTP to a single flow.

> **Note:**
>
> The AS updates the SDP only if a called party was selected during the answer phase other than the called party that was selected during alerting phase. Therefore, it is important that the SBC performs latching and re-latching based on IP address and port from the SDP. This way, the AS is also aware of which called party media stream is active.

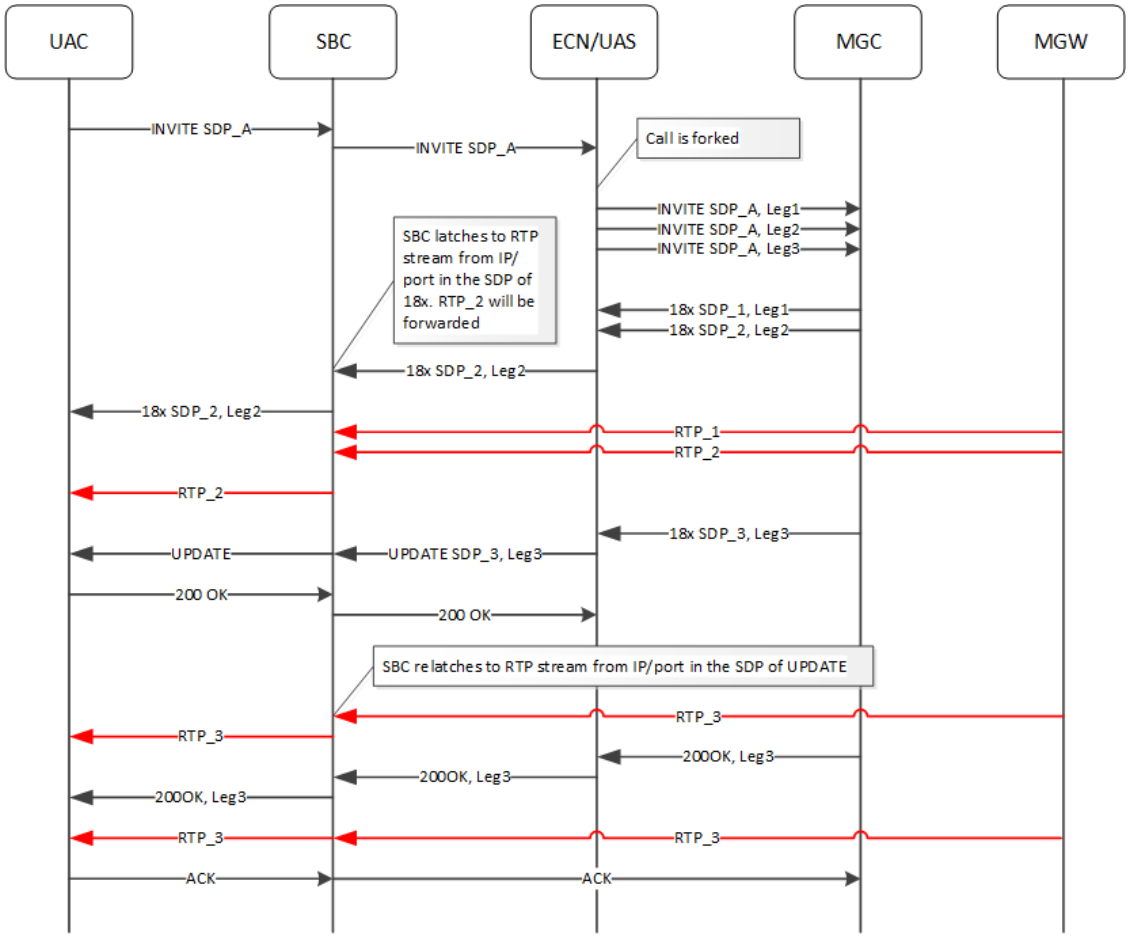
> **Note:**
>
> If the final 200 OK for the INVITE does not contain SDP, the SBC stays latched to the most recent IP/port received in a 18x during the early media stage.

In the call flow below, you have configured **restricted-latching** to **sdp-ip-port** on the UAS realm. The called stations could be behind the Media Gateway (MGW) interfacing the PSTN/PLMN) or multiple IP's, and served by a Media Gateway Controller (MGC).



Parallel Ringing and Re-Latching before Answer Based on an UPDATE

If an UPDATE comes from the ECN before the call is answered, the SBC is able to re-latch to a new RTP stream, as shown below. In the call flow, the SBC latches to SDP_2, forwarding only that RTP. The MGC, however, issues a 18x requesting SDP_3 before the call is answered. Upon receiving this UPDATE, the SBC re-latches to SDP_3. Subsequently, the call is answered and RTP_3 media proceeds without issue.



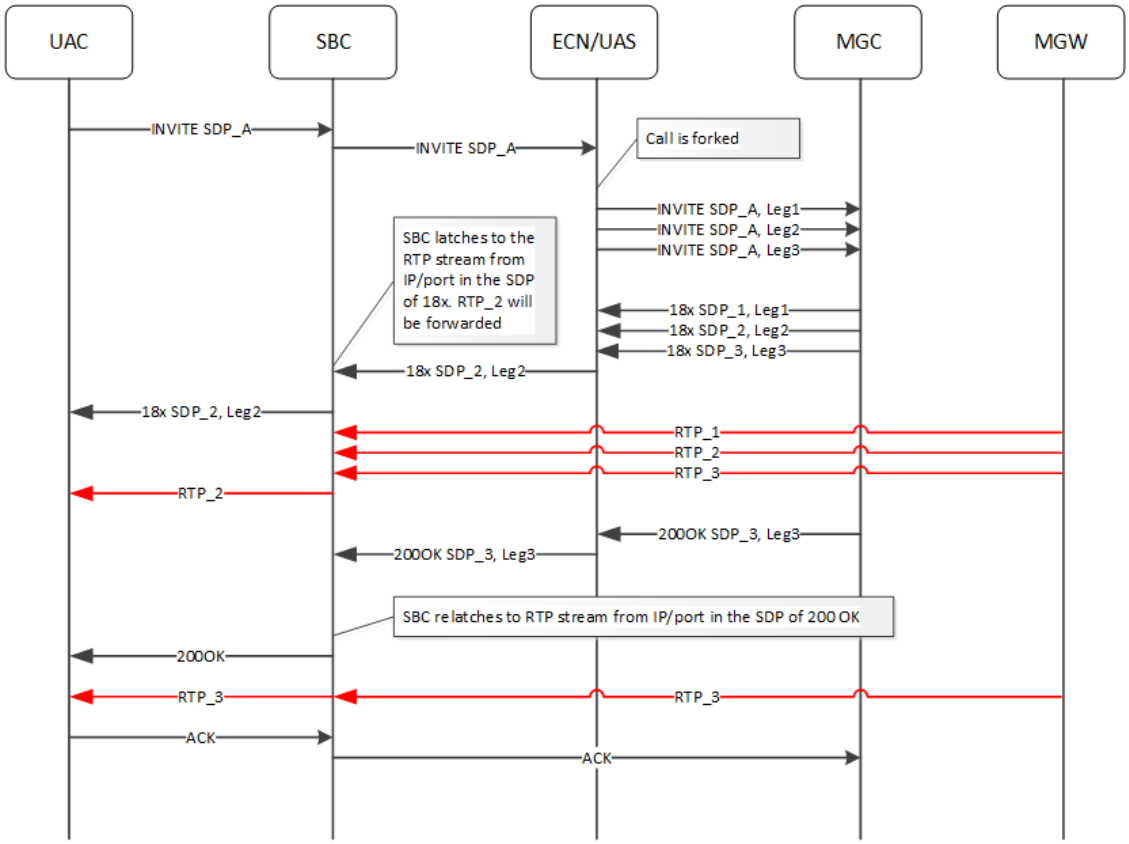
Latching during Answer

At the end of the signaling, the ECN AS sends a 200 OK that often has the SDP that the SBC must forward to the calling party. Without enhanced restricted latching, the SBC latches to the first RTP stream that arrives after the 200 OK. This RTP stream may not be the same RTP stream identified in the SDP of the 200 OK, resulting in silence at the caller.

When you set **restricted-latching** to **sdp**, the SBC admits any RTP stream from the IP in the SDP. When you set **restricted-latching** to **sdp-ip-port**, the SBC controls the latching using both IP and port received in the 200 OK SDP. This setting ensures that the correct RTP stream gets forwarded.

> **Note:**
>
> Note: If the SDP is not updated after the early media stages, the 200 OK may not include any SDP. Therefore, it is important that the AS knows the media stream onto which the SBC has latched. This requires enhanced restricted latching.

In the call flow below, you have configured **restricted-latching** to **sdp-ip-port** on the UAS realm. The called stations could be behind the Media Gateway (MGW) interfacing the PSTN/ PLMN) or multiple IP's, and served by a Media Gateway Controller (MGC).



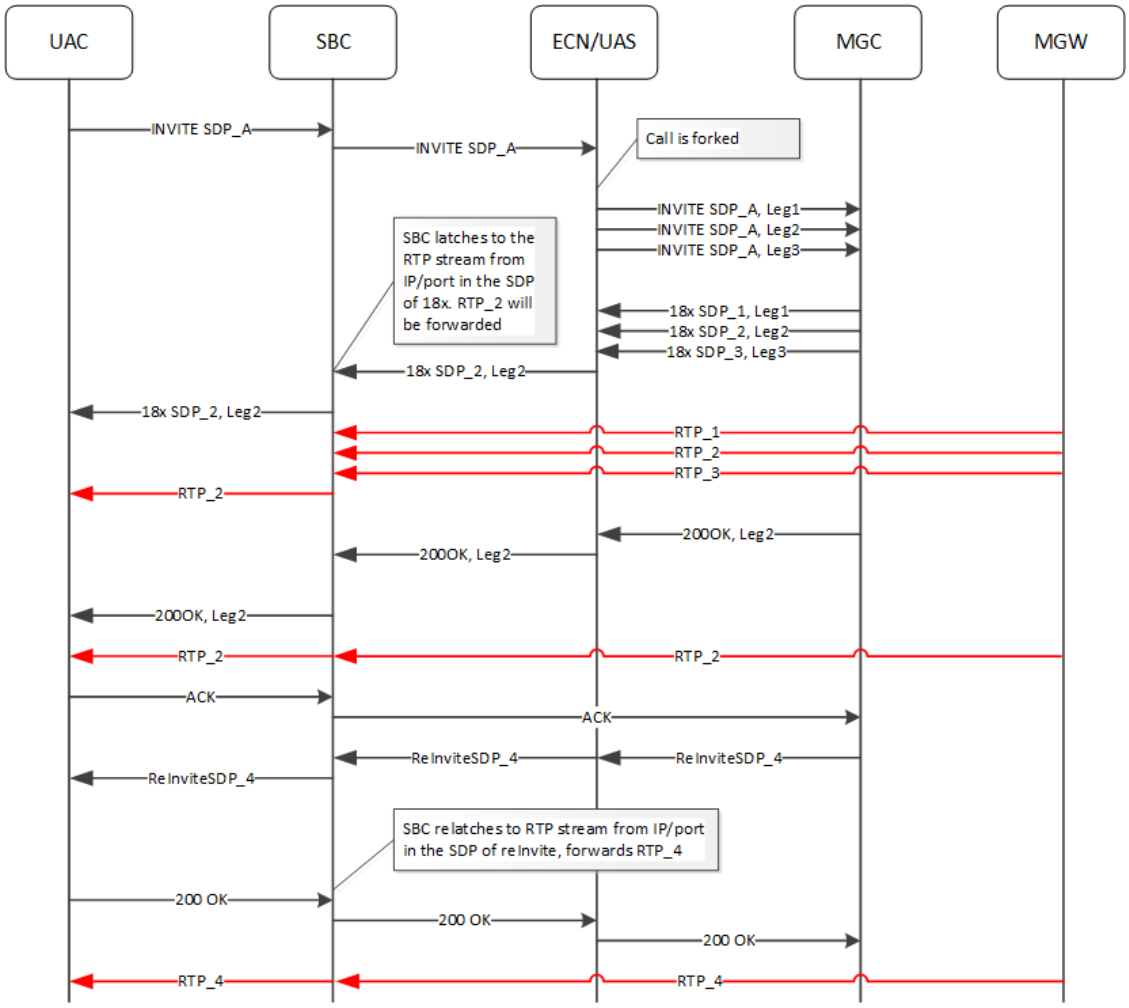
Latching after 200 OK in Case of reINVITE/UPDATE

Applicable scenarios include media re-negotiation after the call is answered are supported. These scenarios may happen when the ECN AS uses parallel forking to redirect or transfer the call after answer.

When you set **restricted-latching** to **sdp**, the SBC admits streams from the same IP, which could be multiple streams if multiple endpoints are behind the gateway. Subsequently, the SBC forwards all RTP streams it receives from, in this case, an MGW.

Depending on the calling party's client, the calling party would hear either a mix of ringback tones, silence, or a single ringback tone. When you set **restricted-latching** to **sdp-ip-port**, the SBC latches the media based on the IP and port in the reINVITE/UPDATE received after the call has been established. This results in the SBC forwarding only the RTP stream identified in the SDP from ECN AS to the calling party.

In the call flow below, you have configured **restricted-latching** to **sdp-ip-port** on the UAS realm. The called stations could be behind the Media Gateway (MGW) interfacing the PSTN/ PLMN) or multiple IP's, and served by a Media Gateway Controller (MGC).



Restricted Latching Configuration

To configure restricted latching:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

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

4. Select the realm where you want to apply this feature.

```
ORACLE(realm-config)# select
identifier:
1: Acme_Realm <none>           0.0.0.0
2: H323REALM  <none>           0.0.0.0
selection:1
ORACLE(realm-config)#
```

5. **restricted-latching**— Enter the restricted latching mode. The default is **none**. The valid values are:
   - **none**—No restricted-latching used
   - **sdp**—Use the address provided in the SDP for latching
   - **peer-ip**—Use the layer 3 signaling address for latching
   - **sdp-ip-port**—Latch to media based on the IP Address received in the SDP c= connect address line, and the port in the mline in the offer and answer.
6. **restriction-mask**— Enter the number of address bits you want used for the source latched address. This field will be used only if the restricted-latching is used. The default is **32**. When this value is used, the complete IP address is matched for IPv4 addresses. The valid range is:
   - Minimum—1
   - Maximum—128

Media Release Across SIP Network Interfaces

This feature lets the Oracle Communications Session Border Controller release media between two SIP peers, between two realms on two network interfaces of the same Oracle Communications Session Border Controller. Use this feature when you want the Oracle Communications Session Border Controller to release media for specific call flows, regardless of the attached media topology.

Media Release Configuration

To configure media release across network interfaces:

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

```
ORACLE# configure terminal
```

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

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

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

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

4. Select the realm where you want to apply this feature.

```
ORACLE(realm-config)# select
identifier:
1: Acme_Realm <none>           0.0.0.0
2: H323REALM  <none>           0.0.0.0
selection:1
ORACLE(realm-config)#
```

5. **mm-in-system**—Set this parameter to **enabled** to manage/latch/steer media in the Oracle Communications Session Border Controller. Set this parameter to **disabled** to release media in the Oracle Communications Session Border Controller.

> **Note:**
>
> Setting this parameter to disabled will cause the Oracle Communications Session Border Controller to NOT steer media through the system (no media flowing through this Oracle Communications Session Border Controller).

The default is **enabled**. The valid values are:

- enabled | disabled

Media Release Behind the Same IP Address

The media management behind the same IP feature lets the Oracle Communications Session Border Controller release media when two endpoints are behind the same IP address, in the same realm. Using this feature prevents the media for intra-site calls from going through the Oracle Communications Session Border Controller. You can use this feature for both hosted NAT traversal (HNT) and non-HNT clients. It works with NATed endpoints and for non-NATed ones that are behind the same IP.

Additional Media Management Options

Additional media management options include:

- Media directed between sources and destinations within this realm on this specific Oracle Communications Session Border Controller. Media travels through the Oracle Communications Session Border Controller rather than straight between the endpoints.
- Media directed through the Oracle Communications Session Border Controller between endpoints that are in different realms, but share the same subnet.
- For SIP only, media released between multiple Oracle Communications Session Border Controllers.
  To enable SIP distributed media release, you must set the appropriate parameter in the realm configuration. You must also set the SIP options parameter to media-release with the appropriate header name and header parameter information. This option defines how the Oracle Communications Session Border Controller encodes IPv4 address and port information for media streams described by, for example, SDP.

Configuring Media Release Behind the Same IP Address

You need to configure both the mm-in-realm and mm-same-ip parameters for the realm:

- If the mm-in-realm parameter is disabled, the mm-same-ip parameter is ignored.
- If the mm-in-realm parameter is enabled and the mm-same-ip parameter is disabled, media will be managed in the realm but released if the two endpoints are behind the same IP address.

To configure media management:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

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

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

   From this point, you can configure realm parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

4. **mm-in-realm**—Enable if you plan to use **mm-same-ip**. If this parameter is disabled, the **mm-same-ip** parameter is ignored. If you set this to **enabled** and mm-same-ip to **disabled**, media is managed in the realm but released if the two endpoints are behind the same IP address. The default is **disabled**. The valid values are:
   - enabled | disabled
5. **mm-same-ip**—Enable if you want media to go through this Oracle Communications Session Border Controller, if **mm-in-realm** is **enabled**. When **disabled**, the media will not go through the Oracle Communications Session Border Controller for endpoint that are behind the same IP. The default is **enabled**. The valid values are:
   - enabled | disabled

Bandwidth CAC for Media Release

The bandwidth CAC for media release feature adds per-realm configuration that determines whether or not to include inter-realm calls in bandwidth calculations. When you use this feature, the Oracle Communications Session Border Controller's behavior is to count and subtract bandwidth from the used bandwidth for a realm when a call within a single site has its media released. When you do not enable this feature (and the Oracle Communications Session Border Controller's previous behavior), the Oracle Communications Session Border Controller does not subtract the amount of bandwidth.

In other words:

- When you enable this feature, an inter-realm media-released call will decrement the maximum bandwidth allowed in that realm with the bandwidth used for that call.
- When you disable this feature (default behavior), and inter-realm media-released call will not decrement the maximum bandwidth allowed for that call.

Bandwidth CAC Configuration

To enable bandwidth CAC for media release:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

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

4. Select the realm where you want to want to add this feature.

   ```
   ORACLE(realm-config)# select
   ```

5. **bw-cac-non-mm**—Enable this parameter to turn on bandwidth CAC for media release. The default is **disabled**. The valid values are:
   - enabled | disabled
6. Save and activate your configuration.

Media Release between Endpoints with the Same IP Address

You can configure your Oracle Communications Session Border Controller to release media between two endpoints even when one of them:

- Is directly addressable at the same IP address as a NAT device, but is not behind a NAT device
- Is at the same IP address of a NAT device the other endpoint is behind

You enable this feature on a per-realm basis by setting an option in the realm configuration.

When this option is not set, theOracle Communications Session Border Controller will (when configured to do so) release media between two endpoints sharing one NAT IP address in the same realm or network.

Media Release Configuration

In order for this feature to work properly, the following conditions apply for the realm configuration:

- Either the **mm-in-realm** or the **mm-in-network** parameter must be disabled; you can have one of these enabled as long as the other is not. The new option will apply to the parameter that is disabled.
- If either the **mm-in-realm** or **mm-in-network** parameter is enabled, then the **mm-same-ip** parameter must be disabled.

To enable media release between endpoints with the same IP address:

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

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

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

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

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

If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **release-media-at-same-nat** with a plus sign in front of it, and then press Enter.

```
ORACLE(realm-config)# options +release-media-at-same-nat
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Media Release Behind the Same NAT IP Address

You can now configure your Oracle Communications Session Border Controller to release media between endpoints sharing the same NAT IP address, even if one endpoint is at—but not behind—the same NAT. This feature expands on the Oracle Communications Session Border Controller'S pre-existing ability to release media between calling and called parties behind the same IP address/NAT device in the same realm or network.

Media Release Configuration

For this feature to work properly, your realm configuration should either have the **mm-in-realm** or **mm-in-network** parameter set to disabled, unless the **mm-same-ip** parameter is set to disabled. If the **mm-same-ip** parameter is enabled, then **mm-in-realm** or **mm-in-network** can both be enabled.

To set the option that enables media release behind the same IP address:

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

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

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

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

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

If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **release-media-at-same-nat** with a plus sign in front of it, and then press Enter.

```
ORACLE(realm-config)# options +release-media-at-same-nat
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Codec Reordering

Certain carriers deploy voice services where their peering partners do not use the carriers' preferred codecs. The Oracle Communications Session Border Controller can now reorder the codecs so that the preferred one is selected first.

Take the example of a carrier that deploys a voice service using G.729 rather than G.711. If that carrier has a peering partner providing call origination for the VoIP customers with G.711 used as the preferred codec, there can be issues with codec selection.

The Oracle Communications Session Border Controller resolves this issue by offering its codec reordering feature. Enabled for realms and session agents, this feature gives the Oracle Communications Session Border Controller the ability to reorder the default codec in an SDP offer to the preferred codec before it forwards the offer to the target endpoint. When you enable this feature, you increase the probability that the target endpoint will choose the preferred codec for its SDP answer, thereby avoiding use of the undesired codec.

You enable codec reordering feature by setting the preferred-codec=X (where X is the preferred codec) option in the realm and session agent configurations. You set it in the realm from which the Oracle Communications Session Border Controller receives SDP offers (in requests or responses), and for which the media format list needs to be reordered by the Oracle Communications Session Border Controller prior to being forwarded. To configure additional codec ordering support for cases when a response or request with an SDP offer is from a session agent, you can set this option in the session agent configuration.

If you enable the option, the Oracle Communications Session Border Controller examines each SDP media description before if forwards an SDP offer. And if necessary, it performs reordering of the media format list to designate that the preferred codec as the default.

The Oracle Communications Session Border Controller determines preferred codecs in the following ways:

- If the response or request with an SDP offer is from a session agent, the Oracle Communications Session Border Controller determines the preferred codec by referring to the session agent configuration. You set the preferred codec for a session agent by configuring it with the preferred-codec=X option.
- If the response or request with an SDP offer is not from a session agent or is from a session agent that does not have the preferred-codec=X option configured, the Oracle Communications Session Border Controller determines the preferred codec by referring to the preferred-codec=X option in the realm.
- If the Oracle Communications Session Border Controller cannot determine a preferred codec, it does not perform codec reordering.

The way that the Oracle Communications Session Border Controller performs codec reordering is to search for the preferred codec in the SDP offer's media description (m=) line, and designate it as the default codec (if it is not the default already). After it marks the preferred codec as the default, the Oracle Communications Session Border Controller does not perform any operation on the remaining codecs in the media format list.

> **Note:**
>
> that the Oracle Communications Session Border Controller performs codec reordering on the media format list only. If the rtpmap attribute of the preferred codec is present, the Oracle Communications Session Border Controller does not reorder it.

Preferred Codec Precedence

When you configure preferred codecs in session agents or realms, be aware that the codec you set for a session agent takes precedence over one you set for a realm. This means that if you set preferred codecs in both configurations, the one you set for the session agent will be used.

In the case where the Oracle Communications Session Border Controller does not find the session agent's preferred codec in the SDP offer's media format list, then it does not perform codec reordering even if the media format list contains the realm's preferred codec.

Codec Reordering Configuration

When you configure codec ordering, the codec you set in either the session agent or realm configuration must match the name of a media profile configuration. If your configuration does not use media profiles, then the name of the preferred codec that you set must be one of the following:

- PCMU
- G726-32
- G723
- PCMA
- G722
- G728
- G729

> **Note:**
>
> If you configure this feature for a session agent, you must configure it for the associated realm as well. Otherwise, the feature will not work correctly.

Setting a Preferred Codec for a Realm

To set a preferred codec for a realm configuration:

These instructions assume that you want to add this feature to a realm that has already been configured.

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

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

4. Select the realm where you want to apply this feature.

   ```
   ORACLE(realm-config)# select
   identifier:
   1: public     media2:0         0.0.0.0
   2: private    media1:0         0.0.0.0
   selection:1
   ORACLE(realm-config)#
   ```

5. **options**—Set the **options** parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**preferred-codec=X**), and then press Enter. X is the codec that you want to set as the preferred codec.

   ```
   ORACLE(realm-config)# options +preferred-codec=PCMU
   ```

   If you type options preferred-codec=X, you will overwrite any previously configured options. In order to append the new option to the realm-config's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

Setting a Preferred Codec for a Session Agent

To set a preferred codec for a session agent configuration:

These instructions assume that you want to add this feature to a session agent that has already been configured.

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

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

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

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

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

4. Select the session agent where you want to apply this feature.

```
ORACLE(session-agent)# select
<hostname>:
1: acmepacket.com realm=       ip=
2: sessionAgent2  realm=tester ip=172.30.1.150
selection:
selection:1
ORACLE(session-agent)#
```

5. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**preferred-codec=X**), and then press Enter. X is the codec that you want to set as the preferred codec.

```
ORACLE(session-agent)# options +preferred-codec=PCMU
```

If you type options preferred-codec=X, you will overwrite any previously configured options. In order to append the new option to the session agent's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

6. Save and activate your configuration.

Media Profiles Per Realm

For different codecs and media types, you can set up customized media profiles that serve the following purposes:

- Police media values
- Define media bandwidth policies
- Support H.323 slow-start to fast-start interworking

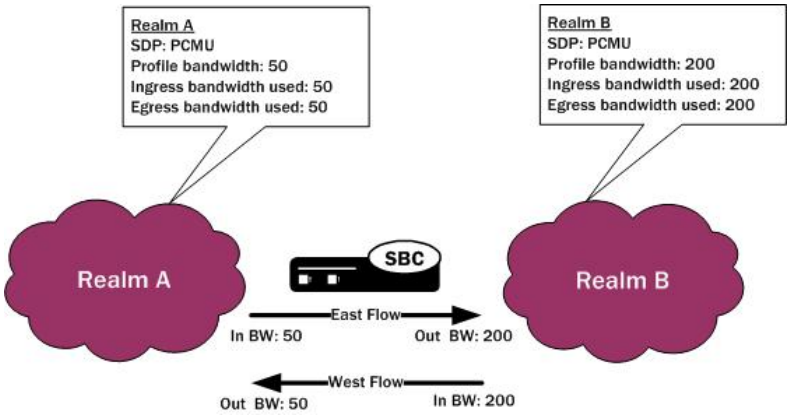
You can use media policies globally for the Oracle Communications Session Border Controller, or—starting with Release C6.1.0—you can configure them for application on a per-realm basis. For a realm, you can configure a list of media profiles you want applied. The Oracle Communications Session Border Controllermatches the value you set for the **match-media-profiles** parameter, and then applies those media profiles to the realm itself and to all of its child realms (but not to its parent realms).

> **Note:**
>
> This feature has no impact on the ways the Oracle Communications Session Border Controller uses media profiles non-realm applications such as: H.323 interfaces, SIP interfaces, IWF, session agents, codec policies, and policy attributes.

Call Admission Control and Policing

The Oracle Communications Session Border Controller supports call admission control (CAC) based on realm, and it applies the limits on either ingress or egress bandwidth counters. If a calls exceeds bandwidth on either the ingress or egress side, the Oracle Communications Session Border Controller rejects the call. You can also use per-user CAC, which limits the maximum bandwidth from the east and west flows for both the TO and FROM users.

When you apply media profiles to a realm, the Oracle Communications Session Border Controller applies bandwidth policing from the flow's ingress realm media profile. In the diagram below, the Oracle Communications Session Border Controller policies traffic for Realm A based Realm A's policing values, and the same is true for Realm B.



Media Profile Configuration

This section shows you how to configure multiple media profiles per realm, and it explains how to use wildcarding.

To reference a media profile in this list, you need to enter its name and subname values in the following format <name>::<subname>. Releases C6.1.0 and later accept the subname so you can configure multiple media profile for the same codec; the codec **name** customarily serves and the name value for a media profile configuration.

About Wildcarding

You can wildcard both portions (name and subname) of this value:

- When you wildcard the **name** portion of the value, you can provide a specific subname that the Oracle Communications Session Border Controller uses to find matching media profiles.

- When you wildcard the subname portion of the value, you can provide a specific **name** that the Oracle Communications Session Border Controller uses to find matching media profiles.

You can also enter the name value on its own, or wildcard the entire value. Leaving the subname value empty is also significant in that it allows the realm to use all media profile that have no specified **subname**. However, you cannot leave the **name** portion of the value unspecified (as all media profiles are required to have names).

Consider the examples in the following table:

Syntax	Example Value	Description
<name>	PCMU	Matches any and all media profiles with the name value configured as PCMU. This entry has the same meaning as a value with this syntax: <name>::*.
<name>::	PCMU::	Matches a media profile with the name with the name value configured as PCMU with an empty subname parameter.
<name>::<subname>	PCMU::64k	Matches a media profiles with the name with the name value configured as PCMU with the subname parameter set to 64k.
*	*	Matches anything, but does not have to be a defined media profile.
::	*::*	Matches any and all media profiles, but requires the presence of media profile configurations.
*::<subname>	*::64k	Matches all media profiles with this subname. You might have a group of media profiles with different names, but the same subname value.
*::	*::	Matches any media profiles with an empty subname parameter.
::	::	Invalid
::*	::*	Invalid

The Oracle Communications Session Border Controller performs matching for wildcarded **match-media-profiles** values last. Specific entries are applies first and take precedence. When the Oracle Communications Session Border Controller must decide between media profiles matches, it selects the first match.

To use media profiles for a realm:

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

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

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

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

4. **match-media-profiles**—In the form **<name>::<subname>**, enter the media profiles you would like applied to this realm. These values correspond to the name and subname parameters in the media profile configuration. You can wildcard either of these portions of the value, or you can leave the <subname> portion empty.

   This parameter has no default.

5. Save and activate your configuration.

Multiple Media Profiles

You can use the media profiles configuration to set up:

- One media profile for a particular SIP SDP encoding (such as G729), where the name of the profile identifies it uniquely. This behavior is your only option in Oracle Communications Session Border Controller release prior to Release C6.1.0.
- Multiple media profiles for the same SIP SDP encoding. Available in Release C6.1.0 and forward, you can create multiple media profiles for the same encoding. To do so, you add a subname to the configuration, thereby identifying it uniquely using two pieces of information rather than one.

The sections below provide two descriptions of deployments where using multiple profiles for the same codec would solve codec and packetization problems for service providers.

Use Case 1

Service Provider 1 peers with various carriers, each of which uses different packetization rates for the same codec. For example, their Peer 1 uses 10 milliseconds G.711 whereas their Peer 2 uses 30 milliseconds for the same codec. The difference in rates produces a difference in bandwidth consumption—resulting in a difference in SLA agreements and in Oracle Communications Session Border Controller call admission control (CAC) and bandwidth policing. Service Provider 1 uses the Oracle Communications Session Border Controller's media profile configuration parameters to determine CAC (**req-bandwidth**) and bandwidth policing (**avg-rate-limit**). Because this service provider's peers either do not use the SDP p-time attribute or use it inconsistently, it is difficult to account for bandwidth use. And so it is likewise difficult to set up meaningful media profiles.

The best solution for this service provider—given its traffic engineering and desire for the cleanest routing and provisioning structures possible—is to define multiple media profiles for the same codec.

Use Case 2

Service Provider 2 supports H.263 video, for which the Oracle Communications Session Border Controller offers a pre-provisioned media profile with a set bandwidth value. And yet, H.263 is not a codec that has a single bandwidth value. Instead, H.263 can have different bandwidth values that correspond to various screen resolution and quality. While it is true that the Oracle Communications Session Border Controllercan learn the requisite bandwidth value from SDP, not all SDP carries the bandwidth value nor do system operators always trust the values communicated.

Configuring multiple media profiles for the same codec (here, H.263) helps considerably with this problem—and moves closer to complete solution. Service Provider 2 can configure H.263 media profiles capable of handling the different bandwidth values that might appear.

Multiple Media Profiles Configuration

Configuring the **subname** parameter in the media profiles configuration allows you to create multiple media profiles with the same name.

To configure the subname parameter for a media profile:

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

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

2. Type **session-router** and press Enter.

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

3. Type **media-profile** and press Enter. If you are adding this feature to a pre-existing media profile configuration, you will need to select and edit your media profile.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. **subname**—Enter the subname value for this media profile. Information such as the rate or bandwidth value make convenient subname values. For example, you might set the **name** of the media profile as PCMU and the **subname** as 64k.

   This parameter is not require and has no default.

5. Save and activate your configuration.

Per-Realm Media Guard Timers

Oracle Communications Session Border Controller realm configurations support media guard timers whose settings take precedence over global media guard timers configured in the media manager. Both generic flow and TCP-specific flow timer settings are available. The user configures these timers in seconds in the realm-config.

Media guard timer parameters configured within a realm use the same syntax and define the same windows as the global timers. They differ in that they affect flows that traverse the realm only. Applicable timers include:

- **flow-time-limit**
- **initial-guard-timer**
- **subsq-guard-timer**
- **tcp-flow-time-limit**
- **tcp-initial-guard-timer**
- **tcp-subsq-guard-timer**

Per-realm media guard timer settings differ slightly from the global timers:

- The default value of -1 disables the realm setting, allowing the system to fall back to the global timer settings.

- A value of 0 disables the use of that type of timer for all flows traversing that realm, regardless of whether there are enabled ingress, egress or global settings.
- Timer ranges differ, as documented in the command reference.

Important operational considerations include:

- The system refers to both ingress and egress realm settings to time each flow, using the lower settings to resolve setting conflicts.
- Upon the expiry of the flow timer sequence, the Oracle Communications Session Border Controller sends a BYE that includes reason header indicating that the system has cleared the session due to media flow guard timer expiry.
- This feature interacts with guard-notify-gap in the media manager. When a flow guard timer expiry is received by MBCD, and if the required 'gap' since the last notification has not elapsed, the system queues the new notification to be sent when the required notify gap time has elapsed.

> **Note:**
>
> Media guard timers in static flow configurations are not impacted by per-realm (or global) media timers. Per-realm and global timers are for dynamic flows.

SIP Disable Media Inactivity Timer for Calls Placed on Hold

Hardware-based media flow guard timers detect when a call has lost media while it is being relayed through the Oracle Communications Session Border Controller. In response, the system tears down the call.

You can configure **disable-guard-timer-sendonly** to disable media inactivity timers for calls placed on hold. The Oracle Communications Session Border Controller disableds initial and subsequent guard timers for media when the SIP or IWF call is put on hold with a 0.0.0.0 address in:

- The c=connection line
- An a=inactive attribute
- An a=sendonly attribute

It should be noted that disabling the media inactivity timers will also disable the guard timers for calls which are not necessarily on hold, but simply are one-way audio applications.

Media Inactivity Timer Configuration

To disable the media inactivity timer for calls placed on hold:

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

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

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager-config** and press Enter.

```
ORACLE(media-manager)# media-manager-config
ORACLE(media-manager-config)#
```

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **disable-guard-timer-sendonly** with a plus sign in front of it, and then press Enter.

```
ORACLE(media-manager-config)# options +disable-guard-timer-sendonly
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Media Manager Configuration for Virtual Machines

The SBC provides you with a means of tuning the media manager for VM deployments.

As the SBC can classify traffic for use in DoS policing, bandwidth may be reserved for certain traffic types. Reserved bandwidth is expressed as a percentage of maximum available system bandwidth. The system's maximum bandwidth is determined by the hardware configuration and the number of available signaling cores. The maximum system bandwidth is defined as the speed of ingress traffic sent to the host, measured in packets per second (PPS). It is reported in the **show platform limits** command, referring to the "Maximum Signaling rate".

The following configuration options are available in the **media-manager-config**. These options are used to configure reserved bandwidth for application signaling, and ARP, and untrusted traffic.

- **max-signaling-packets**—Set the maximum overall bandwidth available for the host path in packets per second, which includes signaling messages from trusted and untrusted sources. The maximum value for each platform is used as the default value for that platform.
  - The maximum depends on the platform, as follows:
    - Acme Packet 1100 platform: the maximum is 10,000
    - Acme Packet 3900 platform: the maximum is 40,000
    - Acme Packet 3950/4900 platform: the maximum is 110,000
    - COTs and VM platforms: the maximum is system dependent
- **min-untrusted-signaling**—The minimum percentage of maximum system bandwidth available for untrusted traffic. The rest of the bandwidth is available for trusted resources, but can also be used for untrusted sources per max-untrusted-signaling. Default: 30. Range: 1-100.
- **max-untrusted-signaling**—The percentage of the maximum signaling packets you want to make available for messages coming from untrusted sources. This is a floating highwater mark and is only available when not in use by trusted sources. Default: 100. Range:1-100.
- **tolerance-window**—The size of the window, in seconds, used to measure host access limits for measuring the invalid message rate and maximum message rate for the realm configuration. Default: 30. Range: 0-4294967295.

- **max-arp-rate**—The maximum percentage or max-signaling-packets available for ARP traffic. Default: 30. Range: 1-100.

The user can also set controls on untrusted traffic from either realm or static ACL configuration using the **untrusted-signal-threshold** parameter.

5
SIP Signaling Services

This chapter explains how to configure the Oracle Communications Session Border Controller to support Session Initiation Protocol (SIP) signaling services for hosted IP services applications. SIP is a text-based application-layer signaling protocol that creates, identifies, and terminates multimedia sessions between devices.

About the Oracle Communications Session Border Controller and SIP

This section describes the Oracle Communications Session Border Controller's support of SIP. It provides the basic information you need to understand before you configure the Oracle Communications Session Border Controller for SIP signaling.

Types of SIP Devices

There are four types of SIP devices:

- SIP user agent (UA) is an endpoint in SIP end-to-end communication. A UA is a user agent client (UAC) when it initiates a request and waits to receive a response. A UA is a user agent server (UAS) when it receives a request and generates a response. A given UA will be a UAC or a UAS depending on whether it is initiating the request or receiving the request.
- A SIP proxy (or proxy server) is an intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server's primary role is routing. Its job is to ensure that a request is sent to another entity closer to the targeted user. A proxy interprets, and if necessary, rewrites specific parts of a request message before forwarding it.
- A SIP redirect server is a UAS that generates redirect responses to requests it receives, directing the client to contact an alternate set of targets. Unlike a proxy which forwards the request to the alternate set of targets, the redirect response tells the UAC to directly contact the alternate targets.
- A SIP registrar is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles. Proxies and redirect servers can use the information from the location service to determine the location of the targeted user.
  A redirect server and a registrar are each a special type of UA because they act as the UAS for the requests they process.

Basic Service Models

The Oracle Communications Session Border Controller operates as a back-to-back user agent (B2BUA) within the following two basic service models:

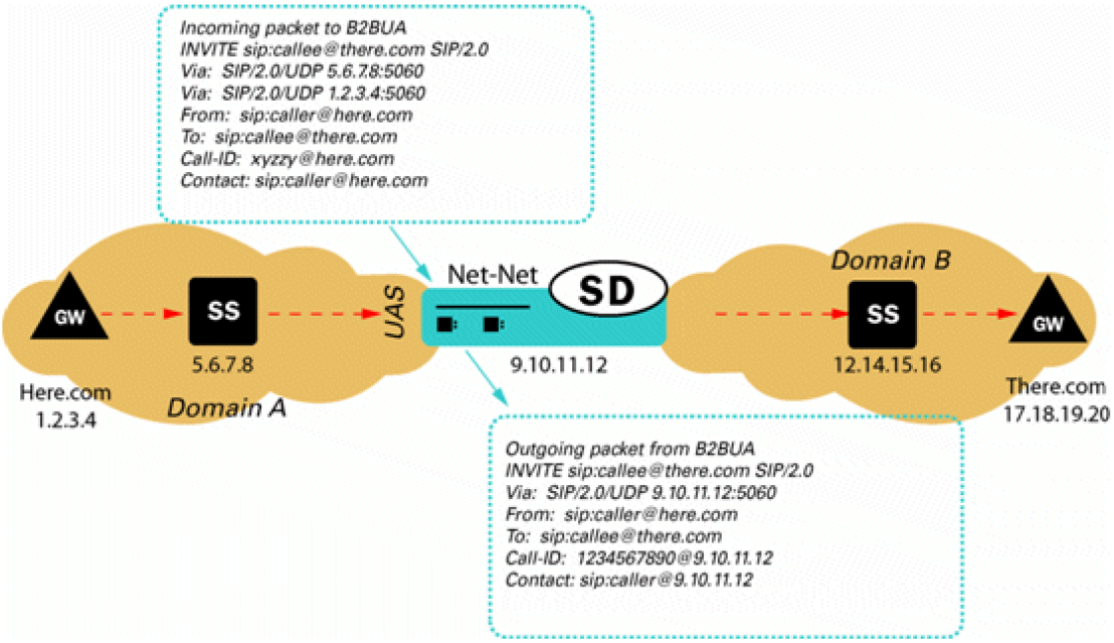
- peering
- hosted IP services

About B2BUA

A B2BUA is a logical entity that receives a request and processes it as a user agent server (UAS). In order to determine how the request should be answered, it acts as a user agent client (UAC) and generates requests. It maintains dialog state and must participate in all requests sent on the dialogs it has established.

SIP B2BUA Peering

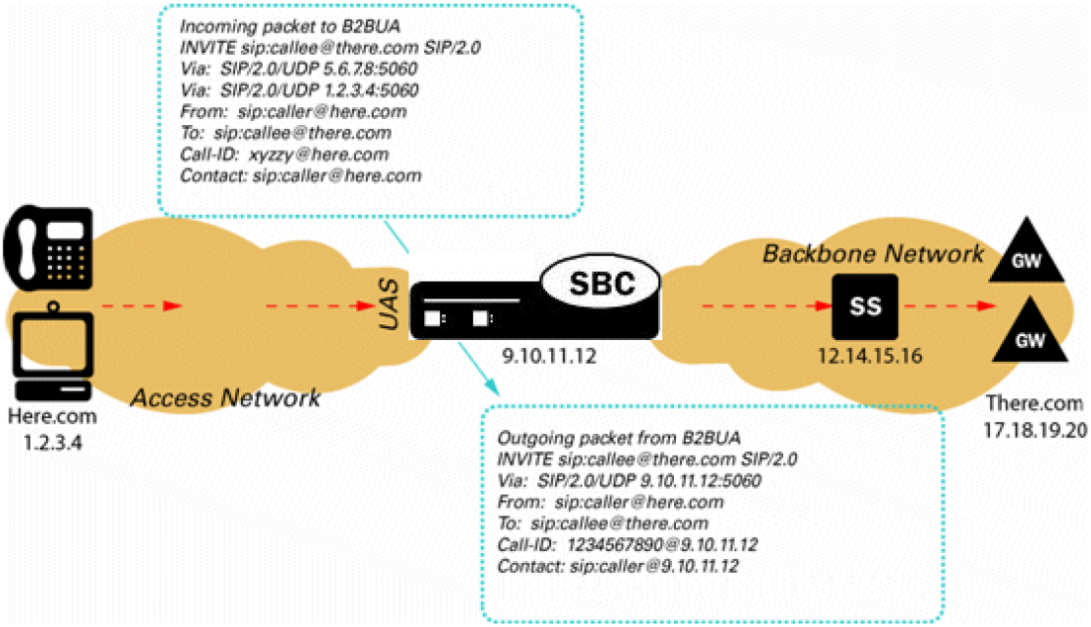
The Oracle Communications Session Border Controller operates as a SIP B2BUA. It terminates SIP sessions and re-originates them as new sessions as they are routed through the Oracle Communications Session Border Controller. For each session, it establishes NAPT translations and re-writes SDP to allow all session related media to be routed through the Oracle Communications Session Border Controller. It generates new call IDs and modifies SIP headers to prevent any protected SIP addresses and route information from being transmitted to external peers. The Oracle Communications Session Border Controller supports multiple SIP interfaces that are associated with a set of media ports, thus appearing as multiple virtual SIP gateways.



B2BUA Hosted IP Services

The Oracle Communications Session Border Controller acts as an outbound proxy for SIP endpoints and performs the operations required to allow UAs behind NATs to initiate and terminate SIP sessions (Hosted NAT Traversal).

The Oracle Communications Session Border Controller caches registration requests from SIP endpoints and forwards them to the appropriate softswitch or registrar in its backbone network. All subsequent signaling between the endpoint and the backbone network is through the Oracle Communications Session Border Controller. Also, all calling features such as caller ID, call waiting, three-way calling, and call transfer are all supported transparently through the Oracle Communications Session Border Controller.



SIP B2BUA and L3 L5 NAT

For each SIP session, the Oracle Communications Session Border Controller establishes NAPT translations and re-writes SDP to route all session related media through the Oracle Communications Session Border Controller. These actions make the Oracle Communications Session Border Controller look like a SIP gateway. Also, the Oracle Communications Session Border Controller support of multiple SIP interfaces associated with different network interfaces makes it appear as multiple virtual SIP gateways.

This functionality enables the Oracle Communications Session Border Controller to deliver VoIP services to multiple end users, across a VPN backbone.

About SIP Interfaces

The SIP interface defines the transport addresses (IP address and port) upon which the Oracle Communications Session Border Controller receives and sends SIP messages. You can define a SIP interface for each network or realm to which the Oracle Communications Session Border Controller is connected. SIP interfaces support both UDP and TCP transport, as well as multiple SIP ports (transport addresses). The SIP interface's SIP NAT function lets Hosted NAT Traversal (HNT) be used in any realm.

SIP INVITE Message Processing

When the session agent element on the softswitch side of the message flow (ingress session agent) has the gateway contact parameter configured as an option, the Oracle Communications Session Border Controller looks for the URI parameter (as defined by the gateway contact parameter) in the Request-URI and decodes the gateway address.

Example

The following example shows a SIP INVITE message from a softswitch to a Oracle Communications Session Border Controller.

```
INVITE sip:05030205555@ss-side-ext-address;gateway=encoded-gw-address
From: "Anonymous"<sip:anonymous@anonymous.invalid>;tag=xxxx
To: <sip:05030205555@ss-side-ext-address;user=phone>
```

The following example shows a SIP INVITE message from a Oracle Communications Session Border Controller to a gateway.

```
INVITE sip:05030205555@gw-ip-address SIP/2.0
From: "Anonymous"<sip:anonymous@anonymous.invalid>;tag=SDxxxx-xxxx
To: <sip:05030205555@ hostpart;user=phone>
```

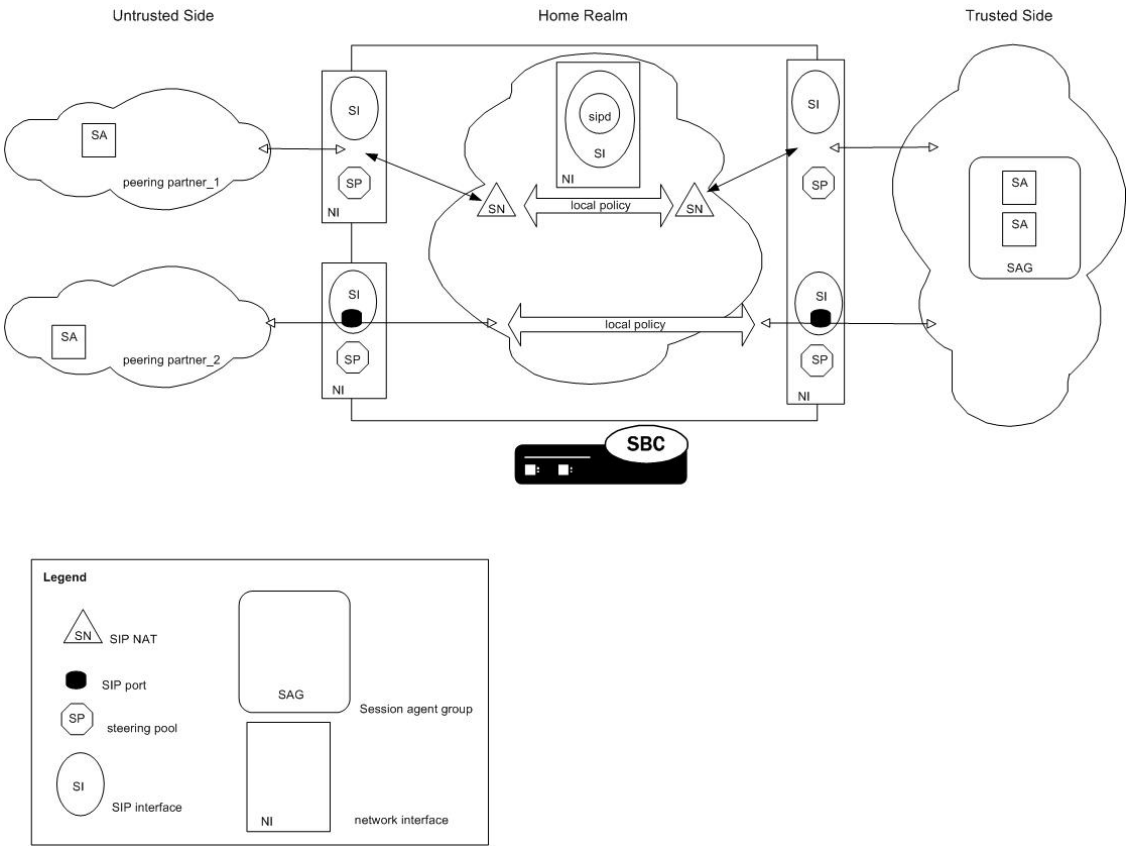
The Oracle Communications Session Border Controller converts the hostpart in the To header except in the following scenarios:

- when the original hostpart value received is an Fully Qualified Domain Name (FQDN)
- when the Oracle Communications Session Border Controller is configured not to NAT the To headers.

Oracle recommends configuring the Oracle Communications Session Border Controller to NAT the To headers to ensure the security of protected addresses. Otherwise, the outgoing hostpart is set to the SIP NAT's external proxy address for the SIP NAT's external realm.

Configuring the Oracle Communications Session Border Controller for SIP Signaling

This section contains a diagram of a B2BUA peering environment that illustrates the Oracle Communications Session Border Controller components you need to configure.



Home Realm

This section explains how to configure a home realm. The home realm applies only to a SIP configuration. It represents the internal default realm or network for the Oracle Communications Session Border Controller and is where the Oracle Communications Session Border Controller's SIP proxy is located.

Overview

You primarily use a home realm when using the SIP NAT function to connect multiple realms/networks to the Oracle Communications Session Border Controller. You define the home realm defined as either public or private for the purposes of using the SIP NAT function. If the home realm is public, all external realms are considered private. If the home realm is private, all external networks are considered public. Usually the home realm is public.

Messages are encoded (for example, the topology is hidden) when they pass from a private to a public realm. Messages are decoded when the pass from a public realm to a private realm.

These external realms/networks might have overlapping address spaces. Because SIP messages contain IP addresses, but no layer 2 identification (such as a VLAN tag), the SIP proxy must use a single global address space to prevent confusing duplicate IP addresses in SIP URIs from different realms.

SIP NAT Function

The SIP NAT function converts external addresses in SIP URIs to an internal home realm address. Usually the external address is encoded into a cookie that is added to the userinfo

portion of the URI and the external address is replaced with a home realm address unique to the SIP NAT (the SIP NAT home address).

URIs are encoded when they pass from a private realm to a public realm. When an encoded URI passes back to the realm where it originated, it is decoded (the original userinfo and host address are restored). The encoding/decoding process prevents the confusion of duplicate addresses from overlapping private addresses. It can also be used to hide the private address when a SIP message is traversing a public network. Hiding the address occurs when it is a private address; or when the owner of the private network does not want the IP addresses of their equipment exposed on a public network or on other private networks to which the Oracle Communications Session Border Controller connects.

Home Realm's Purpose

A home realm is required because the home address for SIP NATs is used to create a unique encoding of SIP NAT cookies. You can define the home realm as a network internal to the Oracle Communications Session Border Controller, which eliminates the need for an actual home network connected to the Oracle Communications Session Border Controller. You can define this virtual home network if the supply of IP addresses is limited (because each SIP NAT requires a unique home address), or if all networks to which the Oracle Communications Session Border Controller is connected must be private to hide addresses.

For example, you can define a public home realm using the loopback network (127.0.0.0) and using the home realm address prefix (for example, 127.0.0.0/8) for encoding addresses that do not match (all addresses outside 127.0.0.0/8) in SIP NAT cookies. The SIP NAT address prefix field can be used to accomplish this while keeping the ability to define an address prefix for the ream for ingress realm determination and admission control. By defining the SIP NAT address prefix as 0.0.0.0, the home realm address prefix is used to encode addresses that do not match.

Home Realm Configuration

To configure the home realm:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type **sip-config** and press Enter. The system prompt changes.

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

   From this point, you can configure SIP configuration parameters. To view all sip-config parameters, enter a **?** at the system prompt.

4. **home-realm-id**—Enter the name of the realm you want to use for the realm ID. For example, **acme**.

   The name of the realm must correspond to the identifier value you entered when you configured the realm.

5. **egress-realm-id**—Optional. Enter the egress realm ID to define the default route for SIP requests addressed to destinations outside the home realm's address prefix.

   If you enter a value for this optional field, it must correspond to the identifier value you entered when you configured the realm.

   > **Note:**
   >
   > You should leave this parameter blank for access/backbone applications. When left blank, the realm specified in the home-realm-id parameter is used by default.

6. **nat-mode**—Indicate the SIP NAT mode. The default is **none**. The valid values are:

   - **public**—Indicates the subnet defined in the addr-prefix-id field of the home realm is public and the subnet defined in the addr-prefix-id field of all external realms identified in the SIP NAT are private networks. IPv4 addresses are encoded in SIP messages received from the external realm defined by the SIP NAT. The IPv4 addresses are decoded in messages that are sent to the realm.
   - **private**—Indicates the subnet defined in the addr-prefix-id field of the home realm is private and the subnet defined in the addr-prefix-id field of all external realms identified in the SIP NAT are public networks. IPv4 addresses are encoded in SIP messages sent to the external realm defined by the SIP NAT and decoded in messages received from the realm.
   - **none**—No SIP NAT function is necessary.

     The following example shows the SIP home realm configured for a peering network.

```
sip-config
        state                           enabled
        operation-mode                  dialog
dialog-transparency        disabled
        home-realm-id                   acme
        egress-realm-id
        nat-mode                        Public
        registrar-domain
        registrar-host
        registrar-port                  0
        init-timer                      500
        max-timer                       4000
        trans-expire                    32
        invite-expire                   180
        inactive-dynamic-conn           32
        red-sip-port                    1988
        red-max-trans                   10000
        red-sync-start-time             5000
        red-sync-comp-time              1000
        last-modified-date              2005-03-19 12:41:28
```

SIP Interfaces

This section explains how to configure a SIP interface. The SIP interface defines the transport addresses (IP address and port) upon which theOracle Communications Session Border Controller receives and sends SIP messages.

Overview

The SIP interface defines the signaling interface. You can define a SIP interface for each network or realm to which the Oracle Communications Session Border Controller is connected. SIP interfaces support both UDP and TCP transport, as well as multiple SIP ports (transport addresses). The SIP interface also lets Hosted NAT Traversal (HNT) be used in any realm.

The SIP interface configuration process involves configuring the following features:

- address and transport protocols (SIP ports)
- redirect action
- proxy mode
- trust mode

About SIP Ports

A SIP port defines the transport address and protocol the Oracle Communications Session Border Controller will use for a SIP interface for the realm. A SIP interface will have one or more SIP ports to define the IP address and port upon which the Oracle Communications Session Border Controller will send and receive messages. For TCP, it defines the address and port upon which the Oracle Communications Session Border Controller will listen for inbound TCP connections for a specific realm.

You need to define at least one SIP port, on which the SIP proxy will listen for connections. If using both UDP and TCP, you must configure more than one port. For example, if a call is sent to the Oracle Communications Session Border Controller using TCP, which it needs to send out as UDP, two SIP ports are needed.

Preferred SIP Port

When a SIP interface contains multiple SIP ports of the same transport protocol, a preferred SIP port for each transport protocol is selected for outgoing requests when the specific SIP port cannot be determined. When forwarding a request that matched a cached registration entry (HNT or normal registration caching), the SIP port upon which the original REGISTER message arrived is used. Otherwise, the preferred SIP port for the selected transport protocol is used. When selecting the preferred SIP port, the default SIP port of 5060 will be selected over other non-default ports.

For SIP interfaces using the SIP NAT function, the preferred SIP port address and port will take precedence over the external address of the SIP NAT when they do not match. If both TCP and UDP SIP ports are defined, the address and port of the preferred UDP port is used.

Proxy Mode

The Oracle Communications Session Border Controller's proxy mode determines whether it forwards requests received on the SIP interface to target(s) selected from local policy; or sends a send a redirect response to the previous hop. Sending the redirect response causes the previous hop to contact the targets directly.

If the source of the request matches a session agent with a proxy mode already defined, that mode overrides the proxy mode defined in the SIP interface.

You can configure the proxy mode to use the Record-Route option. Requests for stateless and transaction operation modes are forwarded with a Record-Route header that has the Oracle

Communications Session Border Controller's addresses added. As a result, all subsequent requests are routed through the Oracle Communications Session Border Controller.

Redirect Action

The redirect action is the action the SIP proxy takes when it receives a SIP Redirect (3xx) response on the SIP interface. If the target of the request is a session agent with redirect action defined, its redirect action overrides the SIP interface's.

You can set the Oracle Communications Session Border Controllerto perform a global redirect action in response to Redirect messages. Or you can retain the default behavior where the Oracle Communications Session Border Controller sends SIP Redirect responses back to the previous hop (proxy back to the UAC) when the UAS is not a session agent.

The default behavior of the Oracle Communications Session Border Controller is to recurse on SIP Redirect responses received from the user agent server (UAS) and send a new request to the Contact headers contained in the SIP Redirect response.

Instead of this default behavior, the Oracle Communications Session Border Controller can proxy the SIP Redirect response back to the user agent client (UAC) using the value in the session agent's redirect action field (when the UAS is a session agent). If there are too many UASes to define as individual session agents or if the UASs are HNT endpoints, and SIP Redirect responses need to be proxied for UASs that are not session agents; you can set the default behavior at the SIP Interface level.

SIP maddr Resolution

Release S-C6.2.0 provides enhanced resolution of addresses found in SIP contact headers, or in the maddr (multicast address) parameter of SIP 3xx REDIRECT messages. Previous releases resolved these addresses as either a host address or as a session agent name. With Release 6.2.0 these addresses can also be resolved as session agent group (SAG) names.

Support for SAG-based resolution is provide by a new **sip-config** parameter, **sag-lookup-on-redirect**. By default, SAG lookup is disabled, providing compatibility with prior releases.

The following sample SIP REDIRECT and ACLI configuration fragment illustrate enhanced processing.

```
Status-Line: SIP/2.0 302 Moved
Message Header
Via: SIP/2.0/UDP
192.168.200.224:5060;branch=z9hG4bKa0fs40009o90sc8oo780.1
From: <sip:1111@192.168.1.222:6000>;tag=1
To: sut <sip:2223@192.168.1.224:5060>;tag=11
Call-ID: 1-28515@192.168.1.222
CSeq: 1 INVITE
Contact: <sip:1111@192.168.1.223;maddr=test.acmepacket.com>
Privacy: user;id;critical;session
P-Preferred-Identity: sipp <sip:sipp@192.168.200.222:5060>
P-Asserted-Identity: abc.com
Subject: abc
Proxy-Require: privacy,prack,abc
Content-Length: 0

session-group
        group-name                      test.acmepacket.com
        description
```

```
        state                           enabled
        app-protocol                    SIP
        strategy                        Hunt
        dest
                                        192.168.200.222
                                        192.168.200.223
...
...
```

In this case, when the Oracle Communications Session Border Controller receives the 302, it resolves the information from maddr to a SAG name. In the above example, it will resolve to the configured SAG – test.acmepacket.com. The destinations configured in SAG test.acmepacket.com will be used to route the call.

SAG-based address resolution is based on the following set of processing rules.

1. When the Contact URI does not have an maddr parameter, and the hostname is not an IP Address, the Oracle Communications Session Border Controller will look for a SAG matching the hostname.
2. When the Contact URI has an maddr parameter that contains an IP address, the Oracle Communications Session Border Controller will not look for a SAG; it will use the IP Address as the target/next-hop.
3. When the Contact URI has an maddr parameter that contains a non-IP-address value, the Oracle Communications Session Border Controller will look for a SAG matching the maddr parameter value.

   The above logic can be turned on by enabling sag-lookup-on-redirect in the sip-config object as shown below.

SIP maddr Resolution Configuration

To configure the Oracle Communications Session Border Controller to perform SAG-based maddr resolution:

1. From superuser mode, use the following command sequence to access sip-config configuration mode. While in this mode, you configure SAG-based address resolution.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

2. Use the **sag-lookup-on-redirect** parameter to enable SAG-based maddr resolution.
3. Use **done**, **exit**, and **verify-config** to complete SAG-based address resolution.

Trust Mode

The Oracle Communications Session Border Controller supports the Calling Identity privacy requirements based on RFC 3323 and RFC 3325. The trust mode in the SIP interface determines whether the source and destination of a request is a trusted entity. With the implementation of this feature, the Oracle Communications Session Border Controller can understand and support the privacy headers and provide the capability for anonymous packets

The Oracle Communications Session Border Controller, which acts as a boundary device between the trusted platform and the untrusted Internet, understands the following headers:

- Privacy Header
- P-Asserted-Identity Header
- P-Preferred-Identity Header

Depending on the value of these headers and the mode in which the Oracle Communications Session Border Controller is being operated (B2BUA or the proxy), the appropriate actions are performed.

About the Process

On receiving a message, the Oracle Communications Session Border Controller checks whether the message source is trusted or not. It checks the SIP interface's trust mode value and, if the source is a session agent, the session agent's trust me value. Depending on these values, the Oracle Communications Session Border Controller decides whether the request's or response's source is trusted. If it receives message from a trusted source and the message contains the P-Asserted-Identity header field, the Oracle Communications Session Border Controller passes this message to the outgoing side. The outgoing side then decides what needs to be done with this request or response.

If the request or the response is received from an untrusted source, the Privacy header value is id (privacy is requested), and the P-Asserted-Identity header field is included, the Oracle Communications Session Border Controller strips the Privacy and the P-Asserted-Identity headers and passes the request or the response to the outgoing side.

If the request or the response contains the P-Preferred-Identity header and the message source is untrusted, the Oracle Communications Session Border Controller strips the P-Preferred-Identity header from the request or the response and passes the message to the outgoing side.

If the source is trusted or privacy is not requested (the value of the Privacy Header is not id) and the request or the response contains the P-Preferred-Identity header, the Oracle Communications Session Border Controller performs the following actions:

- inserts the P-Asserted-Identity header field with the value taken from the P-Preferred-Identity header field
- deletes the P-Preferred-Identity header value
- passes this request or the response to the Outgoing side for the appropriate action, depending on the whether the destination is trusted or not

After the Oracle Communications Session Border Controller passes the request or the response to the outgoing side, it checks whether the destination is trusted by checking the SIP interface's trust mode value and the session agent's trust me value (if the destination is configured as session agent).

- The destination is trusted
  The Oracle Communications Session Border Controller does nothing with the request or the response and passes it to the destination. If the P_Asserted_Identity headers are present, they are passed to the session agent (if the destination is configured as session agent).
- The destination is untrusted
  The Oracle Communications Session Border Controller looks at the value of the Privacy header. If set to id, the Oracle Communications Session Border Controller removes all the P-Asserted-Identity headers (if present). It strips the Proxy-Require header if it is set to privacy. The Oracle Communications Session Border Controller also sets the From field of SIP header to Anonymous and strips the Privacy header.

If the Privacy header is set to none, the Oracle Communications Session Border Controller does not remove the P-Asserted-Identity header fields.

If there is no Privacy header field, the SD will not remove the P-Asserted-Identity headers.

To implement this feature, you need to configure the session agent's trust me parameter to enabled (if the message source is a session agent) and the SIP interface's trust mode to the appropriate value.

Call Duration Counters

The Oracle Communications Session Border Controller maintains aggregate call duration in seconds for the current period, lifetime total and the lifetime-period-maximum. These counters are maintained for each session agent, realm, SIP Interface, and globally across the system. The call duration counter can count up to a 32 bit value, after which time it rolls over.

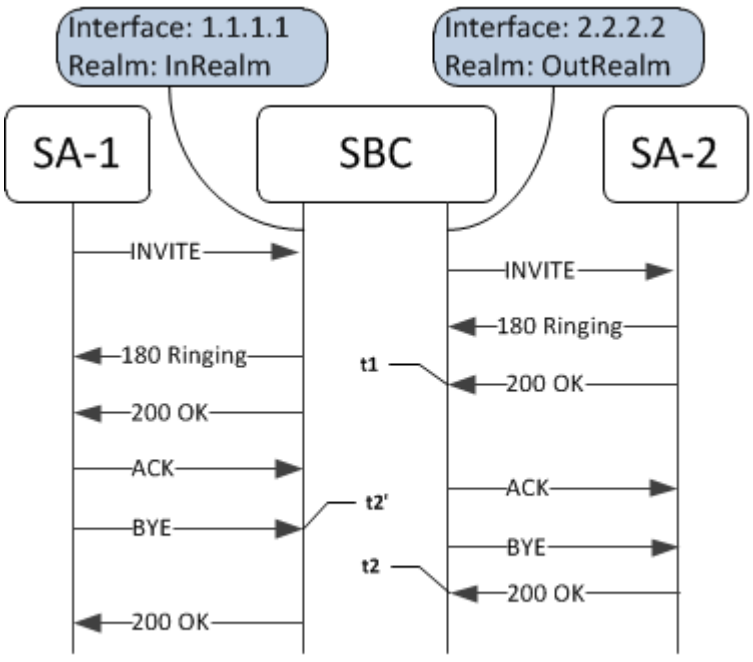
The call duration counters are displayed in the following show commands:

- show sipd agents
- show sipd realms
- show sipd interface
- show sipd status

Call Duration Calculation

Call duration is calculated as the time between (t1) when the SBC receives the 200OK for the INVITE from terminating endpoint, and (t2) the time when SBC receives the final 200OK for the BYE message from terminating endpoint, regardless of whether media is released. If the **set-disconnect-time-on-bye** parameter is enabled in the **sip-config**, then the call termination time (t2') is when the SBC receives the BYE message originally from the endpoint ending the call.

Figure 5-1 Call Duration Calculation Example - Ladder Diagram



t2' when `sip-config > set-disconnect-time-on-bye=enabled`

Table 5-1 Call Duration Calculation Example - Results

Element	Inbound Duration (sec)	Outbound Duration (sec)
Session Agent: SA-1	t2 - t1	0
Session Agent: SA-2	0	t2 - t1
Realm: InRealm	t2 - t1	0
Ream: OutRealm	0	t2 - t1
Interface: 1.1.1.1	t2 - t1	0
Interface: 2.2.2.2	0	t2 - t1

The global system call duration for the previous example is t2' - t1 seconds if the **set-disconnect-time-on-bye** parameter is enabled.

Configurable Timers and Counters

SIP timers and counters can be set in the global SIP configuration, and two can be specific for individual SIP interfaces.

You can set the expiration times for SIP messages, and you can set a counter that restricts the number of contacts that the Oracle Communications Session Border Controller tries when it receives a REDIRECT. These are similar to two parameters in the global SIP configuration, trans-expire and invite-expire. You can also set a parameter that defines how many contacts/routes the Oracle Communications Session Border Controller will attempt on redirect.

Timer to Tear Down Long Duration Calls

The Oracle Communications Session Border Controller currently provides the “flow-time-limit” timer to terminate long duration calls. However, this timer is reset whenever the Oracle Communications Session Border Controller receives a Re-INVITE or UPDATE message, even when it is provided for the session timer. This feature adds a non-resettable timer that, when enabled and upon expiration, tears down long duration calls.

When the “flow-time-limit” timer for terminating long media flow expires, the Oracle Communications Session Border Controller sends a SIP BYE or H323 Release Complete message to tear down the long call. However, this timer is reset whenever the SBC receives a Re-INVITE or UPDATE message. In most call scenarios, long duration calls are terminated with the expiration of this timer, but there are some cases where a call can stay connected for a longer duration. For example, if a user connects to an IVR service and does not hang up the phone receiver properly, there is no way for the network provider to free up the IVR resources if the user devices send session updating requests. To prevent this situation, this feature adds a new timer **session-max-life-limit**, which starts when the call or session is established and does not reset for any session update, keep-alive or SBC switchover. On expiry, the call is torn down if it’s in established state.

The new timer **session-max-life-limit** can be provisioned in the following configuration elements, in order of precedence from highest to lowest: **session-agent**, **realm-config**, **sip-interface**, and **sip-config**. Its range of values is {0-2073600} seconds with an additional special case value of “Unlimited”, which is treated as the highest possible value. The default value is 0 (no timer).

Difference between 0 and Unlimited

No timer is created when **session-max-life-limit** is configured to either the value 0 or “Unlimited”, so no timeout can occur. The difference between the two values is how they are

handled when determining which value of **session-max-life-limit** to use when there are several specified within the various configuration elements. When a session is created the timer examines both the ingress side and the egress side and, in cases where both sides have a configured value for **session-max-life-limit**, uses the side with the lower (stricter) value. On each side, the SBC reviews the configuration elements relevant to the session and uses the value of **session-max-life-limit** from the configuration element with the highest precedence (**session-agent**, then **realm-config**, then **sip-interface**, and lastly**sip-config**). When the value is set to 0, the configuration element is ignored and the next configuration element in the precedence chain is looked at. A value between 1 and 2073600 (24 days) or the value “Unlimited” is treated as a valid configured value. In this case the SBC will not move onto the next element in the precedence chain and the value is used in the final comparison between the egress and ingress values. The value “Unlimited” is viewed as the highest possible value, and therefore is considered greater than any other value it is compared against. The value 0 is skipped over and completely ignored.

For example, on the ingress side the value of **session-max-life-limit** in **realm-config** is set to 86400 and the value of **session-max-life-limit** in **session-agent** is set to "Unlimited". The **session-agent** value has a higher precedence than the **realm-config** value so, therefore, the value “Unlimited” is used for the ingress side. On the egress side the value of **session-max-life-limit** in **realm-config** is set to 43200 and the value of **session-max-life-limit** in **session-agent** is set to 0 (no timer), so the value of **session-max-life-limit** in **realm-config** is used. When compared against the ingress side the value 43200 is less than “Unlimited”; therefore, the value set for the timer is 43200.

Timer to Tear Down Long Duration Calls Configuration

Use the following procedure to configure, in the **session-agent**, **realm-config**, **sip-interface**, and **sip-config** configuration elements, a non-resettable timer that, when enabled and upon expiration, tears down long duration calls.

Although the timer occurs in four separate configuration elements, for brevity only the procedure for configuring **realm-config** is shown as the general procedure does not vary for the other configuration elements.

For the **realm-config** configuration element:

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

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

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **session-max-life-limit** — Enter the maximum interval in seconds before the SBC must terminate long duration calls. The value supercedes the value of **session-max-life-limit** in the **sip-interface** and **sip-config** configuration elements and is itself superceded by the value of **session-max-life-limit** in the **session-agent** configuration element. The default value is 0 (off/ignored).

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

Asymmetric Preconditions

This feature allows you to reserve network resources for a session before the called party is alerted by establishing preconditions for individual call legs. Currently, the Oracle Communications Session Border Controller transparently passes precondition attributes in SIP signaling and the UEs negotiate preconditions end to end. However, some networks support the SIP preconditions and other networks do not. To help provide precondition interworking between these networks, the Oracle Communications Session Border Controller supports SIP preconditions on a per call leg basis.

Some architectures require minimizing the chance of session establishment failure after alerting the called party. One source of failure is the inability to reserve network resources for a session, which can cause media clipping and "ghost rings". The solution is to allow preconditions to reserve network resources for the session before alerting the called party. A precondition is a set of constraints about the session which are introduced in the offer. The recipient of the offer generates an answer, but does not alert the user or otherwise proceed with session establishment until the preconditions are met. This can be known through a local event (such as a confirmation of a resource reservation), or through a new offer sent by the caller. The concept of preconditions is explained in RFC 3312, updated by RFC 4032, and also 3GPP TS 24.229.

The Oracle Communications Session Border Controller handles preconditions either leg by leg or transparently passes preconditions through in SDP based on the configuration. When leg by leg management is done, the SBC can behave as a UAS or a UAC. When behaving as a UAS, the SBC complies with 3GPP TS 24.229 Sections 5.1.4.1 and 6.1.3. When the parameter **asymmetric-preconditions** is enabled on the ingress interface and disabled for the egress interface at the SBC:

- The SBC does not use preconditions when the received INVITE request does not include the **precondition** option tag in the Supported or Require header fields.
- When the received INVITE request includes the **precondition** option tag in the Supported or Require header fields, the SBC uses the precondition mechanism and inserts a Require header field with the **precondition** option tag in any response including an SDP or subsequent request (also with SDP) that it sends to the originating UE during session establishment.
- When the received INVITE request includes the **100rel** and **preconditions** option tags and the SDP offer contains no precondition attributes, the equipment behaves as if the INVITE was received without an SDP. In this case the SBC just proxies the request and no preconditions interworking occurs.
- When the received INVITE request includes the **100rel** and **preconditions** option tags and there is no SDP (content) in the incoming INVITE, the SBC treats the INVITE as an offerless case and preconditions interworking occurs.
- The SBC always has resources reserved, so the value of the SDP parameter **direction-tag** in the preconditions current attribute line will always be **sendrecv** for the SDP parameter **status-type** value **local** for the SBC.

When behaving as a UAC, the SBC complies with RFC 3312 and 3GPP TS 24.229 Sections 5.1.3.1 and 6.1.2. When the parameter **asymmetric-preconditions** is enabled on the egress interface and disabled for the ingress interface, upon generating an initial INVITE request using the precondition mechanism, the SBC indicates preconditions interworking by including the precondition option tag in the Supported header field and not in the Require header.

- The SBC always has resources reserved, so the value of the SDP parameter **direction-tag** in the preconditions current attribute line will always be **sendrecv** for the SDP parameter **status-type** value **local** for the SBC.
- If the UE answers with a request for confirmation of resources, the SBC resends the UPDATE with **sendrecv** on the local preconditions for the SBC. As the SBC always confirms the preconditions in the first response or in the request it sends towards the UE, there is no need for the UE to confirm resources on the SBC.

When appropriately configured, the Oracle Communications Session Border Controller inserts SDP into an outgoing INVITE when the corresponding incoming INVITE has none. Because no SDP information is available for the session, the Oracle Communications Session Border Controller uses a media profile from a previously configured list to determine the content to insert. You cannot enable asymmetric preconditions without first enabling PRACK interworking by setting the value of **sip-interface**, **options** to **100rel-interworking**. The SBC does precondition interworking by creating SDP offers or answers before the actual ones have reached the UE; thus, transcoding policies are needed to cover the cases where different codecs are negotiated. However, in cases where different codecs are not expected, interworking works without defining codec policies. When codec policies are not defined and a final SDP answer comes with a different codec, the SBC will drop the call with 488 Not Acceptable.

To establish preconditions interworking, you must first enable it on the SIP interface with the parameter **asymmetric-preconditions** and then define when the initial offer will be sent to the endpoint that doesn't support preconditions with the parameter **asymmetric-preconditions-mode**. When the value of **asymmetric-preconditions-mode** is **send-with-nodelay**, the SBC forwards the INVITE as soon as it is received; however, it strips the **precondition** option tag from the Require and Supported header fields and also removes the preconditions related SDP. This triggers preconditions interworking on the calling side of the SBC. Responses received from the called party are queued until resource reservation is complete on the calling side. When resource reservation is complete, all buffered responses from the called party are sent to the calling party (after, if necessary, removing the SDP as in PRACK interworking) and the call proceeds as normal. When the value of **asymmetric-preconditions-mode** is **send-with-delay** on the egress, the initial INVITE forwarded to the called party is held until preconditions are met by the calling party. For offerless 18x INVITEs, that is, when the incoming INVITE has no STP, the SBC inserts the SDP, as configured in the media profile names listed in **add-sdp-profiles-in-msg**, in the 18x (183) response towards the UE.

Asymmetric Preconditions Configuration Guidelines

Set the **asymmetric-preconditions** parameter to **enabled** on any **sip-interface** that connects to networks requiring preconditions support.

> **Note:**
>
> Offer-less scenarios do not use SDP attributes to denote the use of preconditions. All such messaging uses precondition tags.

Egress	Ingress	Notes
disabled	disabled	INVALID configuration Proxy the request, no IWF

Egress	Ingress	Notes
disabled	enabled	If the INVITE has no precondition attributes, then proxy. If the INVITE has precondition tags and attributes, trigger precondition IWF. If, the egress interface has **asymmetric-preconditions-mode** set to **send-with-no delay**, the INVITE will be forwarded immediately as soon as SBC receives it. If, the egress interface has asymmetric-preconditions-mode=send-with-delay, the INVITE will be held until preconditions are complete and SBC will forward it.
enabled	disabled	If, Incoming INVITE has preconditions, then proxy, no IWF. If, Incoming INVITE has no preconditions, insert precondition tags and SDP. If response comes with precondition tag, then continue with IWF. If response has no precondition option tags, no IWF - disable the functionality and proxy.
enabled	enabled	INVALID configuration Proxy the request, no IWF

When an **INVITE** arrives with both precondition tags and precondition SDP attributes, the SBC behaves as listed below, sending the message with or without delay, based on the **asymetric-preconditions-mode** configuration on the outgoing interface.

- The SBC performs preconditions-to-nonpreconditions interworking by stripping the precondition tags and attributes and forwarding when it is configured with:
  - Incoming interface:
    - **100rel-interworking** option **enabled**
    - **asymetric-preconditions** parameter set to **enabled**
  - Outgoing interface:
    - **asymetric-preconditions** parameter set to **disabled**
- The SBC proxies the request, and does not perform interworking when it is configured with:
  - Incoming interface:
    - **100rel-interworking** option **disabled**
    - **asymetric-preconditions** parameter set to **disabled**
  - Outgoing interface:
    - **100rel-interworking** option **enabled**
    - **asymetric-preconditions** parameter set to **enabled**
- The SBC proxies the request, and does not perform interworking when it is configured with:
  - Incoming and outgoing interfaces have **asymetric-preconditions enabled**

  - Incoming and outgoing interfaces have **asymetric-preconditions disabled**

In situations when the **INVITE** arrives with no preconditions tags and no precondition SDP attributes, the SBC behaves as listed below.

- The SBC proxies the request, and does not perform interworking when it is configured with:
  - Incoming interface:
    - **100rel-interworking** option **enabled**
    - **asymetric-preconditions** parameter set to **enabled**
  - Outgoing interface:
    - **asymetric-preconditions** parameter set to **disabled**
- The SBC performs nonpreconditions-to-preconditions interworking by inserting the precondition tags and attributes and forwarding when it is configured with:
  - Incoming interface:
    - **100rel-interworking** option **disabled**
    - **asymetric-preconditions** parameter set to **disabled**
  - Outgoing interface:
    - **100rel-interworking** option **enabled**
    - **asymetric-preconditions** parameter set to **enabled**

  The SBC proxies the request, and performs no interworking when it is configured with:
  - Incoming and outgoing interfaces have **asymetric-preconditions enabled**
  - Incoming and outgoing interfaces have **asymetric-preconditions disabled**

Transcoder Free Operation for Asymmetric Preconditions

You can configure the SBC to avoid using transcoding resources while supporting call flows with asymmetric preconditions. After establishing a call that includes transcoding, the SBC can trigger this Transcoder Free Operation (TrFO) feature if the asymmetric preconditions parameter is present in the caller's SDP and a compatible codec can still be identified. Having determined that the call can proceed without transcoding, the SBC originates a reINVITE towards the calling party containing the called side codec. Once the reINVITE is completed, the call can continue without transcoding. The negotiated codec on the called party side must have been included in the calling party's original offer (after ingress codec-policy execution).

When preconditions are enabled on either the caller or callee side and disabled on the other side, the SBC is able to trigger preconditions interworking. The SBC stores all the provisional and final responses from the callee until the preconditions are met. At this point in the call flow, the negotiated codec in the 200 OK may be different than that of the calling side triggering transcoding. To avoid this transcoding, you can configure the SBC to issue a re-INVITE to the caller after call establishment with a compatible codec on the called side. The SBC initiates this re-INVITE only if the following conditions are true:

- Transcoding is induced by asymmetric preconditions
- The negotiated codec on the called side is within the allowed codecs of the calling party's initial offer

To determine whether it can send out this re-INVITE, the SBC retains the codec list sent by the calling party, usually within the initial offer with all supported codecs. It also monitors the result of the ingress **codec-policy** for compatible codecs.

Important limitations to when the SBC can use TrFO with asymmetric preconditions include:

- Although asymmetric preconditions are supported for either side of a call, the SBC only supports TrFO for the caller side. If asymmetric preconditions are applied on the called interface or on both the interfaces, thisTrFO feature does not work.
- This feature does not apply to call flows with offerless INVITEs.
- The TrFO for ringback and asymmetric preconditions features can conflict with each other. If there is a configuration wherein a session agent has ringback enabled, and TrFO for Asymmetric Preconditions is enabled on that session agent's realm, the configuration is invalid and, thisTrFO feature does not work.
- The SBC does not fully support TrFO when you configure the **feature-trfo** parameter with both **ringback** and **asymmetric-preconditions**. If you configure both **ringback** and **asymmetric-preconditions** on the ingress and egress realms, such that the system is expected to perform both functions simultaneously on opposite ends of a call, the TrFO would be applicable only to one of the features.

Configuration

This functionality requires that you configure the EVS WB media profile. In addition, you must configure the initial INVITE ingress realm with the **feature-trfo** parameter using the syntax below.

```
ORACLE(realm-config)# feature-trfo asymmetric-preconditions
```

Under the conditions described in this section, this configuration allows the SBC to force SDP re-negotiation and prevents the call from requiring transcoding.

As mentioned above, although you can configure the **feature-trfo** parameter with multiple parameters, the SBC only acts on one of those parameters at any give time. Under the condition where more than one parameter applies, the SBC refers to your configuration's parameter order to determine which function to perform:

- If the configuration order appear with ringback first, the SBC applie TrFO for ringback. (**feature-trfo : ringback,asymmetric preconditions**).
- If the configuration order appear with asymmetric preconditions first, the SBC applie TrFO for asymmetric preconditions. (**feature-trfo : asymmetric preconditions, ringback**).

TrFO Statistics for Asymmetric Preconditions

The **show sipd preconditions-trfo** command shows Asymmetric Preconditions TrFO statistics:

- Initiated: number of initiated TRFO reINVITE transactions
- Success: number of successfully completed TRFO reINVITE transactions
- Failure: number of failed TRFO reINVITE transactions (either error returned or timed out)

```
ORACLE# show sipd preconditions-trfo
 Asymmetric Preconditions TrFO statistics (2011-09-24 12:03:37.051)
             ---- Lifetime ----
            Recent Total PerMax
 Initiated    0      3      2
 Success      0      3      2
 Failure      0      0      0
```

You can reset this data back to zero using the **reset sipd** command.

Asymmetric Preconditions and EVS using Super Wide-band

The SBC supports transcoding the EVS codec, but not when the bandwidth presented is SWB, NB-SWB, or WB-SWB. When presented with these within asymmetric preconditions call flows, the SBC performs an "internal transcode" using the EVS WB instead of the EVS-SWB codec. This works because a UE that supports EVS-SWB also supports EVS-WB. The SBC performs this function for asymmetric preconditions call flows only.

To support EVS-SWB for within an asymmetric precondition call flow, the SBC monitors initial INVITEs for the EVS codec with SWB in the offer/answer SDP. If present, the system maps the bandwidth parameter from SWB to WB/NB and updates the media flows. Once media flows are updated, and before sending out the request/response, the SBC uses the original offer to compare and revert the bandwidth parameter back to support the caller or callee.

The SBC performs this mapping when triggered by the caller or callee. This could be the initial INVITE from the caller as well as the 200 OK from the callee. Internally, the SBC maps these to evs-wb and, before sending out the request or response, replaces them with evs-swb, in accordance with the stored offer.

Supported methods include INVITE, UPDATE and PRACK within the context of asymmetric preconditions call flow. The system maps EVS-SWB bandwidth as follows:

- bw=nb-swb mapped to bw=nb-wb
- bw=wb-swb mapped to bw=wb
- bw=swb mapped to bw=wb

You configure this behavior with the following, in addition to the TrFo configuration.

```
ORACLE(sip-config)# asymm-preconditions-evs-swb-support enabled
```

Important considerations while deploying this feature include:

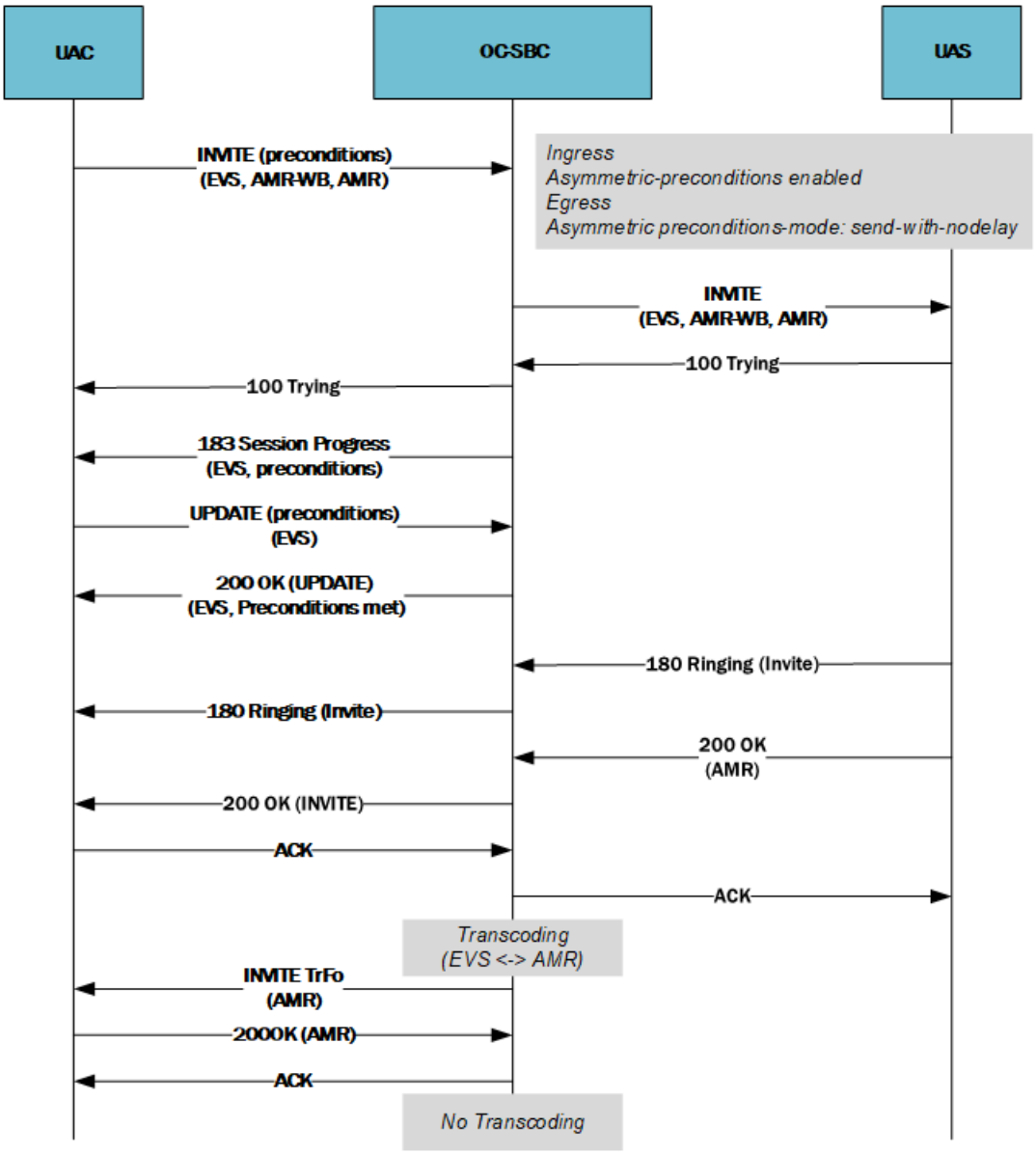
- The SBC does not support EVS-SWB within codec policy, because it does not encode EVS-SWB packets.
- If the environment changes the codec's payload type before forwarding the request, the SBC still tries to negotiate using wideband bandwidth. This is because , while reverting the bw parameter, the SBC continues to use the original SDP, performing the inherent operation by referring to the payload type.
- If and end-station issues EVS-SWB within a re-INVITE, such as when the caller rejects the TrFO re-INVITE, the SBC does not perform this mapping function. The SBC only negotiates preconditions before the call is established, and clears them after it receives the final response for the INVITE.

Call Flows for Asymmetric Preconditions TrFo

The image below depicts the SBC performing TrFo with asymmetric preconditions, wherein the egress mode is **send-with-nodelay**.

1. The UAC issues an invite with SDP that offers EVS, AMR-WB and AMR.
2. The SBC, supporting asymmetric preconditions, forwards this list to the UAS, per SBC configuration. Note the use of the**send-with-nodelay** mode causes the SBC to immediately send the INVITE.
3. The SBC proceeds with confirming preconditions with the UAC.

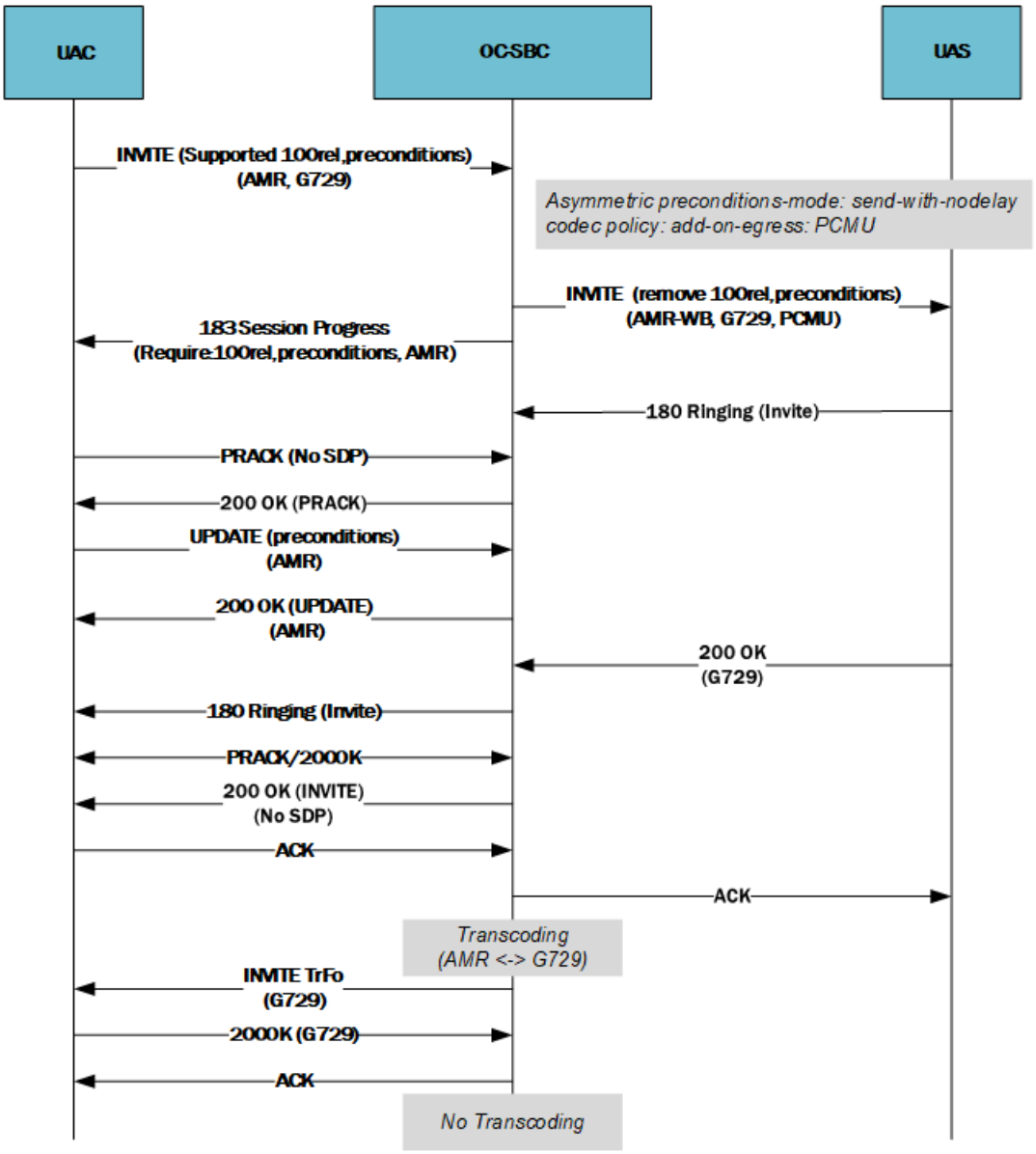
4. In the meantime, the UAS proceeds with agreeing to the call using AMR.
5. The UAC and SBC confirm the call with transcoding.
6. Immediately after confirming the call, the SBC triggers its TrFO function, changing the agreed upon codec to AMR with the UAC.
7. The call proceeds without transcoding (TrFo).



The image below depicts the SBC performing TrFo with asymmetric preconditions and PRACK interworking, wherein the egress mode is **send-with-nodelay**.

1. The UAC issues an invite with SDP that offers AMR and G729.
2. The SBC, supporting asymmetric preconditions, forwards this list to the UAS, per SBC configuration. Note the use of the **send-with-nodelay** mode causes the SBC to immediately send the INVITE.
3. The SBC proceeds with confirming preconditions with the UAC. This process specifies AMR as the codec agreed upon by the UAC and the SBC.
4. In the meantime, the UAS proceeds with agreeing to the call using G729.

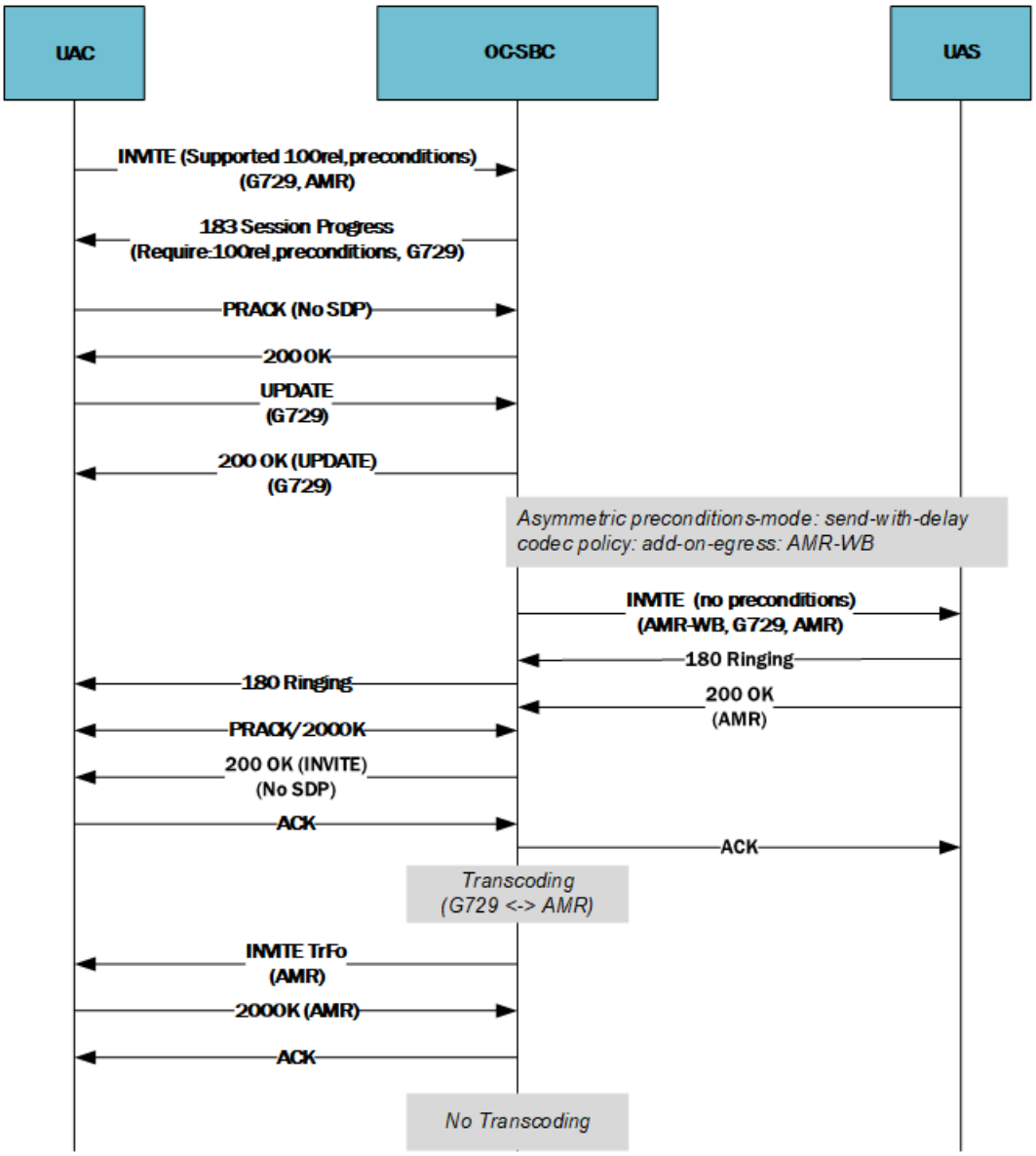
5. The UAC and SBC confirm the call with transcoding.
6. Immediately after confirming the call, the SBC triggers its TrFO function, changing the agreed upon codec to G729 with the UAC.
7. The call proceeds without transcoding (TrFo).



The image below depicts the SBC performing TrFo with asymmetric preconditions, wherein the egress mode is **send-with-delay**.

1. The UAC issues an invite with SDP that offers AMR and G729.
2. The SBC, supporting asymmetric preconditions, forwards this list to the UAS, per SBC configuration. Note the use of the **send-with-delay** mode causes the SBC to hold the INVITE.
3. The SBC proceeds with confirming preconditions with the UAC. This process specifies G729 as the codec agreed upon by the UAC and the SBC.
4. After establishing preconditions and agreeing the G729 with the UAC, the SBC forwards the INVITE.

5. The UAS proceeds with agreeing to the call using AMR.
6. The UAC and SBC confirm the call with transcoding.
7. Immediately after confirming the call, the SBC triggers its TrFO function, changing the agreed upon codec to AMR with the UAC.
8. The call proceeds without transcoding (TrFo).

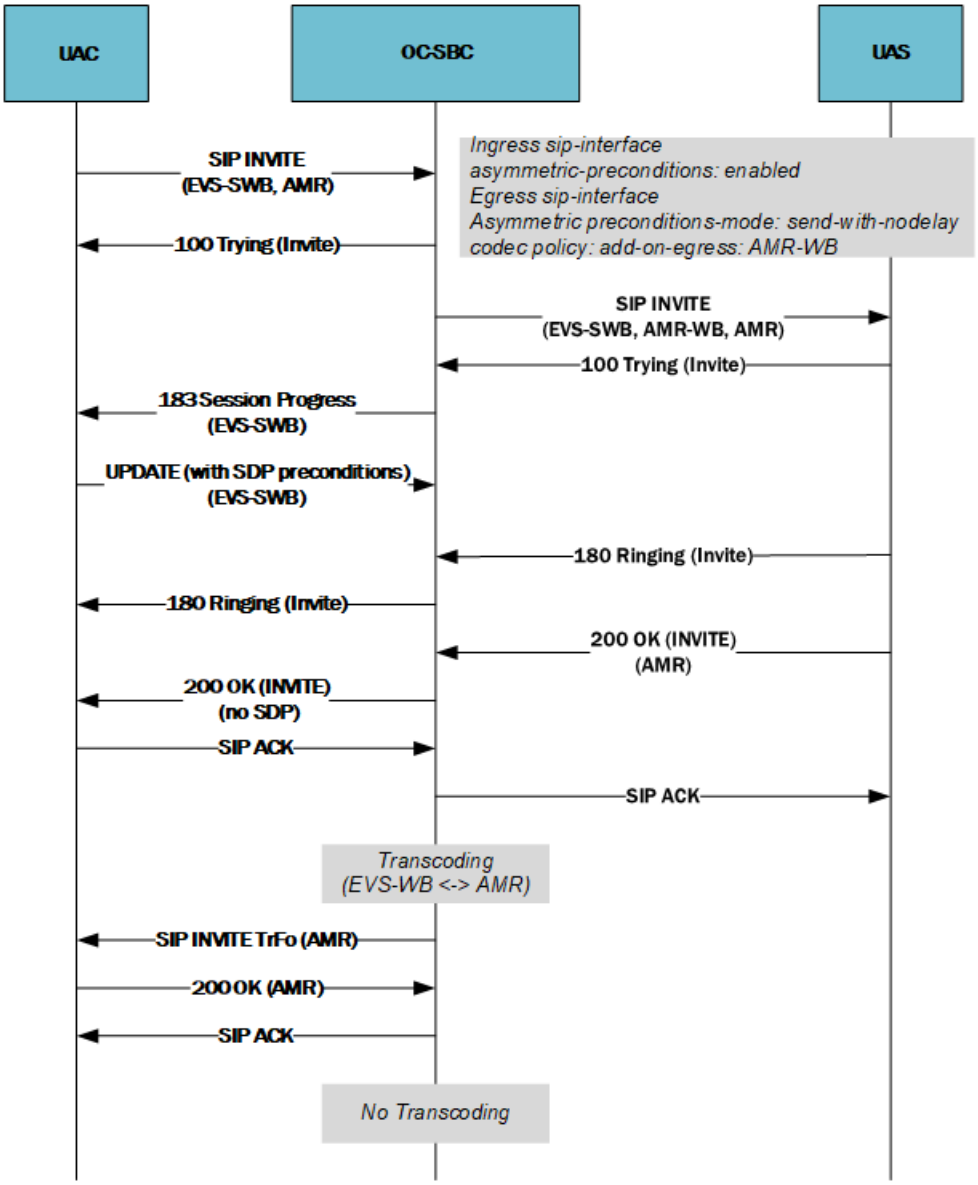


The image below depicts the SBC performing a simple TrFo process for a call flow with asymmetric preconditions. In addition, you have enabled the **asymm-preconditions-evs-swb-support** parameter in the **sip-config**.

1. The UAC issues an invite with SDP that offers EVS-SWB and AMR. The SBC, supporting asymmetric preconditions, forwards to the UAS adding AMR-WB to the offered list, per SBC configuration.
2. The UAS continues with the call while the SBC exchanges an UPDATE with the UAC.
3. As the UAS proceeds with accepting the call using AMR, the SBC reverts from EVS-SWB to EVS-WB and proceeds with the call using transcoding. Note the enabled **asymm-**

preconditions-evs-swb-support parameter allows the SBC to use EVS wideband instead of super-wideband without transcoding.

4. The SBC recognizes AMS as an option and issues a re-INVITE, as described, the establish the leg using AMR.
5. Once the leg is confirmed for AMR, the call proceeds without transcoding (TrFo).



Dynamic Preconditions

You can configure the SBC to extend its support of preconditions with dynamic preconditions, which allows the SBC to determine whether and where to support preconditions for a given call. Assuming proper configuration, the SBC refers to the INVITE from the caller and the 18x response to the initial INVITE from the callee. The INVITE informs the system on preconditions support within the caller's network. The 183 response informs the SBC whether the callee supports preconditions. The information in these messages trigger the system to support end-to-end, asymmetric, or no preconditions for a given call.

You configure the system to support dynamic preconditions by:

- Enabling the **precondition-enhancement** parameter within the **sip-config**
- Setting the **asymmetric-preconditions-mode** parameter on the called side **sip-interface** to **send-without-delay**
- Enabling **asymmetric-preconditions** on the **sip-interface** for both call legs
- Enabling the **100rel-interworking** option on the **sip-interface** for both call legs

With dynamic preconditions configured, the system is able to establish:

- End-to-end preconditions support, wherein the SBC transparently passes preconditions values between the caller and callee. In this case, the end stations perform preconditions negotiation with each other.
- Leg-by-leg preconditions, wherein one side of the call requests preconditions and the other side does not. In this case, the SBC performs preconditions negotiation with the side that supports it, and performs preconditions interworking for the call. This effectively establishes an asymmetric preconditions call flow.
  For asymmetric preconditions, the SBC performs preconditions interworking by stripping out preconditions information before forwarding the applicable message(s) to the unsupported side.

The SBC recognizes the attempt to establish preconditions by:

- The caller - based on the presence or absence of the preconditions tag in the initial INVITE.
- The callee - based on the presence or absence of the preconditions tag in the 183 Progress message.

The SBC always supports any preconditions requested by an end station. This means that preconditions negotiation within asymmetric preconditions call flows between the SBC and an end station consists of the SBC agreeing to all preconditions.

> **Note:**
>
> For cases involving an offer-less INVITE from the caller, meaning the INVITE includes the 100rel and preconditions option tags but no SDP, the system relays preconditions to the caller if the callee supports them and adds preconditions towards the caller if the callee does not support them.

Whenever the calling party (UAC) supports preconditions, it includes a preconditions tag in the INVITE to the SBC. If not, there is no tag. Regardless of the calling party's preconditions status, the SBC forwards the INVITE to the called party (UAS) with a preconditions tag in all cases where you have configured dynamic preconditions:

- When the Called party (UAS) supports preconditions, and:
  - The Calling party supports preconditions, then after the SBC forwards its INVITE:
    1. The UAS send 18x with preconditions to the SBC
    2. The SBC PROXIES the received messages
  - The Calling party does not support preconditions, then after the SBC forwards its INVITE:
    1. The UAS sends a 18x with preconditions to the SBC
    2. The SBC performs precondition negotiation towards the UAS
- When the Called party (UAS) does not support preconditions, and:

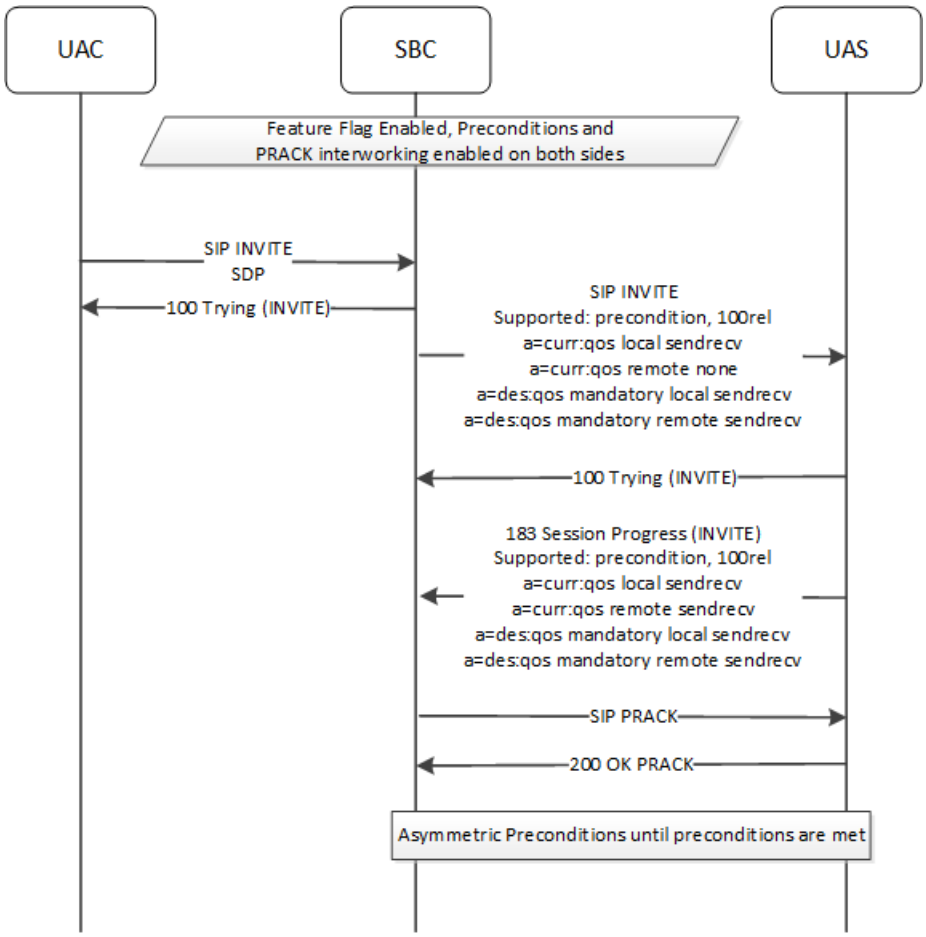
- The Calling party supports preconditions, then after the SBC sends its INVITE:
  1. The UAS sends a 18x without preconditions to the SBC
  2. The SBC performs precondition negotiation towards the UAC
- The Calling party does not support preconditions, then after the SBC adds preconditions attributes and forwards its INVITE:
  1. The UAS sends a 18x without preconditions to the SBC
  2. The SBC PROXIES the received messages

Dynamic Preconditions Call Flows

Key call flows for understanding SBC operation within dynamic preconditions contexts include depicting the calling and called network either supporting or not supporting preconditions. For all flows, the SBC has the required configuration for dynamic preconditions set on both sides of the call.

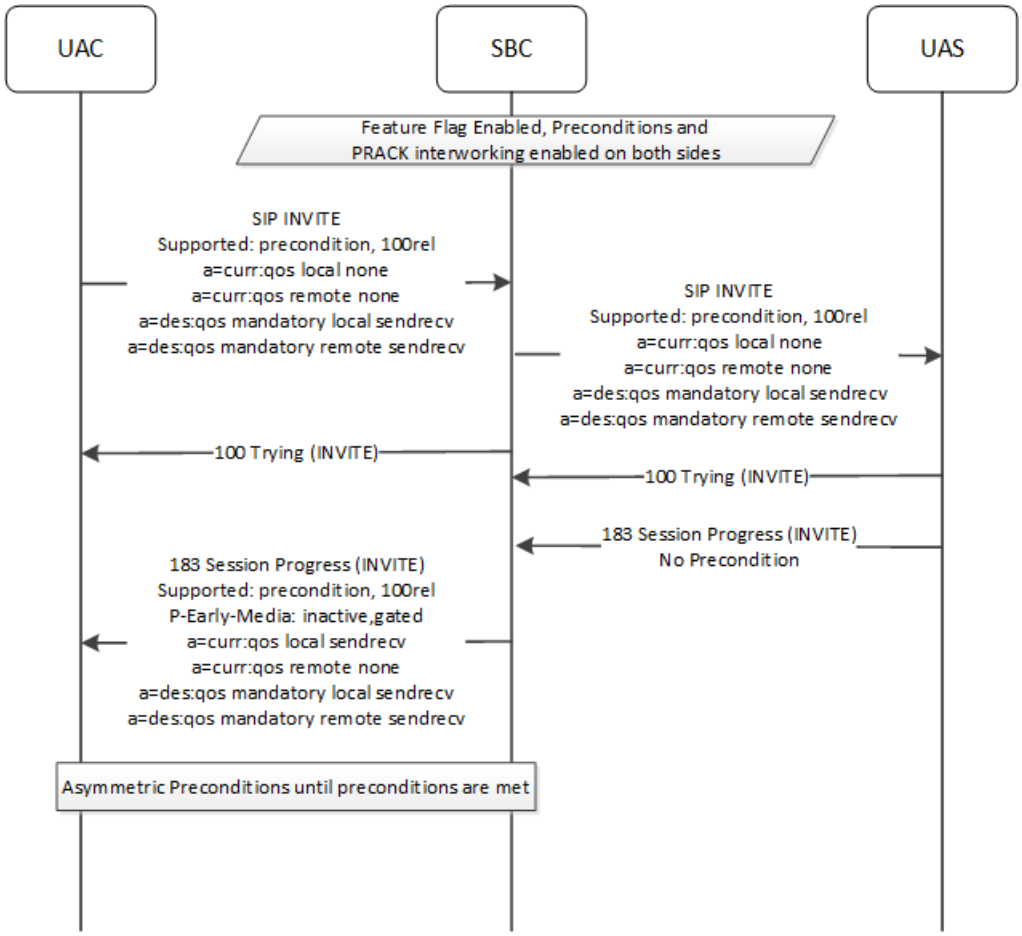
Calling Network Does Not Support Preconditions - Called Network Does

In this flow, the UAC sends an INVITE towards the SBC that does not specify preconditions. Because the system has dynamic preconditions configured on both sides, it offers preconditions to the UAS. The UAS accepts and the system establishes asymmetric preconditions for the flow.



Called Network Does Not Support Preconditions - Calling Network Does

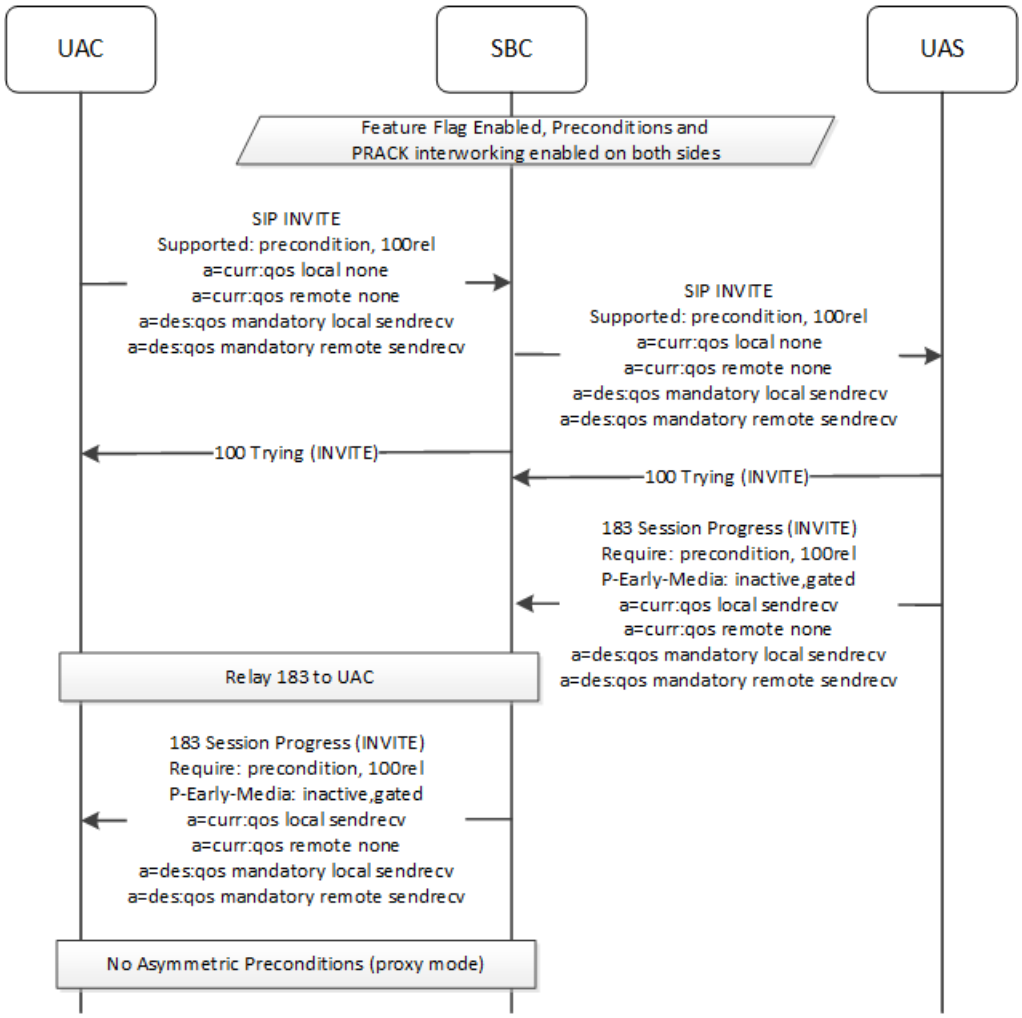
In this flow, the caller's network supports preconditions, so the caller presents its preconditions request within the initial INVITE. Because the UAC's side includes dynamic preconditions, the SBC sends an INVITE specifying the desired parameters. The UAS responds with a 183 that does not include preconditions, which signals to the SBC that there is no support in that network. Given the UAC has requested preconditions, the SBC complies and sets up the flow for asymmetric preconditions.



> **Note:**
>
> Notice that the SBC includes early media gated. This is a result of the behavior wherein the SBC inserts the inactive early media direction attribute, as described below.

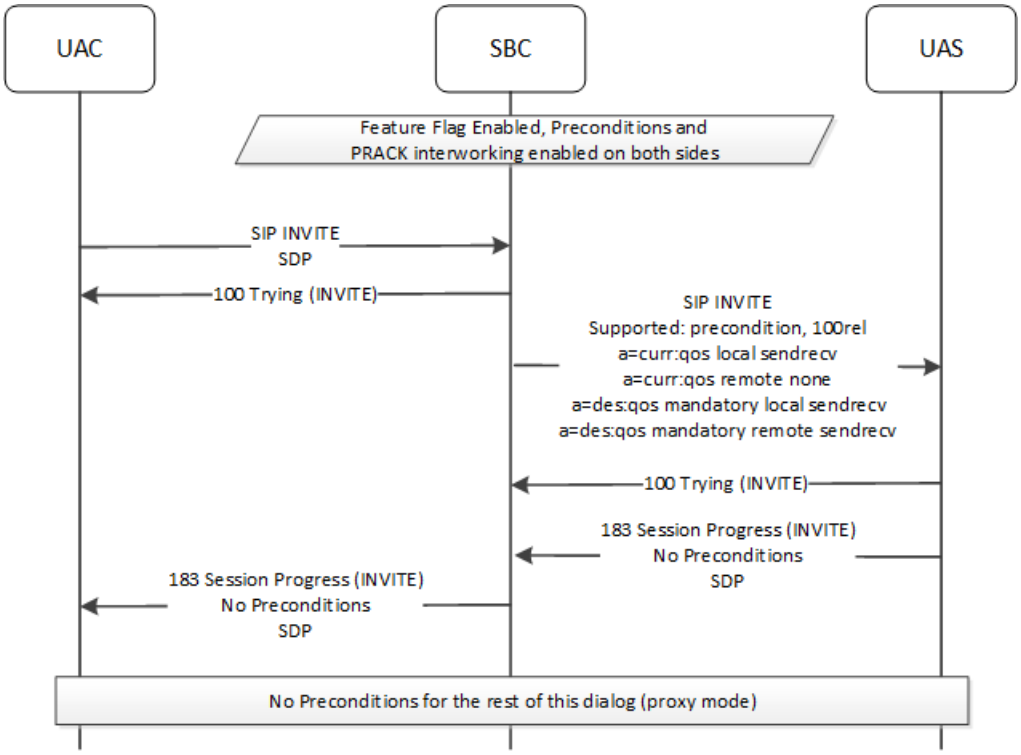
Both Calling and Called Network Supports Preconditions

Both networks in this next flow support preconditions. As described earlier, the SBC recognizes this by the presence of preconditions in the INVITE from the UAC and the 183 from the UAS. As described above, when both network sides of a call support preconditions, the SBC acts as proxy for preconditions, in the case below, by simply forwarding the 183 from the UAS to the UAC (proxy mode).



Neither Calling or Called Network Supports Preconditions

In this flow, the UAC sends an INVITE towards the SBC that does not specify preconditions. Because the system has dynamic preconditions configured on both sides, it offers preconditions to the UAS. The UAS network, however also does not support preconditions, so the system establishes the flow without preconditions on either side.



Early Media Gating using Locally Generated 183

When you enable the **precondition-enhancement** parameter within the **sip-config**, the SBC can manipulate the PEM header within both static asymmetric and dynamic preconditions call flows to change the direction attributes.

Within asymmetric preconditions flows, the SBC by default replies to the calling party's initial INVITE with a "183 Session Progress" response that includes SDP with preconditions and a PEM header set to sendrecv. This can be mistakenly interpreted by the calling party and intermediary nodes as an indication of early media. When you enable the **precondition-enhancement** parameter, the SBC sets the level of early-media authorization in the PEM header in this and other responses to a value less likely to be misinterpreted.

This behavior differs if the preconditions flows are static asymmetric or dynamic:

- When static, the SBC generates the applicable 183 message to the caller independently.
- When dynamic, the SBC refers to the callee's 183 response before it generates its own, applicable 183 to the caller.

In addition to requiring the **precondition-enhancement** configuration, this feature also refers to **p-early-media-direction** parameter on the **sip-interface** facing the calling party.

Applicable preconditions configurations include:

- Static Asymmetric Preconditions—when the calling party supports preconditions, the SBC sets the PEM header in the 183 response it generates towards the calling party to:
  - “inactive”, if **p-early-media-direction** is not configured
  - The configured value, if **p-early-media-direction** is configured
- Dynamic Preconditions—When the calling party supports preconditions but the called party does not, the SBC sets the PEM header in its 183 response to the calling party based on whether the response from the called party includes a PEM header:

- If not, the SBC sets the PEM value in its 183 response towards the calling party to:
  - “inactive”, if **p-early-media-direction** is not configured
  - The configured value, if **p-early-media-direction** is configured
- If so, the SBC sets the PEM value its 183 response towards the calling party as described in the tables below.

The tables below present the logic the SBC uses to set the PEM header in locally generated 183 based on various configurations.

The table below maps PEM with preconditions settings within flows from a trusted caller to a trusted callee.

Message Parameters configured on egress interface	**Precondition Ingress Interface**	**Precondition Egress Interface**	**Response (18x) w/o PEM header**	**Response (18x) with PEM header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", “recvonly", or "inactive"**
p-early-media-header: "add/ modify" **p-early-media-direction**: "sendonly"	Enabled	Enabled	modify or add: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated	add: Populate PEM header based on the value in the incoming PEM header. modify: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated
p-early-media-header-"add/ modify" No p-early-media-direction	Enabled	Enabled	modify or add: Populate PEM header with default “inactive” direction. That is, P-Early-Media: inactive, gated	modify or add: Populate PEM header based on the value in the incoming PEM header
p-early-media-header-"add/ modify" p-early-media-direction -"sendonly"	Enabled	Disabled	modify or add: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated	modify or add: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated

Message Parameters configured on egress interface	Precondition Ingress Interface	Precondition Egress Interface	Response (18x) w/o PEM header	Response (18x) with PEM header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", “recvonly", or "inactive"
p-early-media-header-"add/ modify" No p-early-media-direction	Enabled	Disabled	modify or add: Populate PEM Header with default “inactive” direction. That is, P-Early-Media: inactive, gated	modify or add: Populate PEM Header with default “inactive” direction. That is, P-Early-Media: inactive, gated

The table below maps PEM with preconditions settings within flows from a trusted caller to an untrusted callee.

Message Parameters configured on egress interface	Precondition Ingress Interface	Precondition Egress Interface	Response (18x) w/o header	Response (18x) with PEM header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or "inactive"
p-early-media-header-"add/ modify" p-early-media-direction -"sendonly"	Enabled	Enabled	modify or add: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated	modify or add: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated
p-early-media-header-"add/ modify" No p-early-media-direction	Enabled	Enabled	modify or add: Populate PEM Header with default “inactive” direction. That is, P-Early-Media: inactive, gated	modify or add: Populate PEM Header with default “inactive” direction. That is, P-Early-Media: inactive, gated

Message Parameters configured on egress interface	Precondition Ingress Interface	Precondition Egress Interface	Response (18x) w/o header	Response (18x) with PEM header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or "inactive"
p-early-media-header-"add/modify" p-early-media-direction -"sendonly"	Enabled	Disabled	modify or add: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated	modify or add: Populate PEM Header with configured PEM direction at egress interface. That is, P-Early-Media: sendonly, gated
p-early-media-header-"add/modify" No p-early-media-direction	Enabled	Disabled	modify or add: Populate PEM Header with default “inactive” direction. That is, P-Early-Media: inactive, gated	modify or add: Populate PEM Header with default “inactive” direction. That is, P-Early-Media: inactive, gated

Changing the Media Direction Attribute

When you enable the **precondition-enhancement** parameter within the **sip-config**, system behavior changes for certain preconditions call flows wherein the SBC changes the direction value of the SDP media attribute to prevent issues.

Direction value changes, based on scenario include:

- For static asymmetric preconditions flows with mode set to **send-without-delay** or dynamic preconditions flows, the SBC sets the SDP direction attribute to sendrecv in the outgoing initial INVITE if the initial INVITE from the calling party has the direction attribute set to inactive.
  This ensures that the call does not remain inactive on the callee side.
- For dynamic preconditions flows where the calling party supports preconditions but the called party does not, the SBC sets the direction attribute to “inactive” when it generates a 183 response locally towards the calling party.

These behaviors allow you to avoid the use of HMR to change the direction attribute in the SDP at both the media and session level.

Strength Tag Support

By default, the SBC rejects calls that include optional preconditions tags, and it does not insert strength tags under any conditions. When you enable the **precondition-enhancement** parameter within the **sip-config** however, the SBC supports all of the strength tag values within all preconditions attributes. In addition, the SBC inserts strength tags under certain conditions.

The strength tag is a preconditions attribute value that specifies how strictly an end-station requires the establishment of preconditions for a call. The strength tag has three values:

- None—where no resource reservation is needed
- Optional—where the user agents should try to provide the requested resources
- Mandatory—where the call is terminated if the resources cannot be provided

This value is included in an end-station's status table to further define qos attributes. Ultimately, end-stations can use the strength tag's value in conjunction with the status-type and direction tag values to determine whether or not to proceed from preconditions negotiation to ringing.

When you enable the **precondition-enhancement** parameter, the SBC adds the following behaviors to its preconditions processing:

- If the SBC receives an optional strength tag while performing preconditions interworking, it treats the precondition as if it is mandatory.
- Whenever it generates a strength tag, the SBC generates and inserts mandatory strength tags to process:
  - Static or dynamic preconditions flows wherein the calling party supports preconditions and has provided optional strength tags in the initial INVITE, but the called party does not support preconditions. In this case, the SBC inserts mandatory strength tags into its locally generated 183 toward the calling party.
  - Dynamic preconditions flows wherein the calling party does not support preconditions in the initial INVITE. In this case, the SBC inserts mandatory strength tags into the INVITE it forwards to the called party.
- When processing dynamic preconditions flows wherein the calling party supports preconditions but has provided strength tags in the initial INVITE, the SBC:
  - Relays the preconditions with strength tags unchanged in messages back to the calling party
  - If the called party also supports preconditions, relays the preconditions with strength tags unchanged in messages to the called party

Enhanced Preconditions Configuration

You can configure the SBC with additional preconditions functionality by enabling the **precondition-enhancement** parameter within the **sip-config**, including:

- Dynamic Preconditions
- PEM Gating using Locally Generated 183
- Changing the Media Direction Attribute
- Supporting the Optional Strength Tag

To configure the SBC to perform the functions listed above:

1. From superuser mode, use the following command sequence to access sip-config configuration mode. While in this mode, you enable enhanced preconditions functions.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. Use the **precondition-enhancement** parameter to enable enhanced preconditions functions.

```
ORACLE(sip-config)#precondition-enhancement enabled
```

3. Use **done**, **exit**, and **verify-config** to complete SAG-based address resolution.

Preconditions and Multiple Early Dialogs

You can configure the SBC with the enhancement described here to overcome limitations in call flows that use the Multiple Early Dialogs (MED) function in conjunction with multiple early dialog merging, preconditions, preconditions interworking and/or transcoding free operation (TrFO). To support the use of MED with preconditions, the SBC implements preconditions processing at the dialog level.

This functionality is based on the *Early Media Support for Multiple Early Dialog Scenarios* and *Asymmetric Preconditions* features, described in this ACLI Configuration Guide. Refer to that documentation for further, applicable background. This documentation includes multiple call flows that depict the key enhancements.

An example of these enhancements includes support for static or dynamic preconditions in conjunction with multiple early dialogs generated by call forking towards multiple devices on the call termination side.

You configure this support by enabling the **precondition-med-enhancement** parameter within the **sip-config**. When enabled, the SBC supports:

- Preconditions with MED merge
- Dynamic Asymmetric Preconditions with TrFO

This support addresses specific precondition with MED merge scenarios, including:

- Static Asymmetric Preconditions with MED merge, with and without delay
- Static Asymmetric Preconditions with MED and TrFO, with and without delay
- Dynamic Preconditions with MED merge
- Dynamic Preconditions with TrFO

In addition, the SBC also includes preconditions in the offer of the resulting INVITE when it receives a 3xx, 4xx, or 5xx error message on the side supporting preconditions for all the scenarios above.

The SBC provides confirmation for dialogs requesting confirmation in the following scenarios:

- Multiple devices on the terminating side resulting into multiple early dialogs from a forking proxy
- Sequential forking initiated by the SBC

Related Configuration

In addition to enabling **preconditionMedEnhancement**, this feature is designed to operate with the following parameters enabled:

- **asymmetric-preconditions**, on the access and core **sip-interface**
- The **100-rel interworking** option, on the access and core **sip-interface**
- **asymmetric-precondition-mode**, on the core **sip-interface**

- **precondition-enhancement** in the **sip-config**
- **multiple-dialogs-enhancement** in the **sip-config**
- **merge-early-dialogs** on the access and core **realm-config**
- **hide-egress-media-update** on the access and core **realm-config**

In addition, you must configure transcoding for call flows using TrFO.

> **Note:**
>
> This MED precondition enhancement feature is not compatible with the **Drop-Additional-Provisional** SPL Option.

Precondition and MED merge Functional Overview

This section presents the SBC supporting call flows with preconditions signaling within the context of merged multiple early dialogs. Multiple dialog merging requires early dialog support in addition to enabling the **merge-early-dialogs** parameter in the caller's realm.

If you do not enable **precondition-med-enhancement** with static or dynamic preconditions, the SBC uses the first dialog it receives to perform preconditions negotiation. It does not include any subsequent forked dialogs in preconditions IWF.

Consider a SIP dialog with early media that proceeds by establishing call scenarios in which the final 200 OK does not include any SDP. This messaging may include multiple 183 (Session Progress) messages with SDP that differ from each other and include different To tags, establishing multiple early dialogs. In these scenarios, the SBC uses the saved SDP from the dialog that matches the dialog indicated in the 200 OK's To tag to anchor the media.

For both static and dynamic preconditions with MED flows, the SBC uses this feature to enhance support by:

- Forwarding calls and accepting multiple 18x messages with different To tags. Based on the To tags, the SBC recognizes a multiple early dialog scenario, which may or may not be established by multiple endstations that could be the first callee. These 18xs may also include different SDP.
- Saving information on all dialogs. This allows the SBC to proceed with the call under a variety of conditions, including when the final 200 OK does not include SDP.
- Generating PRACKs locally for all forked responses to 18x message that include 100rel.
- Updating MBCD when it receives SDP from each forked terminal.
- Generating UPDATEs locally towards all the dialogs requesting confirmation to complete the preconditions negotiation between all subsequent forked destination and itself.

When you enable **precondition-med-enhancement**, the SBC modifies its behavior for call flows with dynamic preconditions and MED merge mode to include:

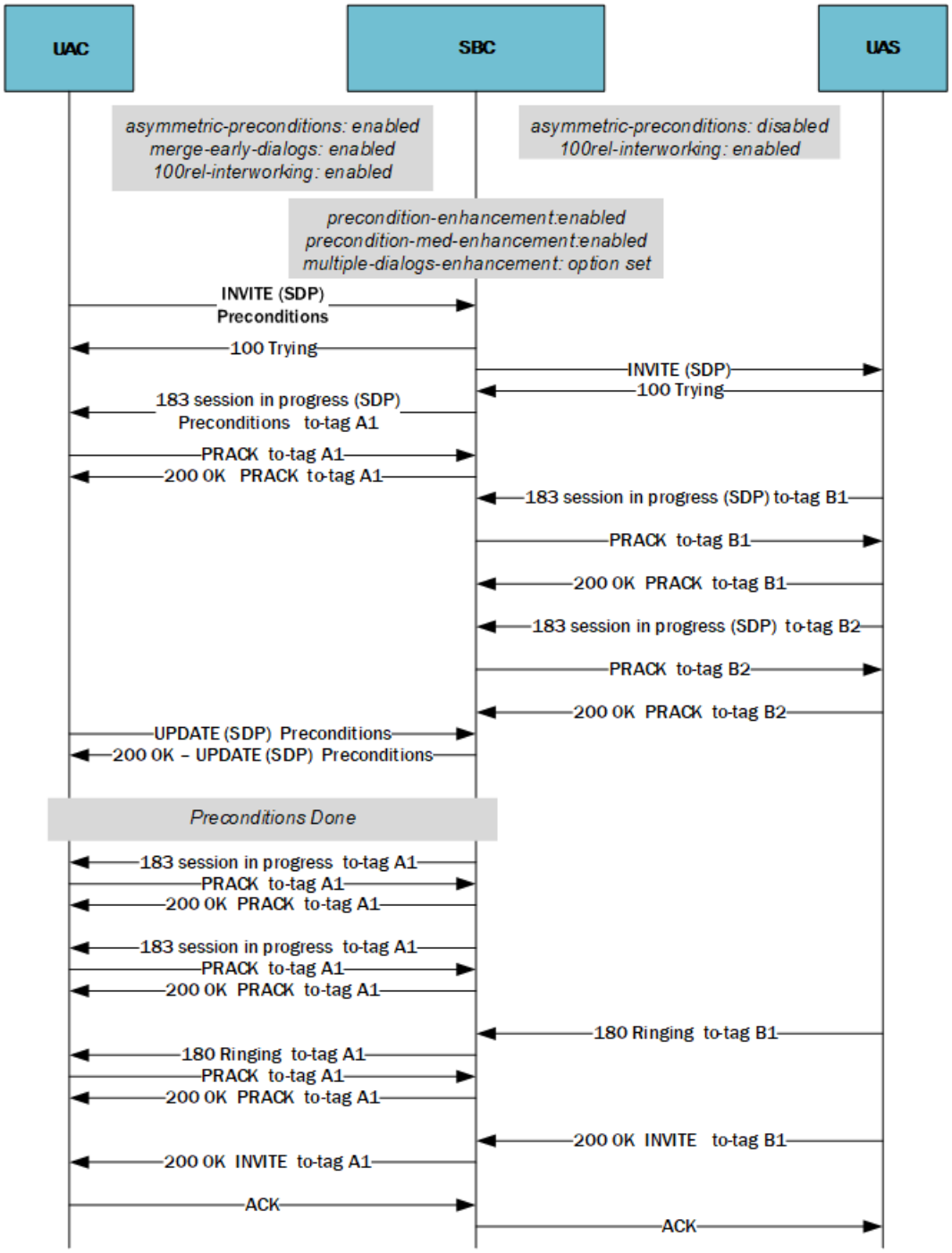
- In the case of dynamic asymmetric preconditions where the calling party supports preconditions and the first terminating called party does not, the SBC uses the locally generated to-tag that was generated for the first 183 and starts preconditioning IWF towards the UAC in merge mode.
- In the case of dynamic preconditions, where both the calling and first terminating called parties support the precondition, the SBC transparently passes the first 183 to the calling side and conducts end-to-end precondition negotiation.

- Also, in this end-to-end precondition case, the SBC never maintains the precondition state machine. Instead, it passes all 18x and 2xx responses without changes towards the calling side on the single merge dialog before precondition completion. This may result in undesirable behavior in cases where a forked UAS, for example B2, sends 18x/2xx before precondition completion between the caller and B1, and when resource reservation is mandatory.

The next table presents a composite overview of these principles when you enable asymmetric preconditions on both the ingress and egress interfaces.

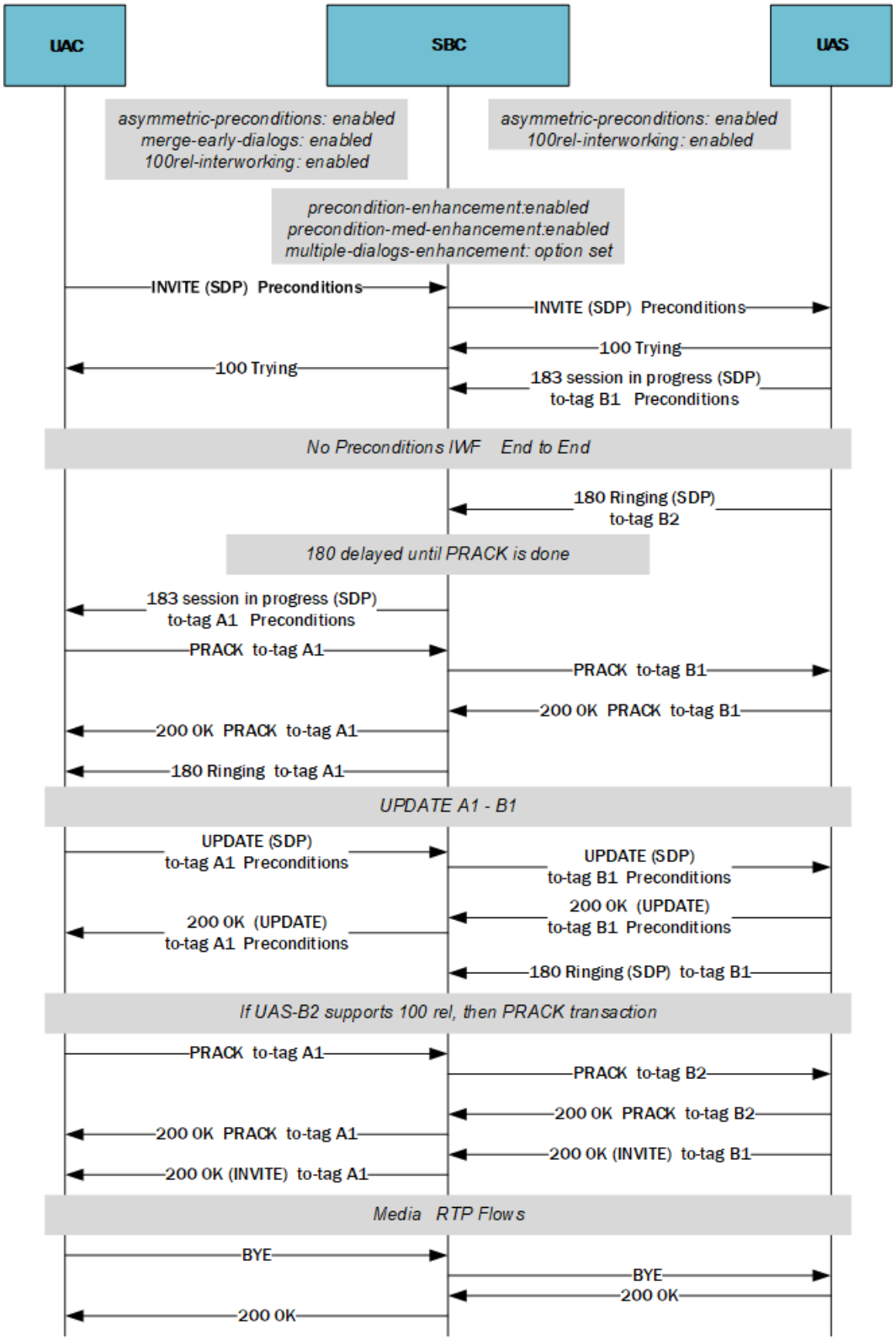
First terminating 18x received	Calling party doesn't support preconditions	Calling party supports preconditions
Without precondition support	Incoming INVITE received without preconditions tag, SBC sends INVITE with preconditions tag towards UAS UAS send 18x without preconditions	Incoming INVITE received with preconditions tag, SBC sends INVITE with preconditions tag towards UAS UAS side sends 1st 18x without preconditions
	SBC will simply PROXY received messages from B1.	SBC will perform precondition IWF negotiation towards UAC.
	SBC will do precondition IWF complete it locally towards all subsequent forked destinations if precondition required in 18x.	SBC will do precondition IWF and complete it locally towards all subsequent forked destinations if precondition required in 18x.
With precondition support	Incoming INVITE received without preconditions tag, SBC sends INVITE with preconditions tag towards UAS UAS send 18x with preconditions	Incoming INVITE received with preconditions tag, SBC sends INVITE with preconditions tag towards UAS UAS sends 1st 18x with preconditions
		SBC will simply PROXY received messages from first terminating dialog.
	SBC will perform precondition IWF negotiation towards UAS and send UPDATE to confirm precondition to each forked dialog.	SBC will do precondition IWF and complete it locally towards all subsequent forked destinations if precondition required in 18x.

The call flow below shows the basic signaling required for the SBC to support static preconditions in conjunction with MED while initiating the call to the UAS. Note that the SBC supports preconditions below in no delay mode. The UAS responds with two early dialogs, which the SBC supports using PRACKs. Having completed preconditions negotiation with the UAC, the SBC merges the two dialogs, presenting them to the UAC, which provides PRACK acknowledgments. Finally, the B1 dialog from the UAS signals ringing and 200 OK to the SBC, which conveys the call setup to the UAC.



This next call flow shows the signaling required for the SBC to support dynamic preconditions in conjunction with MED. Again, the SBC imposes no delay during with the call setup while it performs preconditions negotiation. In this case, the UAS supports preconditions allowing the SBC to proxy end-to-end preconditions setup.

The SBC supports preconditions negotiation with the UAC while initiating the call to the UAS. The UAS responds with two early dialogs which the SBC supports using PRACKs. Dialog B1 supports preconditions, but B2 does not. The SBC ignores a 180 ringing from B2, instead proceeding with support for end-to-end preconditions with B1. Once the preconditions negotiation between the UAC and the UAS (B1) is complete, the SBC accepts a 180 ringing from dialog B1 and sets up the call.



Static and Dynamic Preconditions with TrFO

This section presents the SBC supporting call flows using the TrFO function within the context of multiple early dialogs scenarios that also support preconditions. This feature also requires

that you set the **trfo-feature** parameter on the ingress realm to **asymmetric-preconditions**. Note that this excludes TrFO for ringback on that realm.

For static preconditions flows, when you enable preconditions on the caller side and disable it on the other side, the SBC triggers its preconditions IWF. Within these contexts the SBC stores provisional and final responses from all callees until the preconditions are met. Similarly, when you enable dynamic preconditions, the SBC triggers preconditions IWF towards the caller if the first 18x from a forked response does not include preconditions support.

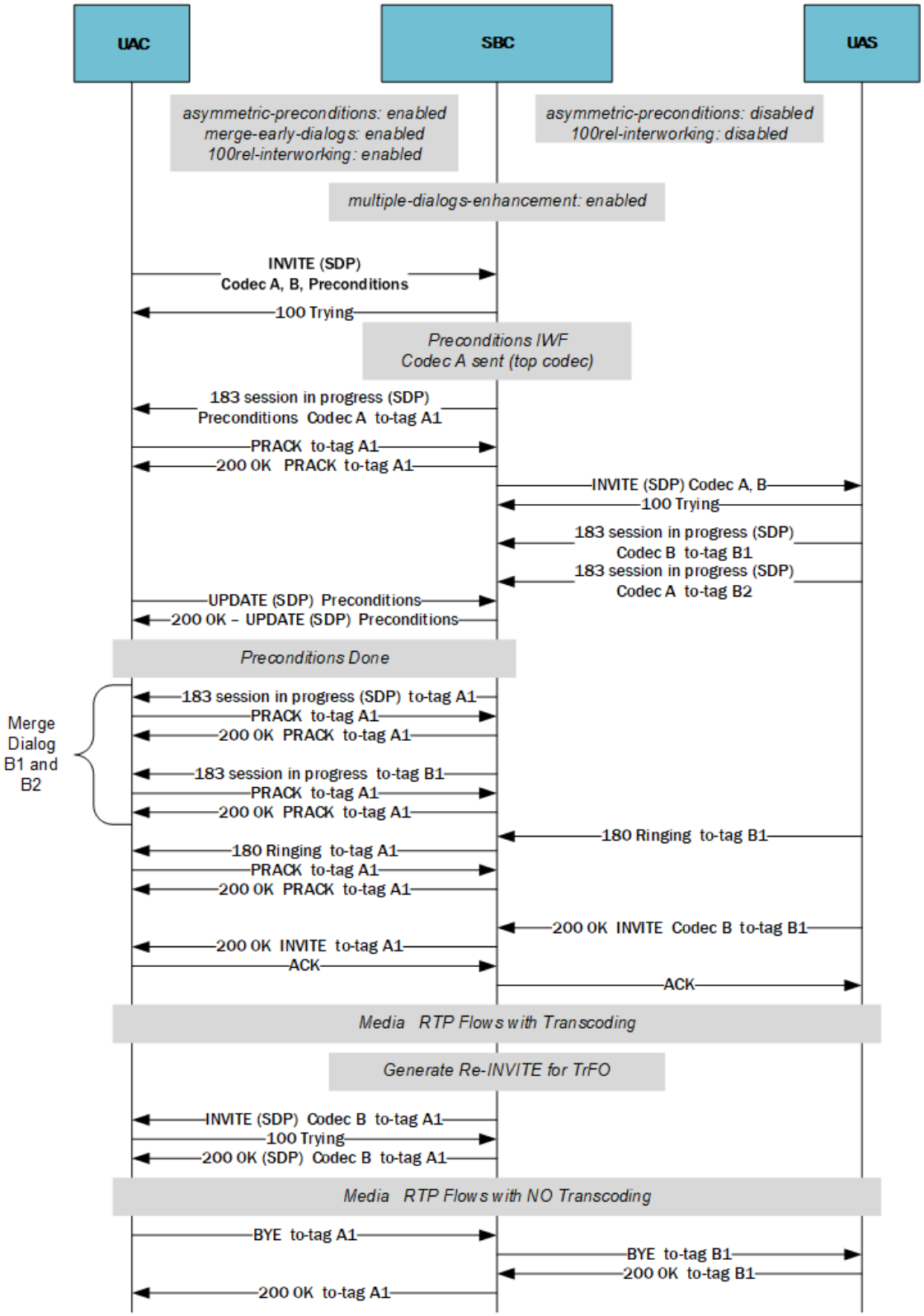
These call flows can also include a negotiated codec in the final answer (200 OK) that is different from that of the calling side. If so, the SBC triggers transcoding upon call establishment, consuming the associated resources. To avoid this, the SBC can use its TrFO function to determine whether the callee's codec is available at the caller based on the stored responses. If so, the SBC can send a re-INVITE to the caller with the negotiated codec on the called side.

Scenarios wherein the SBC can identify a codec that allows it to avoid transcoding include:

- It has received the first 18x with SDP, but the final response from the same callee has a different SDP.
- It has received the first 18x with SDP, but the final response, from a different, forked callee has a different codec.
- It has received the first 18x with no SDP, but the negotiated codec by the callee is in the list of initial offer of ingress INVITE.

In these cases, the codecs negotiated by the SBC and the callee are available. Once the preconditions are met and the SBC has identified a codec available, it sends the re-INVITE to the caller with the callee's negotiated codec. The called side simply agrees and the call proceeds without transcoding.

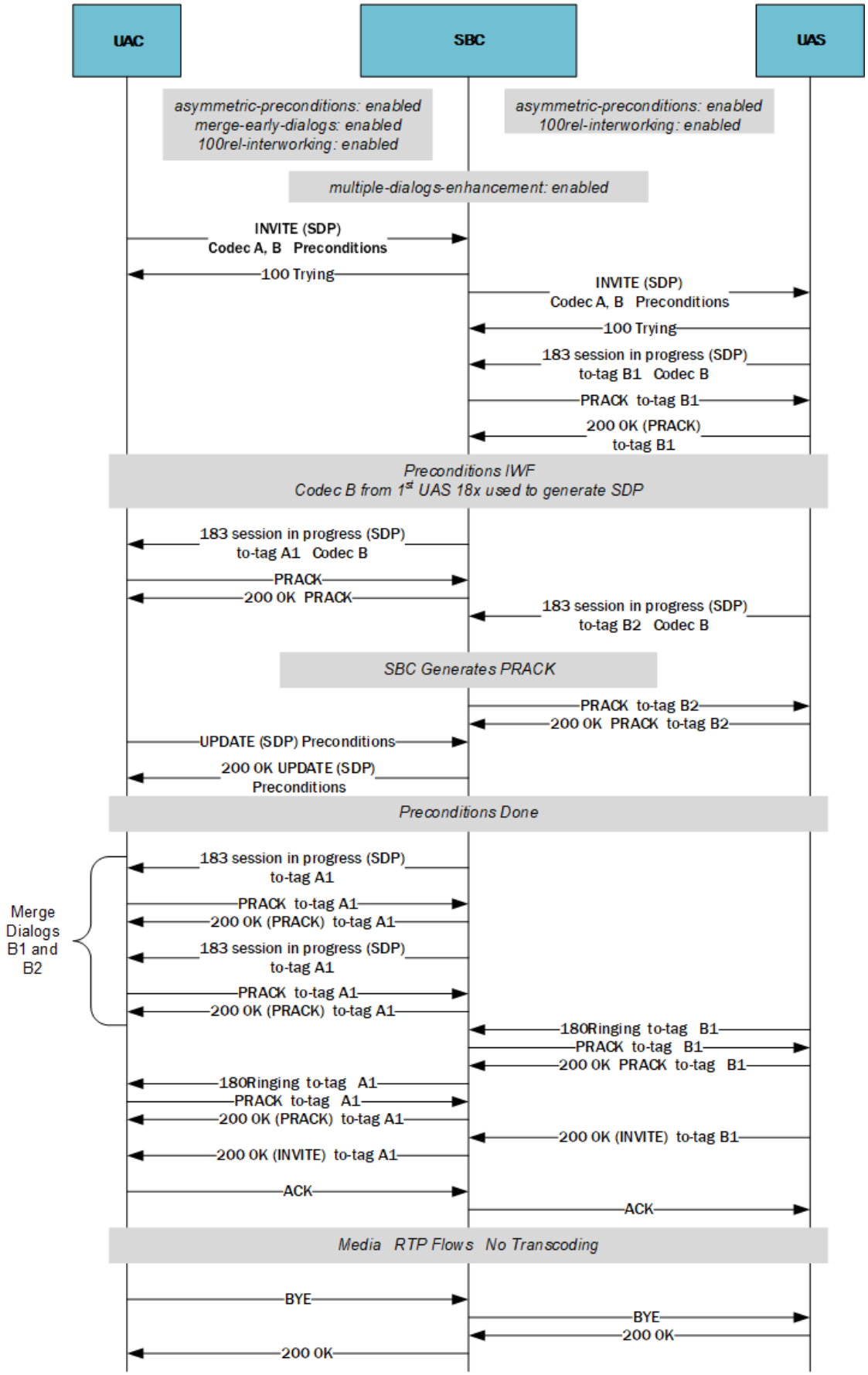
The example call flows below contrast the system operating with MED and needing to consider TrFO support within dynamic preconditions and static preconditions scenarios. For all flows, you have enabled MED facing the UAC.



This next flow depicts the system setting up asynchronous preconditions because, although you have enabled both realms for asymmetric preconditions, none of the dialogs returned from the UAS include preconditions. In addition, you have configured the UAC realm for preconditions to send without delay mode.

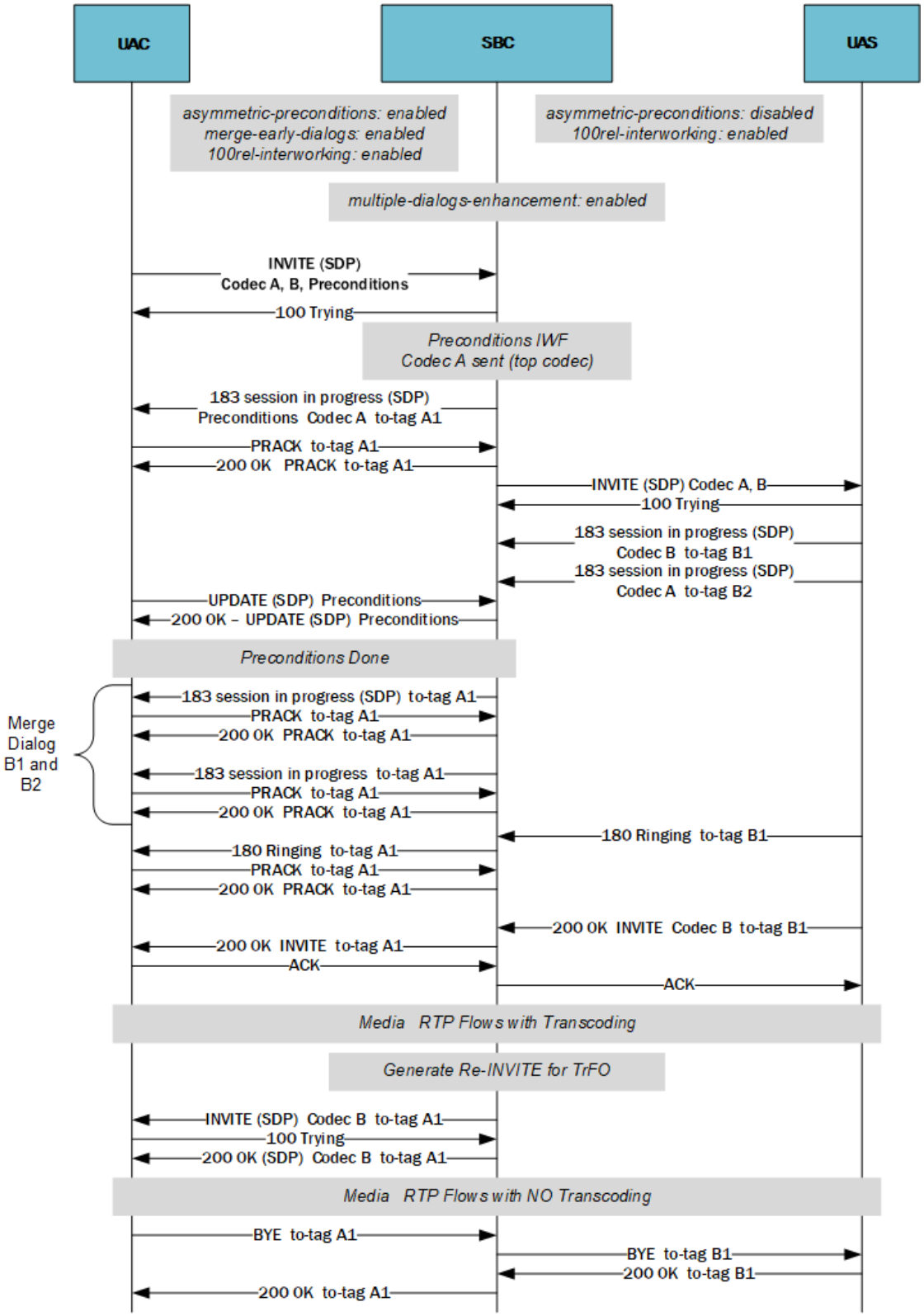
> **Note:**
>
> The SBC supports TrFO in both send with, and send without delay mode.

The system determines that the UAC has specified Codec A and the UAS has specified Codec B after the preconditions and merge dialog functions are complete. This is because the B1 session is the first dialog to present a 200 OK to the INVITE. The SBC determines that the UAC can support Codec B, so it performs its TrFO function by issuing a re-INVITE to the UAC with Codec B.


UAC
SBC
UAS
asymmetric-preconditions: enabled
merge-early-dialogs: enabled
100rel-interworking: enabled
asymmetric-preconditions: enabled
100rel-interworking: enabled
multiple-dialogs-enhancement: enabled
INVITE (SDP)
Codec A, B Preconditions
100 Trying
INVITE (SDP)
Codec A, B Preconditions
100 Trying
183 session in progress (SDP)
to-tag B1 Codec B
PRACK to-tag B1
200 OK (PRACK)
to-tag B1
Preconditions IWF
Codec B from 1st UAS 18x used to generate SDP
183 session in progress (SDP)
to-tag A1 Codec B
PRACK
200 OK PRACK
183 session in progress (SDP)
to-tag B2 Codec B
SBC Generates PRACK
PRACK to-tag B2
200 OK PRACK to-tag B2
UPDATE (SDP) Preconditions
200 OK UPDATE (SDP)
Preconditions
Preconditions Done
Merge
Dialogs
B1 and
B2
183 session in progress (SDP)
to-tag A1
PRACK to-tag A1
200 OK (PRACK) to-tag A1
183 session in progress (SDP)
to-tag A1
PRACK to-tag A1
200 OK (PRACK) to-tag A1
180Ringing to-tag B1
PRACK to-tag B1
200 OK PRACK to-tag B1
180Ringing to-tag A1
PRACK to-tag A1
200 OK (PRACK) to-tag A1
200 OK (INVITE) to-tag B1
200 OK (INVITE) to-tag A1
ACK
ACK
Media RTP Flows No Transcoding
BYE
BYE
200 OK
200 OK

This last TrFO flow depicts the system setting up static, asymmetric preconditions because the UAS realm has asymmetric preconditions disabled. In addition, the system determines that both stations agree to Codec A after the preconditions and merge dialog functions are complete, so there is no need for TrFO.

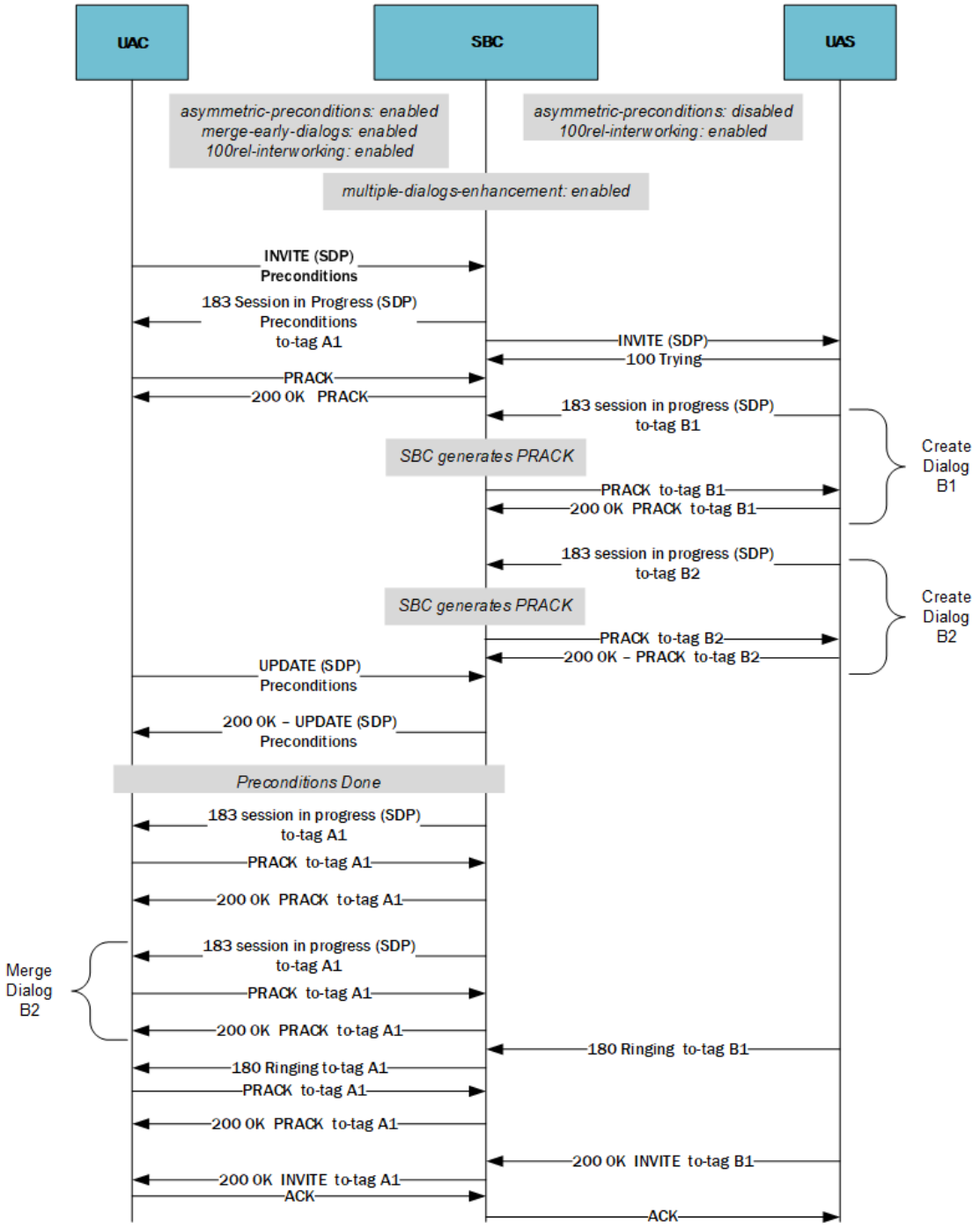
Note that, if the 200 OK came from dialog B2, the SBC would have triggered transcoding, and then TrFO.



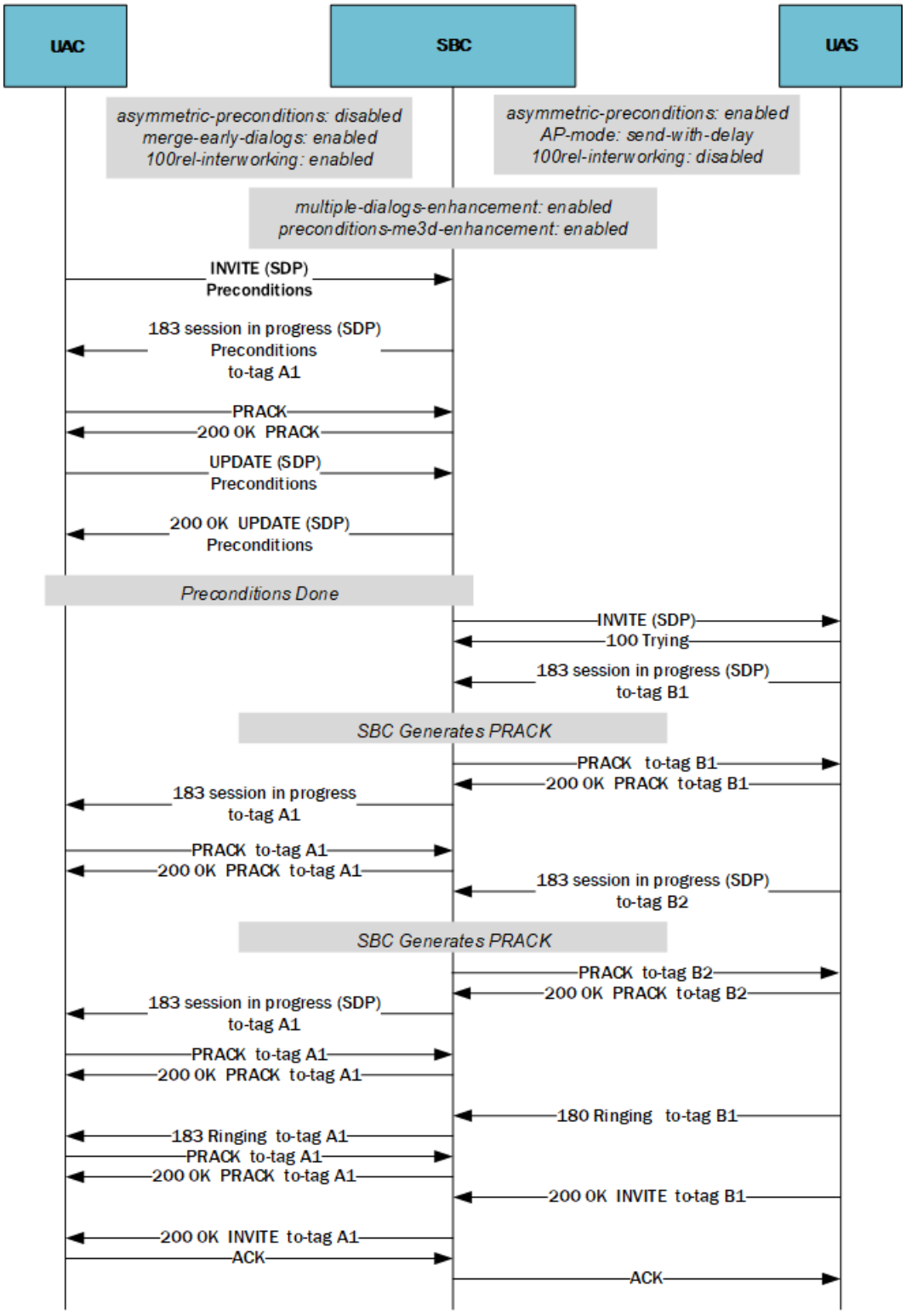
Preconditions and MED with and without Delay

This section presents the SBC supporting call flows with preconditions in conjunction with MED, including when configured to merge mode. In addition, you can set preconditions to operate under the 'send with delay' or 'send with no delay' mode.

The call flow below shows the SBC supporting asymmetric preconditions signaling sending the egress INVITE without delay in conjunction with multiple early dialogs. The system has invited the UAS without waiting for preconditions confirmation with the UAC (without delay). The system instantiates each client dialog, based on the 183s. In the meantime, the system confirms preconditions with the UAC and merges the multiple dialogs to a single dialog for the UAC. Ultimately, the system supports the call using dialog B1 from the UAS.



Similar to the above, the next call flow shows the system supporting preconditions with delay. Notice that the preconditions negotiation happens on the UAC side. In this case, the system needs to confirm preconditions for the dialog prior to proceeding with the call. The system merges the dialogs and ultimately supports the call using dialog B1 from the UAS.



Related Preconditions and MED Call Flows

This section presents call flows related to precondition support in conjunction with multiple early dialogs within sequential forking/recursion scenarios.

When you enable this feature, the SBC also provides precondition support, with or without simultaneous MED support, in redirected, recursive, or sequential forking calls. To achieve this

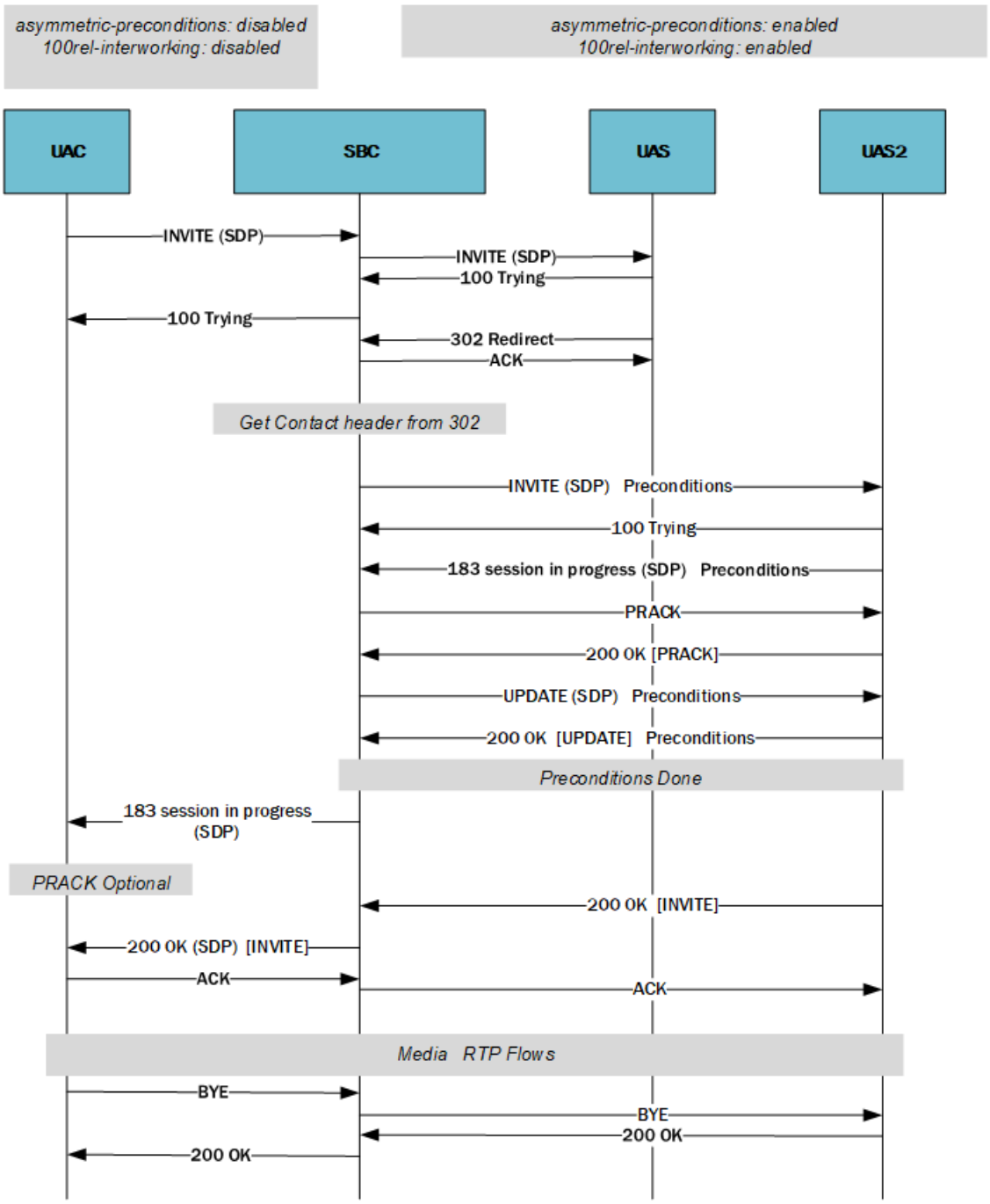
functionality, the SBC offers preconditions, in the context of both the supported header and SDP attributes in the egress INVITE used for the next destination. This assume proper **sip-interface** configuration. To accommodate this scenario, the SBC sends UPDATEs that provide confirmation for all serial forked users, and dialogs requesting confirmation within multiple early dialogs.

Message specific recursion and sequential forking considerations also include:

- If an endstation responds with a 6xx global failure, the SBC forwards it towards the UAC without changes. There is no serial or sequential forking applicable to 6xx error cases.
- If an endstation responds with a 491 request pending message in response to an initial INVITE, the SBC proxies the 491 error and does not retry that endstation after a timeout. If a new INVITE arrives, the SBC generates its own new INVITE towards its destinations, including that endstation, and continues to support recursion/sequential forking.

Static Preconditions Flows with 302 and 4xx Responses

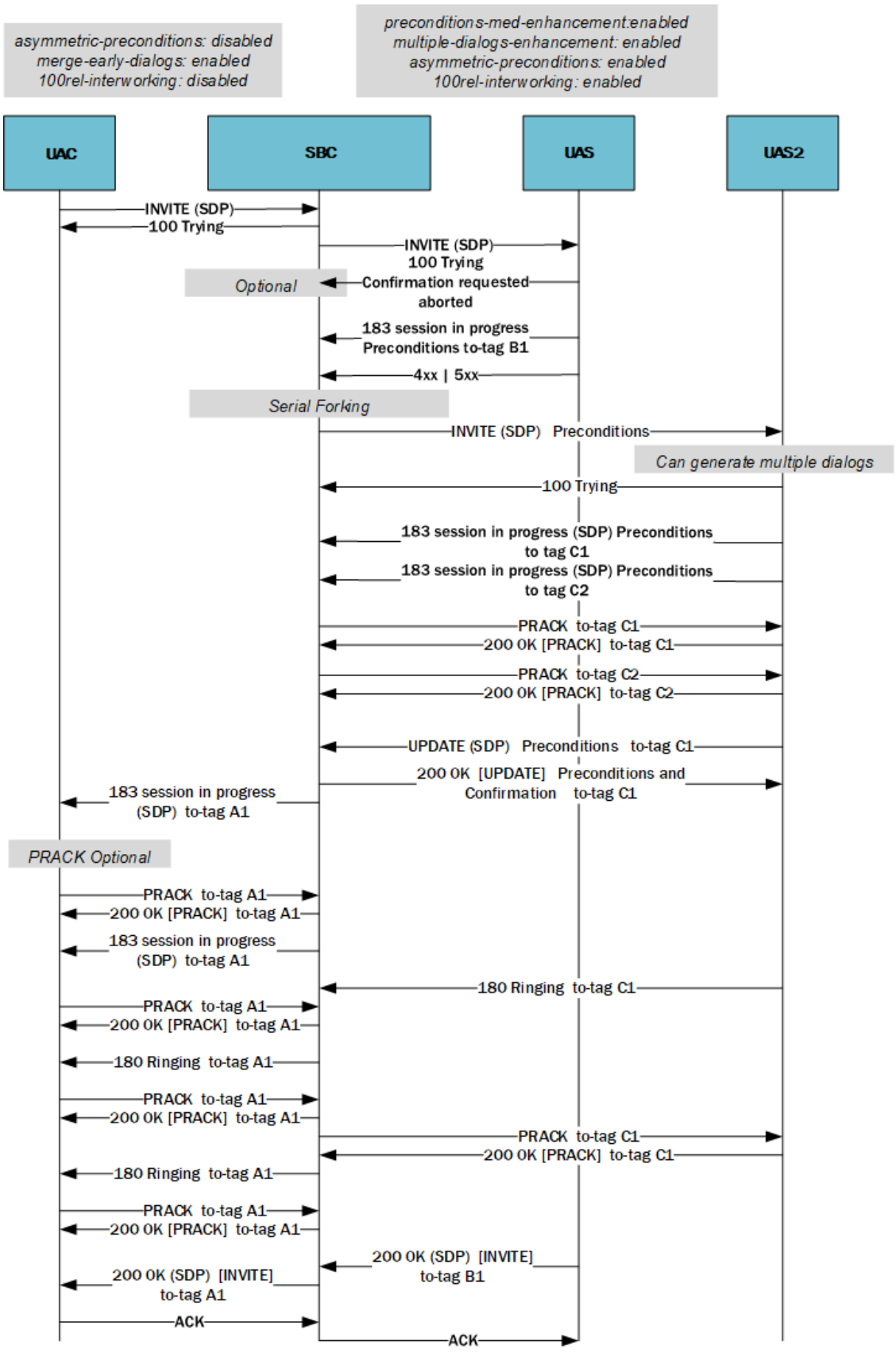
The call flow below presents the SBC fielding a redirect within a multiple early dialog scenario. The flow also includes static preconditions negotiation with the endstation to which the first UAS redirected the call. The call includes the preconditions UPDATE prior to successful call setup.



This next call flow shows the SBC supporting 4xx or 5xx responses, which generate multiple early dialogs with two alternative endstations, which the SBC merges to A1. In this case, an error message generates the contact to the second UAS. Preconditions are supported on the egress realm, and the SBC issues an INVITE with preconditions based on the 183 from the first UAS. UAS2 generates an UPDATE to confirm preconditions.

Ultimately, the first UAS sends a 200 OK to the initial INVITE without preconditions, so the SBC proceeds with the call to the first UAS.

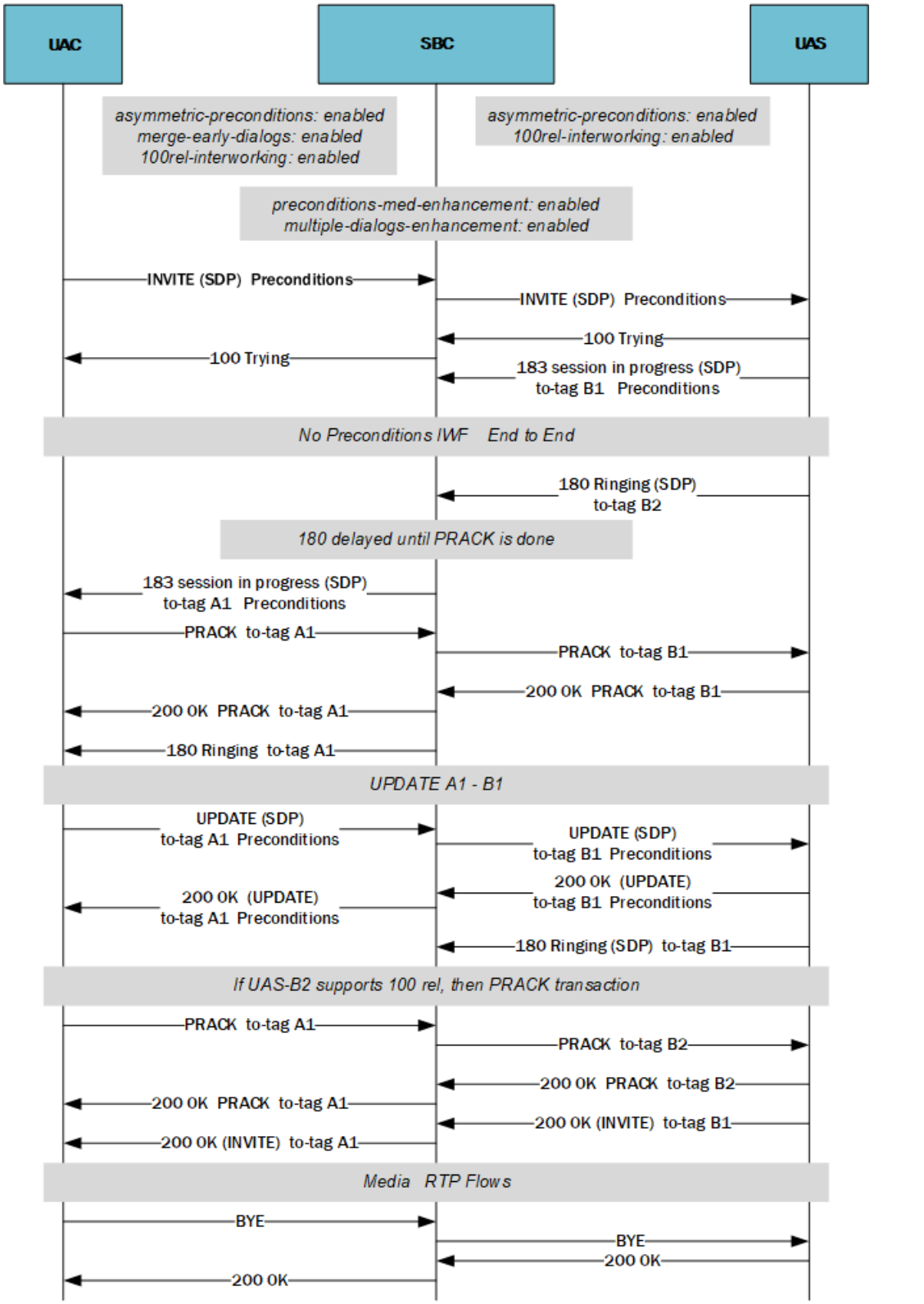
You must have enabled the **preconditions-med-enhancement** for the SBC to support preconditions with UAS2 despite the error. In addition, this parameter is also supporting MED merge in this call flow.


asymmetric-preconditions: disabled
merge-early-dialogs: enabled
100rel-interworking: disabled
preconditions-med-enhancement:enabled
multiple-dialogs-enhancement: enabled
asymmetric-preconditions: enabled
100rel-interworking: enabled
UAC
SBC
UAS
UAS2
INVITE (SDP)
100 Trying
INVITE (SDP)
100 Trying
Optional
Confirmation requested
aborted
183 session in progress
Preconditions to-tag B1
4xx | 5xx
Serial Forking
INVITE (SDP) Preconditions
Can generate multiple dialogs
100 Trying
183 session in progress (SDP) Preconditions
to tag C1
183 session in progress (SDP) Preconditions
to tag C2
PRACK to-tag C1
200 OK [PRACK] to-tag C1
PRACK to-tag C2
200 OK [PRACK] to-tag C2
UPDATE (SDP) Preconditions to-tag C1
200 OK [UPDATE] Preconditions and
Confirmation to-tag C1
183 session in progress
(SDP) to-tag A1
PRACK Optional
PRACK to-tag A1
200 OK [PRACK] to-tag A1
183 session in progress
(SDP) to-tag A1
180 Ringing to-tag C1
PRACK to-tag A1
200 OK [PRACK] to-tag A1
180 Ringing to-tag A1
PRACK to-tag A1
200 OK [PRACK] to-tag A1
PRACK to-tag C1
200 OK [PRACK] to-tag C1
180 Ringing to-tag A1
PRACK to-tag A1
200 OK [PRACK] to-tag A1
200 OK (SDP) [INVITE]
to-tag B1
200 OK (SDP) [INVITE]
to-tag A1
ACK
ACK

Dynamic Preconditions Flows with 302 and 4xx Responses

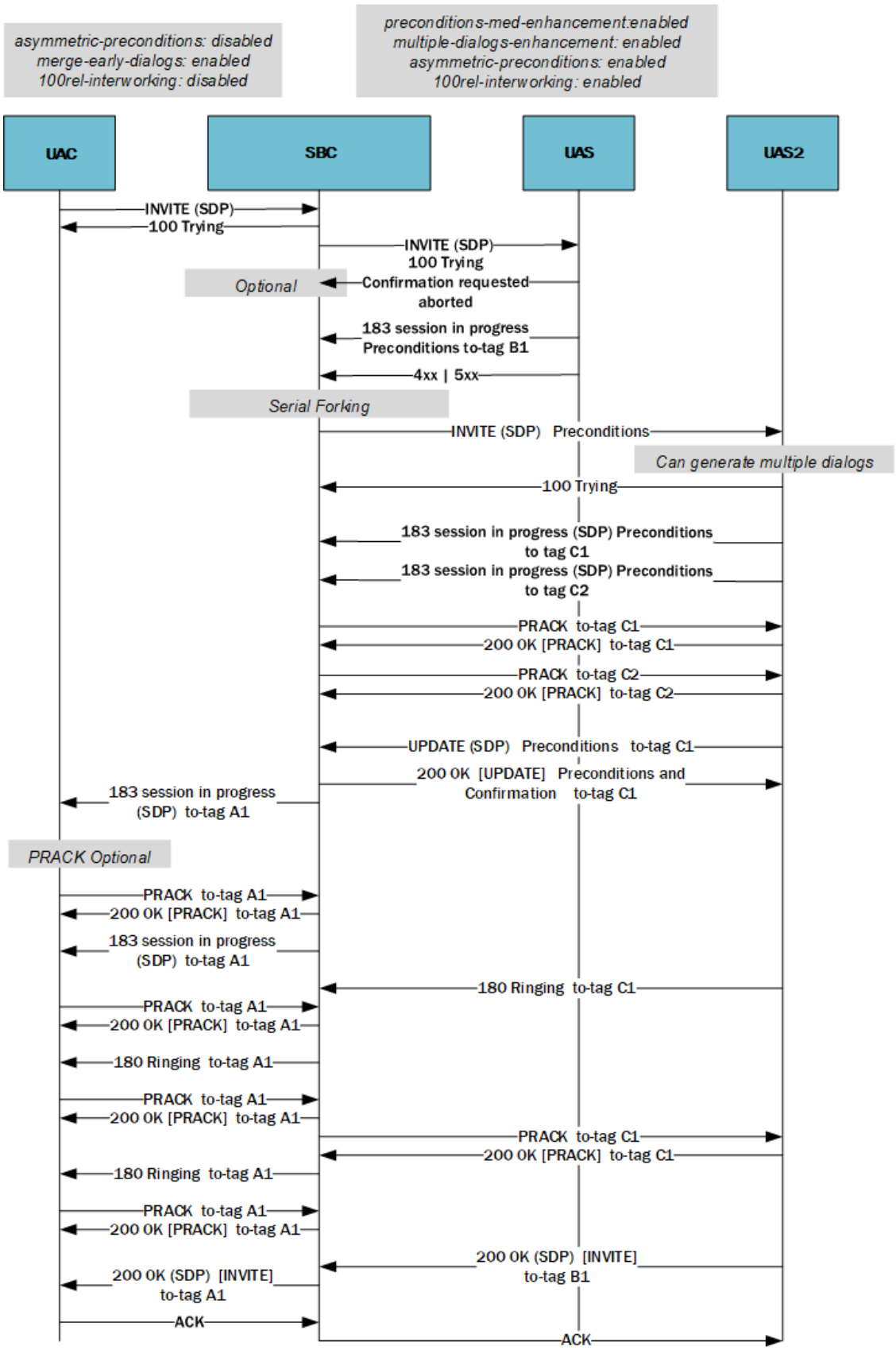
This next call flow shows the SBC supporting a 302 redirect within a dynamic preconditions scenario. (This generates multiple early dialogs with two alternative endstations.) The SBC offers preconditions to both the first and second UAS. The UPDATE exchange verifies the preconditions between the UAC and UAS2. Having completed preconditions negotiation, UAS2 answers the call, supported by the SBC.

You must have enabled the **preconditions-med-enhancement** for the SBC to support preconditions across the redirect.


UAC
SBC
UAS
asymmetric-preconditions: enabled
merge-early-dialogs: enabled
100rel-interworking: enabled
asymmetric-preconditions: enabled
100rel-interworking: enabled
preconditions-med-enhancement: enabled
multiple-dialogs-enhancement: enabled
INVITE (SDP) Preconditions
INVITE (SDP) Preconditions
100 Trying
100 Trying
183 session in progress (SDP)
to-tag B1 Preconditions
No Preconditions IWF End to End
180 Ringing (SDP)
to-tag B2
180 delayed until PRACK is done
183 session in progress (SDP)
to-tag A1 Preconditions
PRACK to-tag A1
PRACK to-tag B1
200 OK PRACK to-tag B1
200 OK PRACK to-tag A1
180 Ringing to-tag A1
UPDATE A1 - B1
UPDATE (SDP)
to-tag A1 Preconditions
UPDATE (SDP)
to-tag B1 Preconditions
200 OK (UPDATE)
to-tag B1 Preconditions
200 OK (UPDATE)
to-tag A1 Preconditions
180 Ringing (SDP) to-tag B1
If UAS-B2 supports 100 rel, then PRACK transaction
PRACK to-tag A1
PRACK to-tag B2
200 OK PRACK to-tag B2
200 OK PRACK to-tag A1
200 OK (INVITE) to-tag B1
200 OK (INVITE) to-tag A1
Media RTP Flows
BYE
BYE
200 OK
200 OK
asymmetric-preconditions: disabled
100rel-interworking: disabled
asymmetric-preconditions: enabled
100rel-interworking: enabled
UAC
SBC
UAS
UAS2
INVITE (SDP)
INVITE (SDP)
100 Trying
100 Trying

In this next flow, similar to the static preconditions example above, the SBC receives an error message from the first UAS, which triggers the recursion to UAS2. The SBC presents the requisite UPDATE to the new station as a step to confirm the preconditions, which UAS2 acknowledges. Ultimately, UAS2 sends a 200 OK to the INVITE and the call proceeds.

You must have enabled the **preconditions-med-enhancement** feature for the SBC to support preconditions with UAS2 despite the error. In addition, this feature supports MED merge in this call flow.


asymmetric-preconditions: disabled
merge-early-dialogs: enabled
100rel-interworking: disabled
preconditions-med-enhancement:enabled
multiple-dialogs-enhancement: enabled
asymmetric-preconditions: enabled
100rel-interworking: enabled
UAC
SBC
UAS
UAS2
INVITE (SDP)
100 Trying
INVITE (SDP)
100 Trying
Optional
Confirmation requested
aborted
183 session in progress
Preconditions to-tag B1
4xx | 5xx
Serial Forking
INVITE (SDP) Preconditions
Can generate multiple dialogs
100 Trying
183 session in progress (SDP) Preconditions
to tag C1
183 session in progress (SDP) Preconditions
to tag C2
PRACK to-tag C1
200 OK [PRACK] to-tag C1
PRACK to-tag C2
200 OK [PRACK] to-tag C2
UPDATE (SDP) Preconditions to-tag C1
200 OK [UPDATE] Preconditions and
Confirmation to-tag C1
183 session in progress
(SDP) to-tag A1
PRACK Optional
PRACK to-tag A1
200 OK [PRACK] to-tag A1
183 session in progress
(SDP) to-tag A1
180 Ringing to-tag C1
PRACK to-tag A1
200 OK [PRACK] to-tag A1
180 Ringing to-tag A1
PRACK to-tag A1
200 OK [PRACK] to-tag A1
PRACK to-tag C1
200 OK [PRACK] to-tag C1
180 Ringing to-tag A1
PRACK to-tag A1
200 OK [PRACK] to-tag A1
200 OK (SDP) [INVITE]
to-tag B1
200 OK (SDP) [INVITE]
to-tag A1
ACK
ACK

SIP Interface Configuration

To configure a SIP interface:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. **state**—Enable or disable the SIP interface. The default is **enabled**. The valid values are:
   - enabled | disabled

   > **Note:**
   >
   > Oracle does not recommend disabling and re-enabling a sip-interface operating with TCP ports. Depending on conditions and circumstances, you may not be able to re-enable this sip-interface without rebooting the system. If you need to disable, then re-enable a sip-interface, ensure that:
   >
   > – There are no ESTABLISHED in-bound sockets
   >
   > – The access-control-trust-level of the realm must not be configured to low or medium

5. **realm-id**—Enter the name of the realm to which the SIP interface is connected.
6. **sip-ports**—Access the **sip-ports** subelement.
7. **carriers**—Enter the list of carriers related to the SIP interface.

   Entries in this field can be from 1 to 24 characters in length and can consist of any alphabetical character (Aa-Zz), numerical character (0-9), or punctuation mark (! ”$ % ^ & * ( ) + - = < > ? ‘ | { } [ ] @ / \ ‘ ~ , . _ : ;) or any combination of alphabetical characters, numerical characters, or punctuation marks. For example, both 1-0288 and acme_carrier are valid carrier field formats

8. **proxy-mode**—Enter an option for the proxy mode parameter. Valid values are:
   - **Proxy**—Forward all SIP requests to selected targets.
   - **Redirect**—Send a SIP 3xx redirect response with the selected target(s) in the Contact header.
   - **Record-Route**—Forward requests to selected target(s) and insert a Record-Route header with the SBC’s address. For stateless and transaction mode only.

9. **redirect-action**—Enter the value for the redirect action. Valid values are:
   - **Proxy**—Send the SIP request back to the previous hop.
   - **Recurse**—Recurses on the Contacts in the response.

     The designated proxy action will apply to SIP 3xx responses received from non-session agents and to 3xx responses received from session agents without configured SIP Redirect message actions (for example, session agents without values for the redirect action field).
   - **Recurse-305-only**—Recurses on the Contacts only in a 305 response.
10. **contact-mode**—Set the Contact header routing mode, which determines how the contact address from a private network is formatted.

    For example, whether a maddr parameter equal to the Oracle Communications Session Border Controller's SIP proxy needs to be added to a URI present in a Contact header.

    The default is **none**. The valid values are:
    - **none**—The address portion of the header becomes the public address of that private realm.
    - **maddr**—The address portion of the header will be set to the IP address of the Oracle Communications Session Border Controller's B2BUA.
    - **strict**—The contents of the Request-URI is destroyed when a Record-Route header is present.
    - **loose**—The Record-Route header is included in a Request, which means the destination of the request is separated from the set of proxies that need to be visited along the way.
11. **nat-traversal**—Define the type of HNT enabled for SIP. The default is **none**. Valid values are:
    - **none**—HNT function is disabled for SIP.
    - **rport**—SIP HNT function only applies to endpoints that include the rport parameter in the Via header. HNT applies when the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address.
    - **always**—SIP HNT applies to requests when the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address. (Even when the rport parameter is not present.)
12. **nat-interval**—Set the expiration time in seconds for the Oracle Communications Session Border Controller's cached registration entry for an HNT endpoint. The default is **30**. The valid range is:
    - Minimum—0
    - Maximum—4294967295

      Oracle recommends setting the NAT interval to one-third of the NAT binding lifetime. A NAT binding lifetime is the network connection inactivity timeout. The value is configured (or hardwired) in the NAT device (firewall). This timer is used to cause the UA to send REGISTER messages frequently enough to retain the port binding in the NAT. Retaining the binding lets inbound requests to be sent through the NAT.
13. **tcp-nat-interval**—Set the registration cache expiration time in seconds to use for endpoints behind a NAT device that register using TCP. On upgrade, the Oracle Communications Session Border Controller assigns this parameter the same value as the existing NAT interval. The default is **90**. The valid range is:

- Minimum—0
- Maximum—999999999

The Oracle Communications Session Border Controller uses the value you set for the TCP NAT interval as the expiration value passed back in SIP REGISTER (200 OK) responses to endpoints behind a NAT that register over TCP. The NAT interval value with which you are familiar from previous releases is used for endpoints behind a NAT that register over UDP. Requiring endpoints that register over TCP to send refresh requests as frequently as those registering over UDP puts unnecessary load on the Oracle Communications Session Border Controller. By adding a separate configuration for the TCP NAT interval, the load is reduced.

For upgrade and backward compatibility with Oracle Communications Session Border Controller releases prior to Release 4.1, when the tcpNatInterval is not present in the XML for a SIP interface configuration, the value of the NAT interval (natInterval) is used for the TCP NAT interval as well.

14. **registration-caching**—Enable for use with all UAs, not just those that are behind NATs. The default is **disabled**. The valid values are:

- enabled | disabled

If enabled, the Oracle Communications Session Border Controller caches the Contact header in the UA's REGISTER request when it is addressed to one of the following:

- Oracle Communications Session Border Controller
- registrar domain value
- registrar host value

The Oracle Communications Session Border Controller then generates a Contact header with the Oracle Communications Session Border Controller's address as the host part of the URI and sends the REGISTER to the destination defined by the registrar host value.

Whether or not SIP HNT functionality is enabled affects the value of the user part of the URI sent in the Contact header:

- HNT enabled: the Oracle Communications Session Border Controller takes the user part of the URI in the From header of the request and appends a cookie to make the user unique. A cookie is information that the server stores on the client side of a client-server communication so that the information can be used in the future.
- HNT disabled: the user part of the Contact header is taken from the URI in the From header and no cookie is appended. This is the default behavior of the Oracle Communications Session Border Controller.

When the registrar receives a request that matches the address-of-record (the To header in the REGISTER message), it sends the matching request to the Oracle Communications Session Border Controller, which is the Contact address. Then, the Oracle Communications Session Border Controller forwards the request to the Contact-URI it cached from the original REGISTER message.

15. **min-reg-expire**—Set the time in seconds for the SIP interface. The value you enter here sets the minimum registration expiration time in seconds for HNT registration caching. The default is **300**. The valid range is:

- Minimum—0
- Maximum—999999999

This value defines the minimum expiration value the Oracle Communications Session Border Controller places in each REGISTER message it sends to the real registrar. In

HNT, the Oracle Communications Session Border Controller caches the registration after receiving a response from the real registrar and sets the expiration time to the NAT interval value.

Some UAs might change the registration expiration value they use in subsequent requests to the value specified in this field. This change causes the Oracle Communications Session Border Controller to send frequent registrations on to the real registrar.

16. **registration-interval**—Set the Oracle Communications Session Border Controller's cached registration entry interval for a non-HNT endpoint. Enter the expiration time in seconds that you want the Oracle Communications Session Border Controller to use in the REGISTER response message sent back to the UA. The UA then refreshes its registration by sending another REGISTER message before that time expires. The default is **3600**. The valid range is:

    - Minimum—0

      A registration interval of zero causes the Oracle Communications Session Border Controller to pass back the expiration time set by and returned in the registration response from the registrar.

    - Maximum—999999999

      If the expiration time you set is less than the expiration time set by and returned from the real registrar, the Oracle Communications Session Border Controller responds to the refresh request directly rather than forwarding it to the registrar.

      Although the registration interval applies to non-HNT registration cache entries, and the loosely related NAT interval applies to HNT registration cache entries, you can use the two in combination. Using a combination of the two means you can implement HNT and non-HNT architectures on the same Oracle Communications Session Border Controller. You can then define a longer interval time in the registration interval field to reduce the network traffic and load caused by excess REGISTER messages because there is no NAT binding to maintain.

17. **route-to-registrar**—Enable routing to the registrar to send all requests that match a cached registration to the destination defined for the registrar host; used when the Request-URI matches the registrar host value or the registrar domain value, not the Oracle Communications Session Border Controller's address. Because the registrar host is the real registrar, it should send the requests back to the Oracle Communications Session Border Controller with the Oracle Communications Session Border Controller's address in the Request-URI. The default is **disabled**. The valid values are:

    - enabled | disabled

      For example, you should enable routing to the registrar if your network uses a N Oracle Communications Session Border Controller and needs requests to go through its service proxy, which is defined in the registrar host field.

18. **teluri-scheme**—Enable to convert SIP URIs to tel (resources identified by telephone numbers) URIs.

    If enabled, the requests generated on this SIP interface by the Oracle Communications Session Border Controller will have a tel URI scheme instead of the SIP URI scheme. Only the Request, From, and To URIs are changed to the tel scheme. After the dialog is established, the URIs are not changed. The default is **disabled**. The valid values are:

    - enabled | disabled

19. **uri-fqdn-domain**—Change the host part of the URIs to the FQDN value set here. If set to enabled, and used with an FQDN domain/host, the requests generated by the Oracle Communications Session Border Controller on this SIP interface will have the host part of

the URI set to this FQDN value. Only the Request, To, and From URIs are changed. After the dialog is established, the URIs are not changed.

20. **trust-mode**—Set the trust mode for the SIP interface, which is checked by the Oracle Communications Session Border Controller when it receives a message to determine whether the message source is trusted. The default is **all**. Available options are:

    - **all**—Trust all SIP elements (sources and destinations) in the realm(s), except untrusted session agents. Untrusted session agents are those that have the **trust-me** parameter set to **disabled**.
    - **agents-only**—Trust only trusted session agents. Trusted session agents are those that have the **trust-me** parameter set to **enabled**.
    - **realm-prefix**—Trust only trusted session agents, and source and destination IP addresses that match the IP interface's realm (or subrealm) address prefix. Only realms with non-zero address prefixes are considered.
    - **registered**—Trust only trusted session agents and registered endpoints. Registered endpoints are those with an entry in the Oracle Communications Session Border Controller's registration cache.
    - **none**—Trust nothing.

      Session agents must have one or more of the following:

    - global realm
    - same realm as the SIP interface
    - realm that is a subrealm of the SIP interface's realm

21. **trans-expire**—Set the TTL expiration timer in seconds for SIP transactions. This timer controls the following timers specified in RFC 3261:

    - Timer B—SIP INVITE transaction timeout
    - Timer F—non-INVITE transaction timeout
    - Timer H—Wait time for ACK receipt
    - Timer TEE—Used to transmit final responses before receiving an ACK

      The default is **0**. If you leave this parameter set to the default, then the Oracle Communications Session Border Controller uses the timer value from the global SIP configuration. The valid range is:

    - Minimum—0
    - Maximum—999999999

22. **invite-expire**—Set the TTL expiration timer in seconds for a SIP client/server transaction after receiving a provisional response.

    You set this timer for the client and the sever by configuring it on the SIP interface corresponding to the core or access side.

    The default is **0**. If you leave this parameter set to the default, then the Oracle Communications Session Border Controller uses the timer value from the global SIP configuration. The valid range is:

    - Minimum—0
    - Maximum—999999999

23. **max-redirect-contacts**—Set the maximum number of contacts or routes for the Oracle Communications Session Border Controller to attempt in when it receives a SIP Redirect (3xx Response). The default is **0**. If you leave this parameter set to the default, then the

Oracle Communications Session Border Controllerwill exercise no restrictions on the number of contacts or routes. The valid range is:

- Minimum—0
- Maximum—10

24. **response-map**—Enter the name of the SIP response map configuration that you want to apply to this SIP interfaces for outgoing responses. This parameter is blank by default.
25. **local-response-map**—Enter the name of the SIP response map configuration that you want to apply to this SIP interfaces for locally-generated SIP responses. This parameter is blank by default.
26. **options**—Optional.

Configuring SIP Ports

To configure SIP ports:

1. From sip-interface, type **sip-ports** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(sip-interface)# sip-ports
ORACLE(sip-port)#
```

2. **address**—Enter the IP address of the host associated with the sip-port entry on which to listen. For example:

```
192.168.11.101
```

3. **port**—Enter the port number you want to use for this sip-port. The default is **5060**. The valid range is:
   - Minimum—1
   - Maximum—65535
4. **transport-protocol**—Indicate the transport protocol you want to associate with the SIP port. The default is **UDP**. The valid values are:
   - **TCP**—Provides a reliable stream delivery and virtual connection service to applications through the use of sequenced acknowledgment with the retransmission of packets when necessary.
   - **UDP**—Provides a simple message service for transaction-oriented services. Each UDP header carries both a source port identifier and destination port identifier, allowing high-level protocols to target specific applications and services among hosts.
   - **TLS**—See the Security chapter for more information about configuring TLS.
5. **allow-anonymous**—Define the allow anonymous criteria for accepting and processing a SIP request from another SIP element.

   The anonymous connection mode criteria includes admission control based on whether an endpoint has successfully registered. Requests from an existing SIP dialog are always accepted and processed. The default is **all**.

   The following table lists the available options.

   - **all**—All requests from any SIP element are allowed.
   - **agents-only**—Only requests from configured session agents are allowed. The session agent must fit one of the following criteria:

  - Have a global realm.
  - Have the same realm as the SIP interface
  - Be a sub-realm of the SIP interface's realm.
    When an agent that is not configured on the system sends an INVITE to a SIP interface, the Oracle Communications Session Border Controller:
  - Refuses the connection in the case of TCP.
  - Responds with a 403 Forbidden in the case of UDP.

- **realm-prefix**—The source IP address of the request must fall within the realm's address prefix or a SIP interface sub-realm. A sub-realm is a realm that falls within a realm-group tree. The sub-realm is a child (or grandchild, and so on) of the SIP interface realm.

  Only realms with non-zero address prefixes are considered. Requests from session agents (as described in the **agents-only** option) are also allowed.

- **registered**—Only requests from user agents that have an entry in the registration cache (regular or HNT) are allowed; with the exception of a REGISTER request. A REGISTER request is allowed from any user agent.

  The registration cache entry is only added if the REGISTER is successful. Requests from configured session agents (as described in the **agents-only** option) are also allowed.

- **register-prefix**—Only requests from user agents that have an entry in the Registration Cache (regular or HNT) are allowed; with the exception of a REGISTER request. A REGISTER request is allowed only when the source IP address of the request falls within the realm address-prefix or a SIP interface sub-realm. Only realms with non-zero address prefixes are considered.

  The Registration Cache entry is only added if the REGISTER is successful. Requests from configured session agents (as described in the **agents-only** option) are also allowed.

Asymmetric Preconditions Configuration

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

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

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **asymmetric-preconditions** — identifies whether to enable preconditions interworking on the interface. Allowable values are **enabled** and **disabled**. The default is **disabled**. You cannot enable asymmetric preconditions unless you have first set the value of **sip-interface**, **options** to **100rel-interworking**.

4. **asymmetric-preconditions-mode** — identifies, when the value of **asymmetric-preconditions** is **enabled**, whether to send egress INVITEs immediately or to delay them until preconditions have been met. Allowable values are **send-with-delay** and **send-with-nodelay**. The value **send-with-delay** delays INVITEs on the egress interface until preconditions are met on the ingress interface. The value **send-with-nodelay** forwards INVITEs to the egress interface immediately, but holds the responses until preconditions are met on the ingress interface. The default is **send-with-nodelay**.
5. **add-sdp-in-msg** — identifies the messages in which to insert SDP offers or answers. The only allowable value is **18xresp** which, for an offerless INVITE that needs preconditions, causes the SBC to insert the SDP, as configured in the media profile names listed in **add-sdp-profiles-in-msg**, in the 18x (183) response towards the UE. The default is null (no value).
6. **add-sdp-profiles-in-msg** — identifies a list of media profiles that contain, based on the codec, the SDP to insert in the 18x response when **add-sdp-in-msg** is configured. The default is null (no value).
7. Type **done** to save your configuration.

Diversion Info and History-Info Header Mapping

History-Info and Diversion are two headers commonly used in SIP signaling to convey information related to call transfer and call diversion. The Oracle Communications Session Border Controller (SBC) supports mapping and interworking between networks that support the History-Info versus the Diversion header. You implement this interworking on the SBC using the **diversion-info-mapping-mode** parameter on egress **sip-interface** configuration elements. This interworking, and the subsequent header behaviors, comply with RFC 7544. When configured, the SBC monitors signaling for the presence of History-info headers that comply with RFC 7044 and Diversion headers that comply with RFC 5806 to trigger the interworking. The SBC also provides support for RFC 8119, with respect to the cause URI parameter.

The History-Info header is the standard solution adopted by the Internet Engineering Task Force (IETF) for storing re-targeting information. The non-standard Diversion header is also used in many existing network implementations. Individual networks typically use one or the other. As both headers address call forwarding needs but have different syntaxes, having both present in a signaling request can cause diverting information to be misinterpreted, thereby making an interworking solution necessary. In addition to using different syntaxes, these methods also use different reason codes for the diversions, different security flags, and list events using opposite chronology. The diversion header lists the last diversion first; the History-Info header lists the first diversion first.

The SBC applies the configuration at the egress interface. You can configure the following header interworking modes:

- Diversion to History-Info
- History-info to Diversion
- A combination of Diversion to History-Info interworking and a forced insertion of the History-info header if the INVITE contains neither

When interworking from a History-Info header to a Diversion header, the SBC initializes the new Diversion header with the value of the History-Info header.

History-Info and Diversion Header Interworking Operations

The Oracle Communications Session Border Controller (SBC) performs this interworking on the information provided in the initial INVITE. History-Info and Diversion header entries may arrive in any order and can be mixed. The SBC supports interworking for interleaved entries and mixed input. History-Info has a much broader scope than Diversion header (not limited to call forwarding) so not all information can be interworked. The SBC manages History-Info headers with or without cause parameter per RFC 7544.

At a high level, the SBC performs the following tasks:

- When interworking History-Info to Diversion headers:
  - Supports the mp parameters in History-Info received and generated to the hi-targeted-to-URI to change the user. Examples include forwarding to a different call center technician.
  - If received in the INVITE, the SBC stores the rc and np parameters:
    * rc—Changes Request-URI, while the target user does not. Examples include forwarding to a user's voice mail.
    * np—No change in the Request-URI. Examples include a proxy forwarding a request to a next-hop proxy when you are using loose routing.
  - Interworks the History-Info 380 cause parameter to the Diversion header (cause=unknown), as described below.
- When interworking Diversion headers to History-Info:
  - Creates placeholder History-Info headers when the Diversion counter is greater than 1.
  - Converts TEL-URIs to SIP-URIs.
  - Forwards History-Info not related to call forwarding without change.

For both directions:

- Preserves unknown elements of History-Info and Diversion headers, such as display name, parameters, and enclosed headers adhering to the HI or DIV syntax. Specifically:
  - If there are any unknown parameters/elements present in the URI part (in between "<" and ">") of HI/Diversion headers, the SBC preserves those values in the converted entries.
  - If there is any extra/unknown supplementary information outside the URI of an HI entry, the SBC maps that HI entry, including the EXTRA info.
  - If there is any extra/unknown supplementary information outside the URI of the Diversion entry, the SBC maps that Diversion entry, but ignores the EXTRA information.
- Based on **sip-config** configuration, makes the Diversion and History-Info headers anonymous if the egress peer is untrusted.

> **Note:**
>
> If there is any information outside of the HI/DIV header syntax, the SBC treats that information as unknown or supplementary information.

You configure the SBC for this interworking support at the egress interface. The applicable parameter, **diversion-info-mapping-mode**, assumes traffic egressing the interface supports either History-Info or Diversion. There are three modes of operation:

- History-Info to Diversion Header Interworking (**hist2div**)
  - If Diversion headers are already present, the SBC considers them when building the egress request.
  - The SBC adds the unknown elements of History-Info headers to the generated Diversion headers (display-name, SIP parameters, SIP headers). See the specific functions presented above.
  - The SBC optionally converts History-Info with a "380" cause to a Diversion header (cause=unknown). By default, it forwards the History-Info cause=380 transparently.
  - Existing Diversion headers received in the input appear at the end of the Diversion list.
  - Converted History-Info entry with cause = 380, shall be at the top of newly created div headers. The SBC adds other HI entries with different cause (other than 380) per the conversion.
  - The SBC retains HI entries if they don't have a cause parameter, and if that entry is not a target entry to any diverting entry.
  - When configured, the SBC inserts the cause 380 History-Info as the topmost Diversion header, because it assumes the applicable events happened before the call forwarding History-Info.
    - When converting hist2div, the SBC removes the last History-Info with a cause parameter if used to build a diversion header.
    - If this cause parameter is 380 and option hist380todiv (hist-to-div-for-cause-380) is not present, then this History-Info header is not taken into account to build a Diversion because it is not recognized as a call forward. This results in the SBC keeping the last History-Info header.
    - If this cause parameter is 380 and option hist380todiv (hist-to-div-for-cause-380) is present, then this History-Info header is taken into account to build a Diversion because it is recognized as a call forward. In that case, the History-Info header removes the last History-Info header.
- Diversion Header to History-Info Interworking (**div2hist**)
  - The SBC generates the mp-param parameter for History-Info headers.
  - If the SBC is adding converted History-Info headers to existing History-Info headers it manages both index and mp-param continuation.
  - The SBC integrates any existing History-Info headers to the top of the list in the resulting request.
  - The SBC adds the unknown R-URI elements of Diversion headers to the generated History-Info headers (display-name, SIP parameters, SIP headers). If there is any other extra info that cannot be mapped to History-Info header, the SBC ignores it.
  - If a Diversion header contains a TEL-URI, the SBC converts it into a SIP-URI and inserts it into the resulting History-Info header.
  - Tel-URI with additional parameters to SIP-URI. If a Diversion header contains a TEL-URI, the SBC converts it into a SIP-URI and inserts it into the resulting History-Info header along with any additional Tel-URI parameters.

  - The SBC adds newly converted History-Info headers to the existing History-Info headers. If the last old/existing HI entry has cause 380, the SBC removes the mp-param from the first HI entry in the list of converted HI entries that it is going to add.
  - All Diversion headers are always converted to HI headers because information in diversion headers can always be converted.
- Forced Mode (**force**)
  - This mode is similar to **div2hist**, but invokes additional actions by the SBC.
  - If History-Info headers are already present, the SBC adds new History-Info header(s) to the existing set.
  - The SBC retargeting feature adds a History-Info entry as well as the expected Index and mp to an outgoing INVITE when certain conditions are met:
    * When **force** mode is set on the realm from which the 3xx or ENUM response came.
    * When the SBC gets a redirect response (3xx) or an ENUM response.
    * When this response also contains the new URI and cause-param.
    * When it receives a cause parameter in a SIP redirect reply.
  - The SBC extracts parameters for creating a History-Info header from the Contact header of the incoming SIP redirect reply. This Contact header must contain the URI and cause. The SBC copies all this information for use in the History-Info it generates.
  - Adds History-Info headers to a SIP INVITE if it has been re-targeted.
  - For repeated re-targeting the SBC keeps adding History-Info headers to each INVITE it sends out.
  - The SBC extracts information from a received Contact header from incoming re-direct replies. For this feature SBC only uses the URI and cause.
  - The SBC ignores a hi-target-param if it is received as the Contact of a 3xx or via an ENUM response. The SBC can not rely on that information.

The SBC retains any added History-Info entry it adds to an outgoing retargeting request for ensuing retargeting requests.

History-Info to Diversion Header Interworking

When configured for this interworking, the SBC performs the following steps for all History-Info headers except the last in the incoming initial INVITE:

1. Create a new diversion header.
2. If there is a subsequent HI header, extract the cause parameter, map it to the corresponding reason parameter, and add this reason parameter to the interworked Diversion header, using the following conversion table.

History-Info Cause	Diversion Reason
404	Unknown (default value)
302	unconditional
486	user-busy
408	no-answer
480	deflection

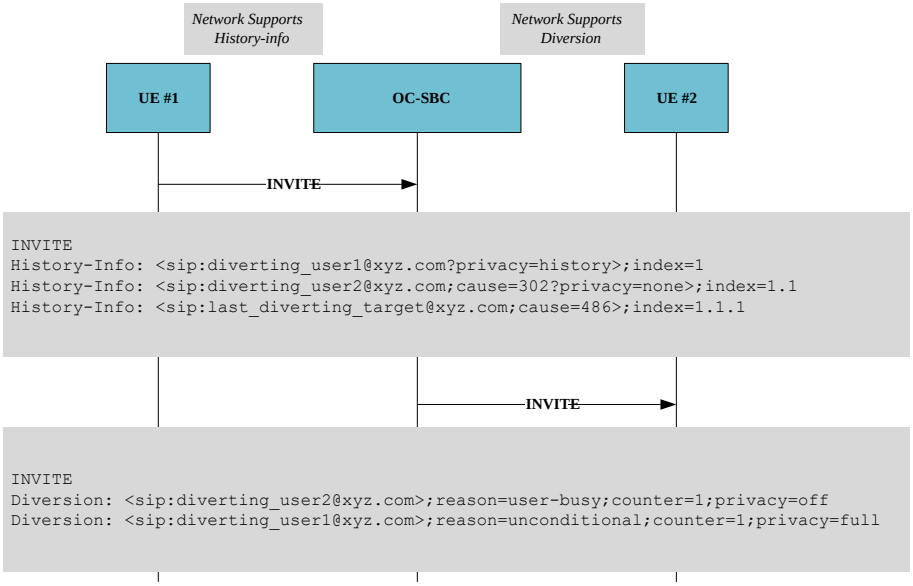
History-Info Cause	Diversion Reason
487	deflection
503	unavailable

3. If an HI entry includes a cause that is not in the list of valid causes, per RFC 7544, use the default value **Unknown** for the reason parameter.
4. If the History-Info header contains a privacy parameter, add a Privacy parameter to the Diversion header, using the following conversion table.

History-Info Privacy	Diversion Privacy
history	full
Field absent or none	Off (default value)

5. If there is no privacy parameter, use the default value **Off** in the Diversion header.
6. Add a counter parameter with the value **1**.
7. Delete the History-Info header.

The diagram below provides an example flow and the header text used for this example.



Diversion to History-Info Header Interworking

When configured for this interworking, the SBC takes the following steps for all Diversion headers in the incoming initial INVITE:

1. Create a new History-Info header.
2. Add the **index** header parameter.
3. Set the value for the first header to **1**. Append **.1** to the value of the last_index_value for the subsequent headers.
4. If the Diversion header contains a privacy parameter, add the Privacy header parameter to the History-Info header using the following conversion table.

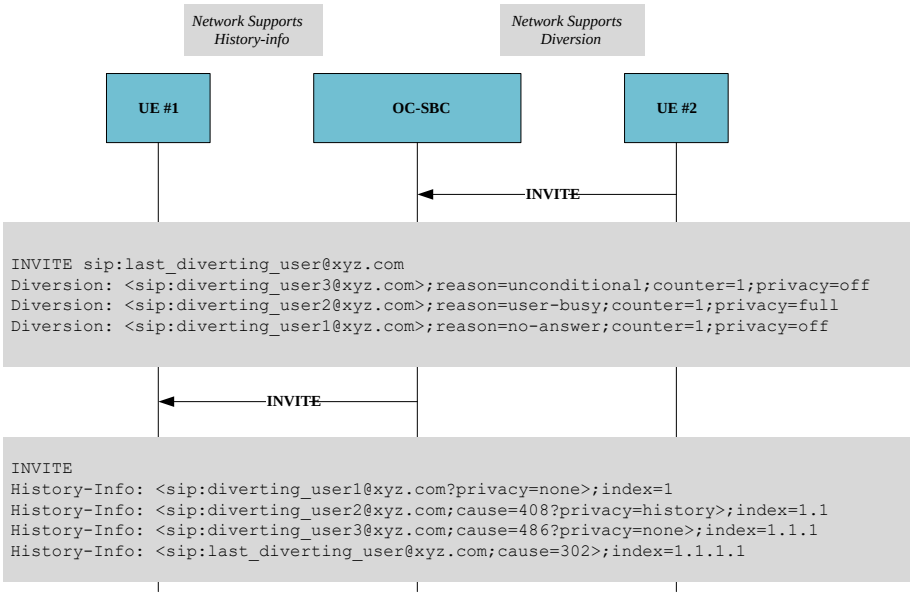
Diversion Privacy	History-Info Privacy
full	history
name	history
uri	history
off	none

5. If the previous Diversion header contains a reason parameter, add the **cause** header parameter to the History-Info header using the following conversion table. The SBC does not add a cause parameter to the first History-info entry.

Diversion Reason	History-Info Cause
unknown	404 (default value)
unconditional	302
user-busy	486
no-answer	408
deflection	480
unavailable	404
time-of-day	404
do-not-disturb	404
follow-me	404
out-of-service	404
away	404

6. If the presented reason parameter is not in the table above, set the parameter to the default value of **404**.
7. Remove the diversion header from the outgoing INVITE.

The diagram below provides an example flow and the header text used for this example.



Mapping the History-Info Cause Parameter

You can configure SBC to map the History-Info's cause 380 parameter to the Diversion header's reason=unknown parameter. You can configure the applicable parameter, **hist-to-div-**

for-cause-380, in both the **sip-config** and **sip-interface**, with the egress **sip-interface** setting taking precedence. This parameter only applies when you set the applicable **sip-interface** interworking mode to **hist2div**. The verify-config command informs you if there is a **hist-to-div-for-cause-380** without a corresponding **hist2div**.

By default, the **hist-to-div-for-cause-380** parameter is disabled, which causes the SBC to transparently forward the Hisory-Info entry with cause 380. Enabling the parameter makes the SBC interwork the parameter, allowing it to be understood by devices that support Diversion headers.

When using **hist-to-div-for-cause-380** for 380 cause mapping, the SBC places those headers at the top of the Diversion header list. This is true regardless of how many History-Info entries are present.

Anonymization of History and Diversion Information

The SBC supports anonymization of entries for both History-Info and Diversion, independent of the interworking function. You configure this function separately from the interworking function. The SBC applies anonymization rules as long as the circumstances meet all anonymization criteria. This anonymization configuration is global. For untrusted peers, the SBC changes input header addresses to the anonymous SIP URI syntax **sip:anonymous@anonymous.invalid**.

This feature uses functionality independent of the IWF to confirm whether or not the remote target is trusted. The SBC checks the **session-agent** trust mode. If the **session-agent** reports "trust-me", then assuming nothing else conflicts, the system trusts the agent. This functionality also refers to the IWF state.

If IWF is configured, the SBC performs anonymization on the output generated after conversion per RFC 7544.

If IWF is not configured:

- Remote peer trusted - The SBC does not anonymize the History-Info header or Diversion header. The Privacy header field values in incoming History-Info or Diversion header of INVITE shall be used likewise in outgoing header.
- Remote peer untrusted - The SBC checks **sip-config**, **anonymize-history-for-untrusted** setting, introduced by this feature, to determine whether it should anonymize the entries in History-Info or Diversion headers with Privacy fields set to full/history. If **anonymize-history-for-untrusted** is configured, the SBC anonymizes the input received in the initial INVITE.

The SBC uses the following steps when it is sending messages to untrusted peer.

1. Based on the mode set, do the conversion between headers.
2. If mode is not set, check for History-Info and Diversion headers entries in outgoing message.
3. For the entries where privacy value is other than "none" in History-Info and "off" in Diversion, anonymize the entries if **anonymize-history-for-untrusted** is set.
4. For HI entries with a privacy value other than "none", and for diversion entries that have a privacy value other than "off", the SBC does not anonymize entries if is not set.
5. In case the entries have Privacy settings as "none" in History-Info or "off" in Diversion, do not anonymize the entries even if **anonymize-history-for-untrusted** is set.

If a privacy header is present in INVITE with value as "history" or "header" or "full" and the outgoing INVITE is going to untrusted remote side, then all the History-Info and Diversion

entries shall be anonymized as per above section, given **anonymize-history-for-untrusted** is set.

The SBC anonymizes all History-Info and Diversion entries regardless of the value of the privacy header of each History-Info or Diversion entry. If a privacy header with above values appears in incoming INVITE, every History-Info and Diversion entry will be anonymized.

Diversion and History-Info Headers Interworking Configuration

You can configure Oracle Communications Session Border Controllers to map call transfer and call diversion information between Diversion and History-Info headers.

To configure interworking between the Diversion and History-Info headers:

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

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

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **diversion-info-mapping-mode**— Configure this parameter to specify how the Diversion and History-Info headers map to and work with each other on the interface. The default value is **none**.
   - **div2hist** — any Diversion headers in the initial INVITEs going out of this sip-interface will be converted to History-Info headers before sending
   - **force** — behavior is the same as **div2hist** when a Diversion header is present in the incoming INVITE. If there are no Diversion headers, a History-Info header for the current URI is added in the outgoing INVITE.
   - **hist2div** — any History-Info headers in the initial INVITEs going out of this sip-interface will be converted to Diversion headers before sending
   - **none** — no conversion applied (default)
4. Type **done** to save your configuration.

History-Info Cause 380 Parameter Interworking Configuration

You can configure Oracle Communications Session Border Controllers to map the History-Info cause 380 parameter information to Diversion headers.

To configure cause parameter interworking on a sip-interface:

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

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

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

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **hist-to-div-for-cause-380**— Configure this parameter to specify whether or not the system should perform cause parameter interworking. The default value is **disabled**.
   - **enabled**—Perform cause parameter interworking.
   - **disabled**—Do not perform cause parameter interworking.
   - **inherit**—Use the setting specified in the sip-config.
4. Type **done** to save your configuration.

Anonymize History Configuration

You can configure Oracle Communications Session Border Controllers to 'anonymize' History-Info and Diversion headers.

You configure History-Info and Diversion header anonymization globally at the sip-config.

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

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

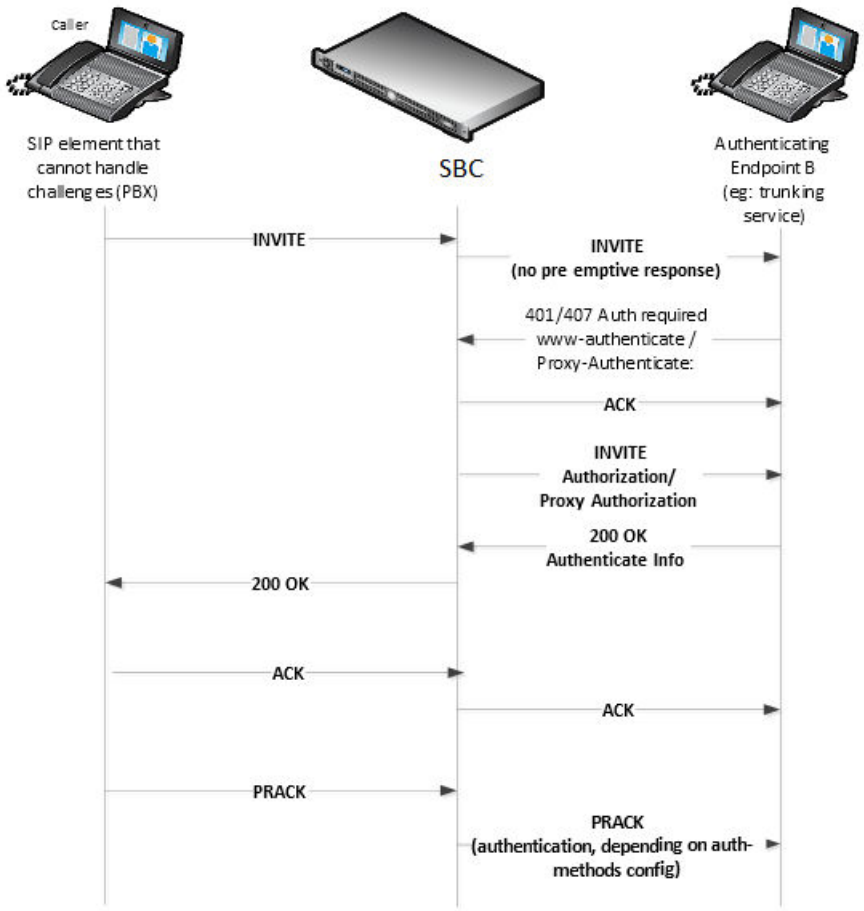
2. **anonymize-history-for-untrusted**— Configure this parameter to specify whether or not the system should anonymize parameter interworking. The default value is **disabled**.
   - **enabled**—Anonymize history interworking.
   - **disabled**—Do not anonymize history interworking.
3. Type **done** to save your configuration.

Digest Authentication with SIP

Digest authentication for Session Initiation Protocol (SIP) is a type of security feature on the SBC that provides a minimum level of security for basic Transport Control Protocol (TCP) and User Datagram Protocol (UDP) connections. Digest authentication verifies that both parties on a connection (host and endpoint client) know a shared secret (a password). This verification can be done without sending the password in the clear.

Digest authentication is disabled by default on the SBC. When digest authentication is enabled, the SBC (host) responds to authentication challenges from SIP trunking Service Providers (endpoint client). The SBC performs authentication for each IP-PBX initiating the call. However, the authentication challenge process takes place between the host and the

client only since the IP-PBX cannot handle authentication challenges. The following illustration shows the digest authentication process.



The digest authentication scheme is based on a simple challenge-response paradigm. A valid response contains a checksum of the “username” and password. In this way, the password is never sent in the clear.

By default, the SBC uses cached credentials for all requests within the same dialog, once the authentication session is established with a 200OK from the authenticating SIP element. If the in-dialog-methods attribute contains a value, it specifies the requests that have challenge-responses inserted within a dialog.

In digest authentication with SIP, the following can happen:

- More than one authenticating SIP element (IP-PBX) may be the destination of requests.
- More than one authentication challenge can occur in a SIP message. This can occur when there are additional authenticating SIP elements behind the first authenticating SIP element.
- The SBC distinguishes whether the IP-PBX is capable of handling the challenge. If Digest Authentication is disabled (no auth-attributes configured) on the Session Agent, the challenge is passed back to the IP-PBX.

> **Note:**
>
> If there are multiple challenges in the request, and if the SBC has only some of the cached credentials configured, the SBC adds challenge-responses for the requests it can handle, and does not pass the challenge back to the IP-PBX.

Challenge-Responses in Requests not in the Dialog

A digest authentication session starts from the client response to a www-authenticate/proxy-authenticate challenge and lasts until the client receives another challenge in the protection space defined by the auth-realm. Credentials are not cached across dialogs; however, if a User Agent (UA) is configured with the auth-realm of its outbound proxy, when one exists, the UA may cache credentials for that auth-realm across dialogs.

> **Note:**
>
> Existing Oracle Communications Session Border Controller behavior with surrogate-agents is that they cache credentials from REGISTER for INVITE sessions only if the Oracle Communications Session Border Controller is considered a UA sending to its outbound proxy.

Configuring Digest Authentication

In the Oracle Communications Session Border Controller ACLI, you can access the Digest Authentication object at the path **session-router**, **session-agent**, **auth-attribute**. If enabled, the Digest Authentication process uses the attributes and values listed in this table.

> **Note:**
>
> If enabling Digest Authentication, all attributes listed below are required except for the in-dialog-methods attribute which is optional.

The following table lists the digest authentication object

```
ORACLE(auth-attribute)# show
        auth-attribute
                auth-realm                      realm01
                username                        user
                password                        ********
                in-dialog-methods               ACK INVITE SUBSCRIBE
```

To configure digest authentication on the Oracle Communications Session Border Controller:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type session-agent and press Enter to access the session agent-related attributes.

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

4. Type auth-attribute and press Enter to access the digest authentication-related attributes.

```
ORACLE(session-agent)# auth-attribute
ORACLE(auth-attribute)#
```

5. auth-realm — Enter the name (realm ID) of the host realm initiating the authentication challenge. This value defines the protected space in which the digest authentication is performed. Valid value is an alpha-numeric character string. Default is blank.

```
ORACLE(auth-attribute)# auth-realm realm01
```

6. username — Enter the username of the client. Valid value is an alpha-numeric character string. Default is blank.

```
ORACLE(auth-attribute)# username user
```

7. password — Enter the password associated with the username of the client. This is required for all LOGIN attempts. Password displays while typing but is saved in clear-text (i.e., *****). Valid value is an alpha-numeric character string. Default is blank.

```
ORACLE(auth-attribute)# password *******
```

8. in-dialog-methods — Enter the in-dialog request method(s) that digest authentication uses from the cached credentials. Specify request methods in a list form separated by a space enclosed in parentheses. Valid values are:

- INVITE | BYE | ACK | CANCEL | OPTIONS | SUBSCRIBE | PRACK | NOTIFY | UPDATE | REFER

```
ORACLE(auth-attribute)# in-dialog-methods (ack invite subscribe)
```

> **Note:**
>
> The methods not in this list are still resubmitted if a 401/407 response is received by the Oracle Communications Session Border Controller.

If you do not specify any in-dialog-method value(s), digest authentication does not add challenge-responses to in-dialog requests within a dialog.

This attribute setting applies to in-dialog requests only.

Additional Notes

The following are additional notes that describe the digest authentication process:

- The Oracle Communications Session Border Controller always challenges the first LOGIN request, and initial authentication begins with that request. The recalculated authorization key — the credentials — are then included in every subsequent request.
- If the Oracle Communications Session Border Controller does not receive any communication from the client within the expiration period, the Oracle Communications Session Border Controller logs the client out and tears down the transport connection. Faced with interface loss, the Oracle Communications Session Border Controller default behavior is to flush all warrant information from the target database. This response necessitates that the client first login/re-register with the Oracle Communications Session Border Controller, and then repopulate the empty database using a series of ADD requests. This behavior ensures that client and Oracle Communications Session Border Controller target databases are synchronized.

  Alternatively, when faced with interface loss, the Oracle Communications Session Border Controller can retain all warrant information within the target database. This response necessitates only that the client first login/re-register with the Oracle Communications Session Border Controller. After successful registration the client should, but is not required to, use a series of GET, ADD, and DELETE requests to ensure that the Oracle Communications Session Border Controller and client target databases are synchronized.
- The Oracle Communications Session Border Controller ignores the Authentication-Info header that comes in the 200OK response after digest authentication is complete. The Oracle Communications Session Border Controller receives a 401/407 response from the client. However, some surrogate-agents may process the Authentication-Info header in a single challenge.

Digest Authentication and High Availability

The Oracle Communications Session Border Controller supports digest authentication in high availability (HA) environments. The session-agent configuration, which includes the digest authentication parameters on the primary Oracle Communications Session Border Controller, are replicated on the HA Oracle Communications Session Border Controller. However, cached credentials on the primary device are not replicated on the HA device.

Surrogate Agents and the SBC

In the case where a surrogate-agent is configured for the IP-PBX, you do not have to configure digest authentication attributes in the session-agent object for the same IP-PBX. The surrogate-agent authentication configuration takes precedence over the session-agent authentication configuration and so it is ignored.

The following illustration shows an example of a surrogate-agent with a session-agent in the network.

Surrogate Registration

The Oracle Communications Session Border Controller surrogate registration feature lets the Oracle Communications Session Border Controller explicitly register on behalf of a Internet Protocol Private Branch Exchange (IP-PBX). After you configure a surrogate agent, the Oracle Communications Session Border Controller periodically generates a REGISTER request and authenticates itself using a locally configured username and password, with the Oracle Communications Session Border Controller as the contact address. Surrogate registration also manages the routing of class from the IP-PBX to the core and from the core to the IP-PBX.

Registration

The Oracle Communications Session Border Controller uses the configuration information of the surrogate agent that corresponds to a specific IP-PBX. After the surrogate agents are loaded, the Oracle Communications Session Border Controller starts sending the REGISTER requests on their behalf. (You can configure how many requests are sent.)

The SIP surrogate agent supports the ability to cache authorization or proxy-authorization header values from a REGISTER 401, 407, and 200 OK messages and reuse it in subsequent requests, such as in an INVITE. This allows the Oracle Communications Session Delivery Manager to support authorization of subsequent requests in cases such as, when a customer PBX doesn't support registration and authentication. It also supports the generation of authorization/proxy-authorization header if subsequent requests get challenged with a 401/407 response.

If the Oracle Communications Session Border Controller receives 401 or 407 responses to REGISTER, requests, it will use the Message Digest algorithm 5 (MD5) digest authentication to generate the authentication information. You need to specify the password. The Oracle Communications Session Border Controller also supports authentication challenge responses with the quality of protection code set to auth (qop=auth), by supporting the client nonce (cnonce) and nonce count parameters.

The Oracle Communications Session Border Controller creates a registration cache entry for each of the AoRs for which it is sending the REGISTER requests. When the Oracle Communications Session Border Controller receives the associated URIs, it checks whether the customer host parameter is configured. If it is configured, the Oracle Communications Session Border Controller changes the host in the received Associated-URI to the customer host. If it is not configured, the Oracle Communications Session Border Controller does not change the Associated-URI. It makes the registration cache entries that correspond to each of the Associated-URIs. The From header in the INVITE for calls coming from the IP-PBX should have one of the Associated-URIs (URI for a specific phone). If the Oracle Communications Session Border Controller receives a Service-Route in the 200 (OK) response, it stores that as well.

The Oracle Communications Session Border Controller uses the expire value configured for the REGISTER requests. When it receives a different expire value in the 200 OK response to the registration, it stores the value and continues sending the REGISTER requests once half the expiry time has elapsed.

REGISTER requests are routed to the registrar based on the configuration. The Oracle Communications Session Border Controller can use the local policy, registrar host and the SIP configuration’s registrar port for routing.

If the Oracle Communications Session Border Controller is generating more than one register on behalf of the IP-PBX, the user part of the AoR is incremented by 1 and the register contact-user parameter will also be incremented by 1. For example, if you configure the register-user

parameter as caller, the Oracle Communications Session Border Controller uses caller, caller1, caller2 and so on as the AoR user.

Authenticating Registrations

There are two ways to configure the SBC to authenticate a registration using a surrogate agent. These include defining authentication criteria within a realm configuration and within the surrogate agent itself. Surrogate agent authentication, for both register requests and for call methods, are applications of digest authentication.

The first method uses **surrogate-agent** and **realm-config** configuration. This method is considered more robust, supporting multi-tenant, diverse IP-IPXs, and intra-realm registration support. The second method refers to the **surrogate-agent** configuration alone, applying to simpler deployments that, for example, do not request registration from a registrar outside of the IP-PBX's realm. The first method takes precedence, if configured, and also works when registrars and IP-IPXs reside on the same realm. The second method is considered a legacy configuration maintained for SBC deployments that have been using it. These methods are considered exclusive and the SBC throws a configuration verification error when it finds you have set up both methods on a surrogate agent.

When confronted with an authentication challenge to a surrogate agent's registration request, the SBC:

1. Checks whether you have configured the **auth-user-lookup** attribute in the **surrogate-agent**. If so, the SBC :
   a. Searches for a **realm-config** that contains an **auth-attribute** object with an **auth-user-lookup** value that is the same as the **auth-user-lookup** value in the **surrogate-agent**. An example **auth-user-lookup** setting is shown below.

      ```
      ORACLE(surrogate-agent)# auth-user-lookup TargetRealmAuthUser
      ```

   b. If found, the SBC issues a response to the challenge using the **username** and **password** values in the **auth-attribute** presented within the **realm-config**.
   c. If not found, the registration fails.
2. If the **auth-user-lookup** is empty, the SBC:
   a. Refers to the **auth-user** and **password** parameters in the surrogate configuration.
   b. If found, Issues a response to the challenge using those **auth-user** and **password** values.
   c. If not found, the registration fails.

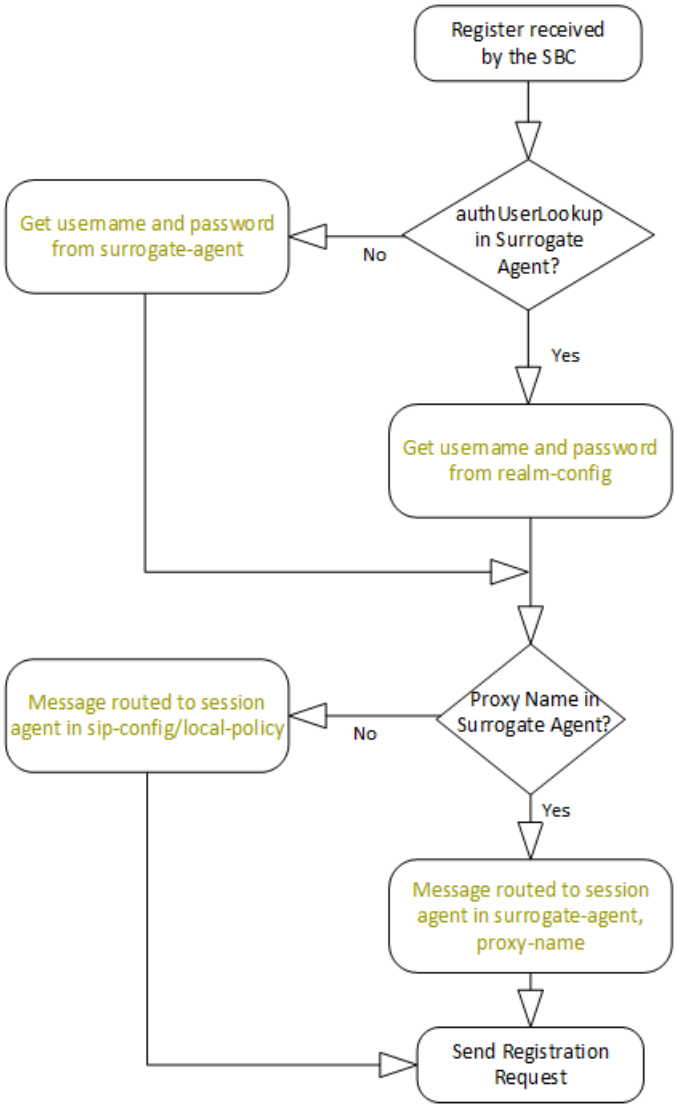
The SBC also has multiple means of routing the response to the correct registrar. The SBC follows this process to route the registration as well as the challenge response. To route to the correct registrar, the SBC:

1. Checks to see if you have configured the **proxy-name** parameter in the **surrogate-agent**. This **proxy-name** setting must match the **name** parameter in a **session-agent** you have configured for the registrar. An example **proxy-name** setting is shown below.

   ```
   ORACLE(surrogate-agent)# proxy-name MyProxyName
   ```

   If you have configured this parameter correctly, the SBC routes to that proxy.
2. If you not have configured the **proxy-name** parameter, the SBC routes the REGISTER using the **registrar-host** and **registrar-port** parameters in the **sip-config**.

3. If the system cannot route using the **sip-config**, the SBC attempts to route the response to the **next-hop** in a matching **local-policy** attribute.
4. If there is no match in any **local-policy**, the SBC replies to the source that the registration has failed.



Setting the un-register Parameter

Additional applicable **surrogate-agent** configuration allows you to enable the **un-register** parameter. Enabling this parameter causes the SBC to send the all REGISTER requests generated from this surrogate agent with Expires: 0 to the REGISTRAR, and remove all associated entries from the SBC registration-cache upon each registration.

```
ORACLE(surrogate-agent)# un-register enabled
```

Surrogate Agent Authentication Across Realms

The SBC uses an authentication attribute element in realms to support surrogate agent authentication requests initiated from other realms. This is a multi-instance element that supports the authentication of non-REGISTER traffic to surrogate agents with different authentication detail. The key for these lookups is the combination of the authentication realm

and the authentication user lookup configurations. If you do not configure authentication attribute element in the realm, the SBC handles surrogate agent authentication using the authentication table setup on the IP-PBX session agent, which supports this traffic in a single realm only.

This feature resolves two key surrogate agent authentication issues:

- In a multi-tenanted environment, there might be multiple IP-PBX systems or realms trying to use the same surrogate agent function. Therefore, there is a need to authenticate traffic to surrogate agents by the SBC on the SIP Trunk Realm with no association or relation to the IP-PBX system or realm.
- In addition, some SIP Trunk providers send 401/407 challenges to all outbound calls. The SBC utilizes the authentication table setup on the **session-agent** for username/password to response to the 401/407 challenge. However, if the **auth-realm** value is the same for all customers, then the SBC cannot provide the correct username/password in response to the 401/407 challenge. When this feature is not set up, the SBC always uses the first entry on the **auth-attributes** in the session agent's table for all outbound calls.

You configure the SBC to populate the authentication headers using your configuration from either the softswitch's surrogate agent or realm during a 401/407 authentication challenge. The SBC uses these tables to look up the authentication realm and obtain the username and password. The SBC uses the username and password to authenticate the request.

> **Note:**
>
> The device initiating the authentication challenge to the surrogate agent does not need to be a softswitch.

The SBC chooses the table based on your configuration:

- If you have configured the **auth-user-lookup** parameter in the **local-policy** attribute, then the SBC builds the authentication headers from the **realm-config** configuration.
- If the **auth-user-lookup** parameter in the **local-policy** attribute is empty, then the SBC builds the authentication headers from the IP-PBX **session-agent** configuration.

The SBC uses the following objects to perform this feature's function:

- A **local-policy** that forwards traffic to the target softswitch, which may or may not reside in a different realm from the source traffic. Further configuration supports intra-realm surrogate agent authentication.
- The **policy-attributes** within that **local-policy** that includes the target **auth-user-lookup** value and the target **realm** of the surrogate agent.
- An **auth-attributes** sub-element in the target **realm-config**. This sub-element includes:
  - The **realm** of the surrogate agent. This is the host realm initiating the authentication challenge. This value defines the protected space in which the digest authentication is performed.
  - An **auth-user-lookup** value that matches the **auth-user-lookup** in the **local-policy**.
  - The **username** and **password** that provide access to the target surrogate agent.

Cross Realm Surrogate Agent Authentication Operation

After configuration, the SBC authenticates applicable cross-realm surrogate agent authentication, allowing registration and call traffic.

When the SBC receives traffic that triggers your **local-policy**, it uses the policy's **auth-user-lookup** value in combination with the target realm of the **local-policy** to prepare for authentication.

The key used for these lookups is a combination of the authentication realm and the user lookup. Typical behavior during operation includes:

- If the **auth-user-lookup** in the **policy-attribute** is empty, the SBC gets the configured **password** for authorization from the softswitch **session-agent** configuration.
- If the **auth-user-lookup** in the **policy-attribute** is not empty, the SBC selects the AuthAttributes (username/password) for the matching **auth-user-lookup** in the **realm-config**.
- If the **auth-user-lookup** in **policy-attribute** is not empty, but there is no corresponding entry in **realm-config**, then the call fails as unauthorized. Additionally, the SBC displays a warning during **verify-config**.

When operating with the **auth-user-lookup** from the **local-policy** attributes, the SBC uses the returned value of **username** and **password** for authenticating the request. During the processing of the INVITE request, a reference to the selected local policy route is stored along with the route.

Additional Notes on Operation

The following are additional notes that describe the digest authentication process:

- The SBC always challenges the first LOGIN request, and initial authentication begins with that request. The selected authorization key — the credentials — are then included in every subsequent request.
- If the SBC does not receive any communication from the client within the expiration period, the SBC logs the client out and tears down the transport connection. Faced with interface loss, the SBC default behavior is to flush all warrant information from the target database. This response necessitates that the client first login/re-register with the SBC, and then repopulate the empty database using a series of ADD requests. This behavior ensures that client and SBC target databases are synchronized.

  Alternatively, when faced with interface loss, the SBC can retain all warrant information within the target database. This response necessitates only that the client first login/re-register with the SBC. After successful registration the client should, but is not required to, use a series of GET, ADD, and DELETE requests to ensure that the SBC and client target databases are synchronized.
- The SBC ignores the Authentication-Info header that comes in the 200OK response after digest authentication is complete. The SBC receives a 401/407 response from the client. However, some surrogate-agents may process the Authentication-Info header in a single challenge.

Routing Calls from an IP-PBX

When it receives a call from an IP-PBX configured as a surrogate agent, the SBC attempts to route, and if challenged, perform authentication on behalf of the IP-PBX to authorize the call. It does this by validating the request with your configuration. After routing the call to its destination, the callee may challenge the call, in which case the SBC has an opportunity to authenticate the call. If the SBC cannot authenticate the call, it leaves authentication procedures to other devices, such as the IP-PBX itself.

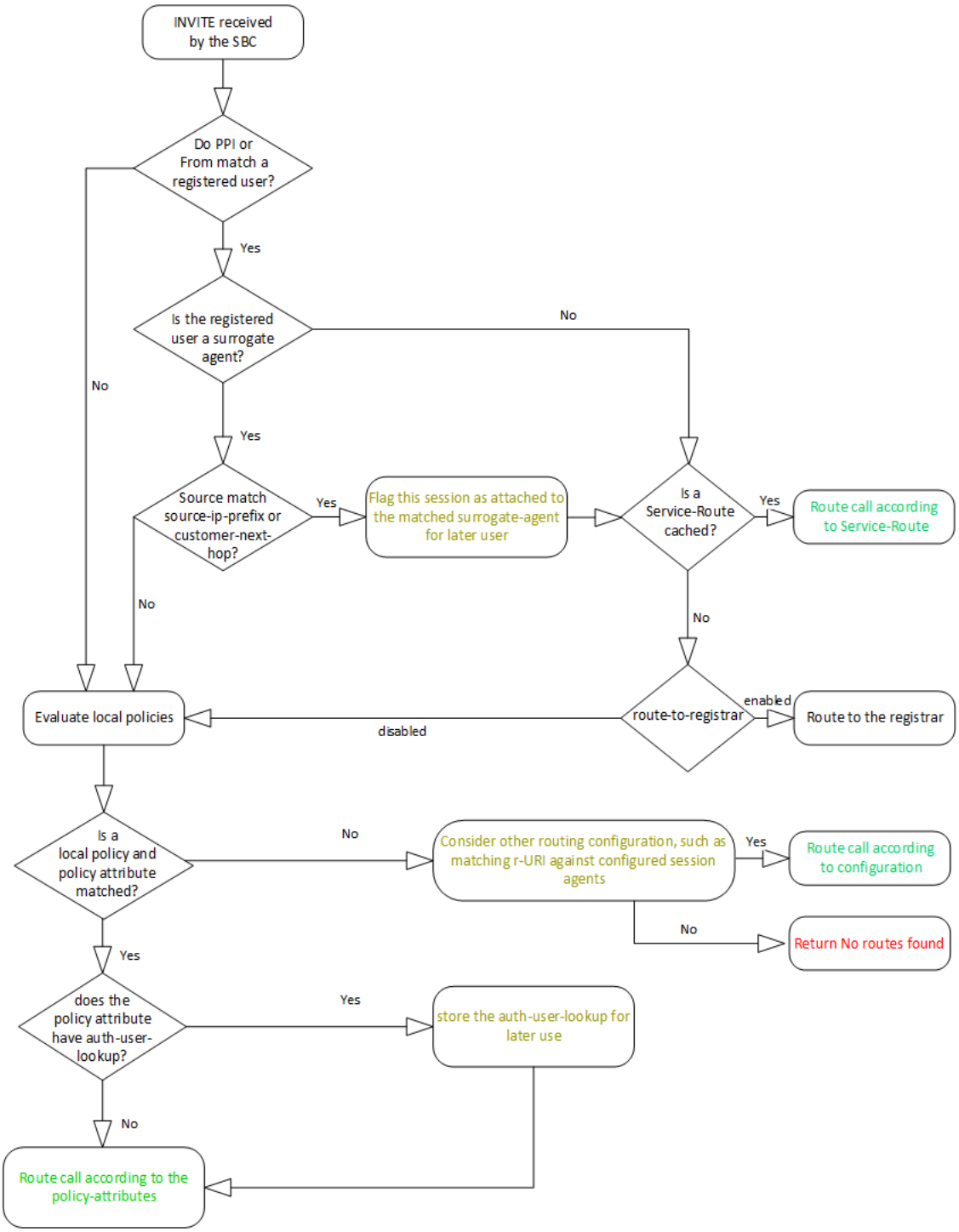
The process of routing a call from an IP-PBX using a Surrogate Agent fits within the overall routing process, as shown in the following diagram. While determining a route, the SBC also

determines whether the call may be authenticated using the SBC as a surrogate. The call may be routed by multiple processes, each of which can include a **surrogate-agent** match and specifying that the SBC can trust the caller.

To route the call, the SBC looks for a service route. After finding the corresponding registration Service-Route entry, the SBC uses the Service-Route for this endpoint to route the call, if it exists.

If no Service-Route exists, but the **route-to-registrar** parameter is enabled on the **sip-interface**, the SBC tries to route the call to the registrar. If **route-to-registrar** is disabled, the SBC refers to **local-policy** for routing.

At this point, the SBC routes the call, and is prepared if it needs to respond to an authentication challenge.



Having received a challenge, the SBC matches the challenge with source and SIP information presented by the caller and stored by the SBC. This matching process is explained below. If the match is successful, the SBC authenticates the call on behalf of the IP-PBX. After authentication succeeds, media between the caller and callee may proceed.

> **Note:**
>
> You can configure the **surrogate-agent** to override the **sip-interface**, **route-to-register** setting. If the surrogate agent's **route-to-registrar** parameter is set to disable, it takes precedence over the **sip-interface** setting. In this case, the SBC does not try to route the call to the registrar.

Matching an INVITE to a Surrogate Agent

You can configure the SBC to perform authentication for an end station sending an INVITE through its surrogate agent in two ways. These methods match information in your **surrogate-agent**, and in some cases **session-agent** configurations, with source information in the request. The SBC uses this function for requests needing authentication, except REGISTERs. This processing is not relevant to calls sent to a **surrogate-agent**.

Assuming configuration, to confirm and authenticate requests originating from a surrogate agent, the SBC matches:

1. Information in the P-Preferred-Identity (PPI) or the FROM header in the request to a registered user, and then your **register-user** and **register-host** configuration in your **surrogate-agent**. These matches confirm the **surrogate-agent** from which the request came. If these do not match, the SBC does not continue with authentication.

   > **Note:**
   >
   > The PPI or FROM header should contain the user portion of one of the Associated-URIs that it received from the registrar in the 200 (OK) responses to REGISTER requests. It should also have a hostname.

   > **Note:**
   >
   > Should the **register-host** match fail, the SBC tries to match the host portion with your **customer-host** configuration.

2. IP source addressing to the **source-ip-prefix** parameter within the **surrogate-agent** element. This confirms that the SBC can perform the authentication. If you have configured this parameter and there is no match, the SBC does not authenticate the request.
3. If there is no **source-ip-prefix** configuration, the SBC tries to match the **customer-next-hop** configuration in the **surrogate-agent** with the source of the incoming request. This can also confirm that the SBC can perform the authentication.

To perform this step, the SBC compares the source IP of the request with the **customer-next-hop** parameter. You can configure the **customer-next-hop** parameter with a **session-agent** name, a **session-agent-group** or any IP address. That value must match:

- The **hostname** Session Agent, or the **group-name** of the Session Agent Group.
- If configured with an IP address, your value must match the source IP of the request.

The **source-ip-prefix** configuration provides the SBC with the flexibility to perform the authentication against specific IP addresses, multiple prefixes and/or IP ranges. The **customer-next-hop** configuration allows for only a single entry, attempting to match to a single

address, a **hostname** configuration on a corresponding **session-agent** or the **group name** of the SAG to which the session agent belongs.

> **Note:**
>
> If **source-ip-prefix** is not configured and the **customer-next-hop** matches a **session-agent-group**, the SBC uses that group to choose a specific **session-agent**.

Process Detail

The steps below present SBC behavior when it receives a request from a **surrogate-agent** that needs authentication, other than a REGISTER. The steps use an INVITE as an example. To support these calls the SBC:

1. Determines if there is a **surrogate-agent** match by ensuring the PPI or FROM matches the **register-user** and the **register-host** or **customer-host** in the **surrogate-agent** agent configuration.

   > **Note:**
   >
   > The SBC attempts to match to the PPI only if you have enabled the **sip-ims-feature** on the applicable **sip-interface**.

   > **Note:**
   >
   > If there is no match with the **register-host**, the SBC attempts to match the applicable host portion with the **customer-host** configuration.

   - If not, the SBC attempts to use other configuration to proceed with the call, such as local policy. If those processes do not find a route, the SBC sends the caller with a 480 - No routes found message.
   - If so, the SBC attempts to verify the source of the INVITE to determine whether or not it can perform the authentication on behalf of the IP-PBX.
2. To determine if it should perform the authentication, the SBC checks whether the **surrogate-agent** has a **source-ip-prefix** configuration:
   - If so, the SBC matches the source IP address with any of your **source-ip-prefix** settings in the **surrogate-agent**.
     - If there is a match, the SBC considers the INVITE as originated from a trusted source and it can perform authentication on behalf of the **surrogate-agent**.
     - If it does not match, the SBC does not perform authentication on behalf of the **surrogate-agent**, and forwards a 401 with challenge towards the IP-PBX.
3. If you have not configured the **source-ip-prefix**, the SBC can next refer to your **customer-next-hop** configuration and check for a match. to the **surrogate-agent**. Detail on matching criteria is above.
   - If there is a match, the SBC considers the INVITE as originated from a trusted source and performs authentication on behalf of **surrogate-agent**.

- If it does not match, the SBC does not perform authentication on behalf of **surrogate-agent**, and forwards 401 with challenge towards the endpoint.

4. If there is a match, the SBC generates an INVITE with authentication parameters and sends it to the REGISTRAR to confirm the authentication.
5. If the authentication is successful, the SBC sends a 200 OK to the IP-PBX, and routes the call to the callee.
6. If the authentication fails, the SBC sends a 401 with challenge to the IP-PBX.
7. At this point, the endpoint may or may not attempt to authenticate itself directly with the REGISTRAR.
   - If not, the call does not proceed.
   - If so, and the authentication is successful, media may proceed between the caller and callee.
   - If so, and the authentication fails, the REGISTRAR replies to the IP-PBX directly with (an authentication failed message), and the call does not proceed.

Configuration Detail on Verifying Source IP

You configure the **source-ip-prefix** and the **customer-next-hop** parameters on the applicable **surrogate-agent**. The **source-ip-prefix** accepts any number of IP addresses and IP address prefixes in the format <ip>/<subnet>. If you set multiple values, separate them with a space and enclose them with parenthesis (). Addressing can be IPv4, IPv6 or a combination of both. The SBC performs individual match checks in the same order as your configuration.

For example, to configure the agent to trust IPs 192.168.1.0, 172.16.10.10 and 172.168.x.x, you can configure the parameter as follows.

```
ORACLE(surrogate-agent)#source-ip-prefix (192.168.1.0 172.168.0.0/16
172.16.10.10)
```

In contrast, the **customer-next-hop** parameter accepts a single entry as an IP address, FQDN, Session Agent name, or Session Agent Group name.

```
ORACLE(surrogate-agent)#customer-next-hop 192.168.1.0
```

The SBC prevents you from configuring parameters using an incorrect format.

Related Configuration

Note the following configurations and their impact on this authentication process.

- This feature fully supports HA deployments.
- If you disable the **sip-ims-feature** on the ingress **sip-interface**, the SBC uses only the FROM header to determine the associated **surrogate-agent**, ignoring any PPI header.

Call Flow Examples

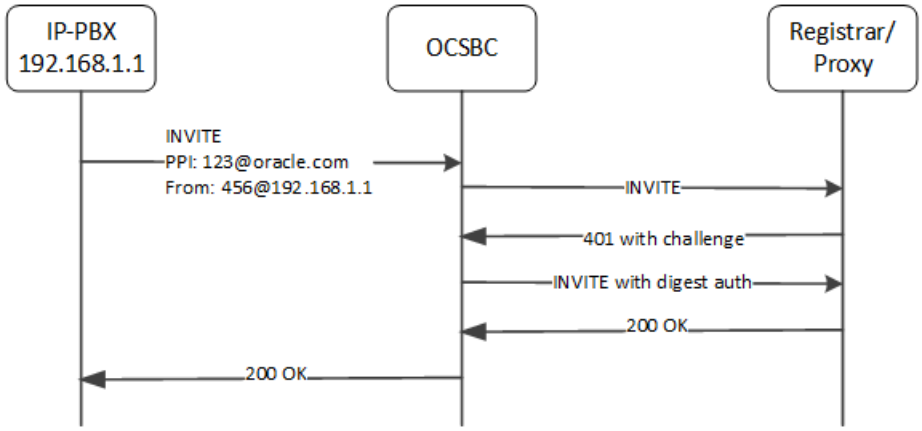
The call flows in this topic demonstrate how the SBC identifies the **surrogate-agent** for an incoming call for the purpose of authentication. Your **surrogate-agent**, **session-agent**, and **sip-interface** configuration are key to processing these calls.

Call Flow 1

This first call flow depicts the SBC successfully handling the end station authentication. In this case, the SBC identifies the **surrogate-agent** by matching PPI with the **register-user** and **register-host** parameters in the **surrogate-agent** configuration.

```
surrogate-agent:
      register-host: oracle.com
      register-user: 123
      source-ip-prefix: 192.168.0.0/16
```

Next, the SBC matches the source IP of the INVITE request with the **source-ip-prefix** parameter in the **surrogate-agent** configuration to determine if it should perform surrogate authentication. Note the source IP address, 192.168.1.1 falls in the range of the setting, 192.168.0.0/16.

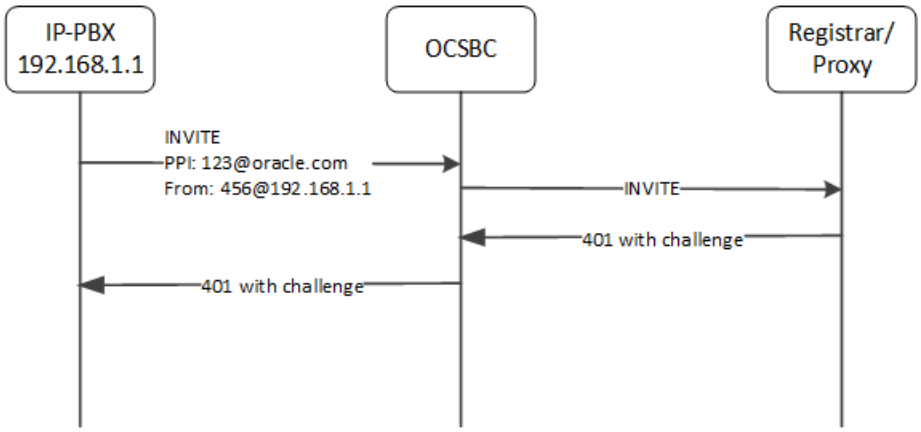


The SBC does not reference the **customer-next-hop** parameter because you have configured the **source-ip-prefix** with some value.

Call Flow 2

This next call flow depicts the SBC not handling an end station authentication. In this case, the **source-ip-prefix**, configured to 172.16.0.0/16, does not match the source IP address.

```
surrogate-agent:
      register-host: oracle.com
      register-user: 123
      source-ip-prefix: 172.16.0.0/16
```

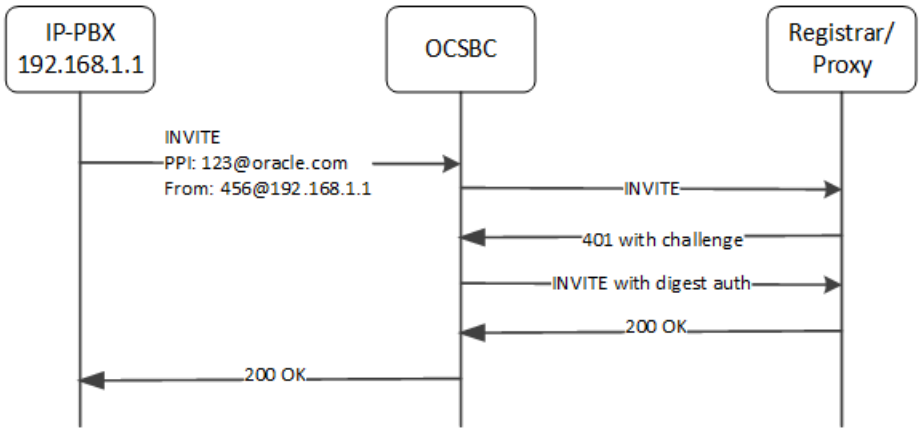


Again, the SBC does not reference the **customer-next-hop** parameter.

Call Flow 3

Consider this next case, wherein the **source-ip-prefix** is not configured. But the relevant configuration, shown below, includes a **customer-next-hop** configuration that provides a match.

```
surrogate-agent:
      register-host: oracle.com
      register-user: 123
      customer-next-hop: SA1
      source-ip-prefix: <empty>
session-agent:
      hostname: SA1
      ip-address: 192.168.1.1
```

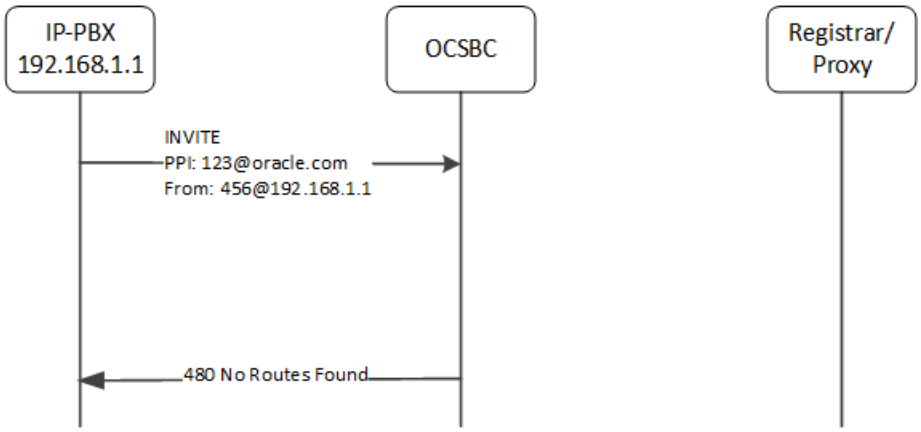


Call Flow 4

Consider this next case, wherein the **surrogate-agent** configuration fails to identify the source correctly. Because the **register-host** and **register-user** values do not match either the PPI or FROM presented in the INVITE, the SBC does not have a means of forwarding the INVITE, so it replies with '480 - No Routes Found' message.

```
surrogate-agent:
      register-host: acmepacket.com
      register-user: 486
```

```
customer-next-hop: <any value>
source-ip-prefix: <any value>
```



Separate from the feature, the SBC could use a **local-policy** or other routing configuration, to route this call.

Surrogate Agent Refresh on Invalidate

Surrogate agent registrations normally only re-register when nearing their expiration time. When a registrar fails, the surrogate agent will wait until the expiration time to refresh the registration with an in-service registrar.

You can configure your Oracle Communications Session Border Controller to immediately refresh the surrogate agent registrar with an in-service registrar by enabling the existing parameter **invalidate-registrations**.

Invalidate Registrations

An existing feature called **invalidate-registrations** located in the session agent keeps track of when surrogate agents go out of service. When REGISTER messages are received, registration entries that had out-of-service session agents since the last REGISTER will always allow the message through to the registrar (as opposed to responding directly from the cache).

The **invalidate-registrations** parameter in session agent configuration enables the Oracle Communications Session Border Controller to detect failed Registrar session agents.

If invalidate-registrations is enabled for the session agent, a response from a surrogate REGISTER that contains a service-route header that corresponds to a session-agent is installed to the registration cache entry.

The surrogate-agents are scanned. Surrogate agents with registration entries matching the out-of-service registrar have their timer reset to initiate a refresh. For an immediate refresh, the registration entry will only be considered when the service-route session agent goes out-of-service. The service-route session agent takes precedence and any previous registrar session agent w ill not be considered for an immediate refresh of the surrogate-agent registration.

Performance Impact

In cases with a large number of surrogate-agent registrations, there may be an impact to CPU usage when a session-agent goes out-of-service. All of the surrogate-agent registrations are scanned at that time. Refresh registrations are then sent out on timers.

Media Inactivity Timer Configuration

To disable the media inactivity timer for calls placed on hold:

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

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

2. Type **session-router** and press Enter.

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

3. Type **session-agent** and press Enter.

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

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

4. **invalidate-registration**—Set this parameter to **enabled** if you want to use the manual trigger to send this session agent offline (and therefore invalidate the registrations associated with it). The default is **disabled**.
5. Save and activate your configuration.

Surrogate Agents Re-Register after Switchover

You can configure the SBC to re-register all of its surrogate agents immediately after an HA switchover. This behavior is valid for all surrogate agent, regardless of transport, but is limited to 10,000 surrogate agent registrations to prevent a registration avalanche. If the system has more than 10,000 surrogate agent registrations to perform, the remaining agents re-register normally.

Surrogate agents normally re-register only within configured windows. After an HA switchover, however, surrogate agents begin using different interfaces (on the standby SBC) thereby immediately breaking the connections, and taking the surrogate agent down. Although every surrogate agent eventually re-registers and becomes operational again without intervention, the delays can disrupt call traffic. To avoid these disruptions, you can enable the **surrogate-reg-switchover** parameter in the **sip-config**.

```
ORACLE(sip-config)surrogate-reg-switchover enabled
```

With this parameter enabled, the SBC re-registers up to 10,000, immediately after the standby becomes active. This allows the SBC to prevent some of the service disruptions between its surrogate agents and the target devices. If the system needs to perform more than 10,000 registrations, it proceeds with the remaining surrogate agent registrations when the timer for those entries expire.

> **Note:**
>
> The system performs its refresh registration processes on an on-going basis. There are no restrictions that limit refresh registrations outside of this feature. This means that refresh registrations can impact CPU performance outside of this switchover feature.

When you save your configuration after enabling this feature, the SBC throws a **verify-config** warning that says "**surrogate-reg-switchover** sends all surrogate re-registrations immediately upon switchover, so it is limited to 10,000 registrations".

> **Note:**
>
> Enabling **dyn-contact-method=randomseed** as an **spl-option** significantly increases the CPU and sipd thread utilization which could lead to crashes. If you set the **surrogate-reg-switchover** parameter and the **dyn-contact-method=randomseed** option, the SBC becomes overloaded before it reaches 10,000 registrations. Configuring both of these features simultaneously is not supported.

Surrogate Registration Configuration

Surrogate registration allows the SBC to explicitly register endstations on behalf of an IP-PBX. Surrogate registration also manages the routing of calls to and from the IP-PBX and core (registrar).

Configure multiple elements depending on your deployment's design for establishing IP-PBX proxies that you want the SBC to register. Elements include:

- surrogate-agent
- realm-config
- session-agent
- local-policy

Configuring Surrogate Registration

You can configure surrogate registration using the ACLI. You need to configure a surrogate agent for each IP-PBX proxy for which the SBC registers. Those parameters that are optional are marked, the rest are mandatory.

To configure the surrogate agent:

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

   ```
   ACMEPACKET# configure terminal
   ```

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

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

3. Type **surrogate-agent** and press Enter. The prompt changes to indicate you can configure individual parameters.

```
ACMEPACKET(session-router)# surrogate-agent
ACMEPACKET(surrogate-agent)#
```

   From this point, you can configure surrogate agent parameters. To view all surrogate agent configuration parameters, enter a **?** at the system prompt.

4. **register-host**—Enter the registrar's hostname to be used in the Request-URI of the REGISTER request. This name is also used as the host portion of the AoR To and From headers.
5. **register-user**— Enter the user portion of the AoR (Address of Record).
6. **state**—Set the state of the surrogate agent to indicate whether the surrogate agent is used by the application. The default value is **enabled**. The valid values are:
   - enabled | disabled
7. **realm-id**— Enter the name of realm where the surrogate agent resides (where the IP-PBX proxy resides). There is no default.
8. **description**—Optional. Enter a description of this surrogate agent.
9. **customer-host**—Optional. Enter the domain or IP address of the IP-PBX, which is used to determine whether it is different than the one used by the registrar.
10. **customer-next-hop**—Enter the next hop to this surrogate agent:
    - session agent group:

      ```
      SAG: <session agent group name>
      ```

    - session agent:

      ```
      <hostname> or <IPV4>
      ```

    - specific IP address:

      ```
      <IPV4> or <IPV4: port>
      ```

11. **register-contact-host**—Enter the hostname to be used in the Contact-URI sent in the REGISTER request. This should always point to the SBC. If specifying a IP address, use the egress interface's address. If there is a SIP NAT on the registrar's side, use the home address in the SIP NAT.
12. **register-contact-user**—Enter the user part of the Contact-URI that the SBC generates.
13. **password**—If you are configuring the auth-user parameter, you need to enter the password used in case the registrar sends the 401 or 407 response to the REGISTER request.
14. **register-expires**—Enter the expires in seconds to be used in the REGISTER requests. The default value is 600,000 (1 week). The valid range is:
    - Minimum—0
    - Maximum—999999999
15. **replace-contact**—This specifies whether theSBC needs to replace the Contact in the requests coming from the surrogate agent. If this is enabled, Contact will be replaced with

the Contact-URI the SBC sent in the REGISTER request. The default value is **disabled**. The valid values are:

- enabled | disabled

16. **route-to-registrar**—This indicates whether requests coming from the surrogate agent should be routed to the registrar if they are not explicitly addressed to the SBC. The default value is **enabled**. The valid values are:
    - enabled | disabled
17. **aor-count**—Enter the number of registrations to do on behalf of this IP-PBX. If you enter a value greater than **1**, the SBC increments the register-user and the register-contact-user values by that number. For example, if this count is 3 and register-user is john then users for three different register messages will be john, john1, john2. It does the same for the register-contact-user values. The default value is **1**. The valid range is:
    - Minimum—0
    - Maximum—999999999
18. **auth-user**—Enter the authentication user name you want to use for the surrogate agent. This name is used when the SBC receives a 401 or 407 response to the REGISTER request and has to send the REGISTER request again with the Authorization or Proxy-Authorization header. The name you enter here is used in the Digest username parameter. If you do not enter a name, the SBC uses the value of the register-user parameter.
19. **max-register-attempts**—Enter the total number of times to attempt registration until success. The default value is **3**. The valid range is:
    - Minimum—0 (No Limit)
    - Maximum—10
20. **register-retry-time**—Enter the time to wait after an unsuccessful registration before re-attempting. The default value is **300**. The valid range is: Range 30-3600
    - Minimum—30
    - Maximum—3600
21. **count-start**—Enter the starting value for numbering when performing multiple registrations. The default value is **1**. The valid range is:
    - Minimum—0
    - Maximum—999999999
22. **register-mode**—Select automatic (default) or triggered (upon trigger from PBX).
    - automatic | triggered
23. **triggered-inactivity-interval**—Enter the maximum time with no traffic from the corresponding PBX. (Valid only with Triggered inactivity interval.) The default value is **30**. The valid range is:
    - Minimum—5
    - Maximum—300
24. **triggered-oos-response**—503 (Default. Send 503 response for core network failure) or drop response, which causes the system to not respond to PBX or core network failure.
    - 503 | dropresponse
25. **source-ip-prefix**—Contains a list of IP address/prefixes that specify the source addressing of endpoints the system can authenticate using this surrogate-agent. Valid entries include any number of IP addresses and IP address prefixes in the format <ip>/<subnet>. If you

set multiple values, separate them with a space and enclose them with parenthesis (). Addressing can be IPv4, IPv6 or a combination of both. The default configuration is null (no entry).

26. **auth-user-lookup**—If you intend to authenticate register requests using a realm configuration, enter the name of the target Auth Attribute configuration in that realm. This name must match an Auth User Lookup name in the realm's Auth Attribute list. When configured, the SBC uses those credentials to authenticate challenged register requests.
27. **proxy-name**—If you have configured the Registrar that validates this surrogate agent's register requests as a session agent, enter the name of that session agent here.
28. **un-register**—Enable this parameter to cause the register requests from this surrogate agent to specify Expires:0 and to remove each of this surrogate agents entries from the registration cache. The default value is **disabled**. The valid values are:
    - enabled | disabled
29. Save and activate your configuration.

Example

The following example shows the surrogate agent configuration.

```
surrogate-agent
register-host    acmepacket.com
register-user    234567
state    enabled
realm-id    public
description
customer-host    acmepacket.com
customer-next-hop    111.222.33.44
register-contact-host    111.222.5.68
register-contact-user    eng
password
register-expires    600000
replace-contact    disabled
route-to-registrar    enabled
aor-count    1
source-ip-prefix
options
auth-user
max-register-attempts   10
register-retry-time   30
count-start   1
register-mode   automatic
triggered-inactivity-interval   30
triggered-oos-response   503
auth-user-lookup
proxy-name charlie
un-register disabled
last-modified-date    2006-05-04 16:01:35
```

Configure Authentication Attributes on a Realm

In the SBC ACLI, you can access authentication parameters applicable to surrogate agent operation using the path **media-manager**, **realm-config**, **auth-attributes**. If using this

authentication method for register requests, this feature uses the attributes and values listed in this table. You perform this configuration to the **realm-config** on which the registrar resides.

> **Note:**
>
> If enabling this means of authentication, all the **auth-attributes** listed below are required except for the in-dialog-methods attribute, which is optional.

To configure authentication on the SBC:

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

   ```
   ORACLE# configure terminal
   ```

2. Type media-manager and press Enter to access the media manager-related objects.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type realm-config and press Enter.

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

4. Create or select the realm-config on which the softswitch resides.
5. Type auth-attributes and press Enter to access the authentication-related attributes.

   ```
   ORACLE(realm-config)# auth-attributes
   ORACLE(auth-attributes)#
   ```

6. auth-realm — Enter the identifier of this realm, which initiates the authentication challenge. This value defines the protected space in which the authentication is performed. Valid value is an alpha-numeric character string. Default is blank.

   ```
   ORACLE(auth-attributes)# auth-realm realm01
   ```

7. username — Enter the username of the client. Valid value is an alpha-numeric character string. Default is blank.

   ```
   ORACLE(auth-attributes)# username user
   ```

8. auth-user-lookup — Enter a name for this auth-user-lookup. You use this same name when configuring the **auth-user-lookup** within the attributes of a local-policy, and within the surrogate agent itself. Default is blank.

   ```
   ORACLE(auth-attributes)# auth-user-lookup reg1
   ```

9. password — Enter the password associated with the username of the client. This is required for all LOGIN attempts. Password displays while typing but is saved in clear-text (i.e., *****). Valid value is an alpha-numeric character string. Default is blank.

   ```
   ORACLE(auth-attributes)# password *******
   ```

10. in-dialog-methods — Enter the in-dialog request method(s) that authentication uses from the cached credentials. Specify request methods in a list form separated by a space enclosed in parentheses. Valid values are:
    - INVITE
    - BYE
    - ACK
    - CANCEL
    - OPTIONS
    - SUBSCRIBE
    - PRACK
    - NOTIFY
    - UPDATE
    - REFER

    ```
    ORACLE(auth-attributes)# in-dialog-methods (ack invite subscribe)
    ```

    If you do not specify any in-dialog-method value(s), authentication does not add challenge-responses to in-dialog requests within a dialog. This attribute setting applies to in-dialog requests only.

    > **Note:**
    >
    > The methods not in this list are still resubmitted if a 401/407 response is received by the Oracle Communications Session Border Controller.

11. Type done to save changes to this realm-config.
12. Save and activate your configuration.

Configure the applicable **local-policy**. This configuration includes setting the **auth-user-lookup** parameter in the applicable **local-policy-attribute** with the same value as the **auth-user-lookup** above.

Configure a Local Policy for Authenticating Surrogate Agent Traffic

To configure a local policy to support intra-realm surrogate agent authentication, you configure the local policy that directs traffic from the surrogate agent to the softswitch, which initiates the authentication challenge for any traffic coming from the surrogate agent (usually a PBX without the ability to authenticate itself).

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

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

```
ACMEPACKET(session-router)# local-policy
ACMEPACKET(local-policy)#
```

4. **from-address**—Indicate the originating address information by entering a From address value. You can use the asterisk (*) as a wildcard to indicate this policy can be used with all originating addresses.

> **Note:**
>
> After entering the from-address value, the Oracle Communications Session Delivery Manager automatically saves it to the configuration when exiting from local policy.

5. **to-address**—Indicate the destination address by entering a To address value. You can use the asterisk (*) as a wildcard to indicate all this policy can be used for any destination address.

> **Note:**
>
> After entering the to-address value, the Oracle Communications Session Delivery Manager automatically saves it to the configuration when exiting from local policy.

6. **source-realm**—Enter the identifier of the realm on which the surrogate agent resides.
7. **state**—Indicate whether you want the local policy to be enabled or disabled on the system. The default value is **enabled**. The valid values are:
   - enabled | disabled
8. **policy-attribute**—Configure local policy attributes required for this feature. All other attributes are optional.
9. **next-hop**—Identify the next signaling host by entering the next hop value. For this feature, then next hop is the soft switch.
10. **realm**—Identify the egress realm (the realm used to reach the next hop) if the system must send requests out from a specific realm.
11. **lookup**—Set this parameter to single.
12. **auth-user-lookup**—Enter the name of the target auth-user-lookup you have configured for this surrogate agent on the Softswitch realm.
13. Type done twice to save changes to your policy-attributes and your local policy.
14. Save and activate your configuration.

Recurse 305 Only Redirect Action

The Oracle Communications Session Border Controller has a SIP feature called redirect action. This is a feature that allows the Oracle Communications Session Border Controller, acting as a SIP Proxy or a Session Agent, to redirect SIP messages after receiving a SIP redirect (3xx) response. Previously, for the ACLI objects of sip-interface and session-agent on the Oracle Communications Session Border Controller, you could set the redirect-action

parameter to **proxy** or **recurse**. In Release 6.3 you can additionally set a value of **recurse-305-only** for the redirect-action parameter.

Redirect Action Process

When the redirect-action parameter is set to proxy, the Oracle Communications Session Border Controller sends SIP Redirect responses back to the previous hop (back to the User Agent Client (UAC)) when the User Agent Server (UAS) is not a session agent. The URI in the Contact of the response is changed from the URI that was in the original request.

> **Note:**
>
> If the target of the request is a session agent, the session agent's redirect action supercedes that of the SIP interface.

When the redirect-action parameter is set to recurse, if the Oracle Communications Session Border Controller receives a SIP redirect (3xx) response on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 3xx response. The responses contain the same Contact URI that was in the original request sent to the UAS.

For example, if UAC X sends an INVITE to the Oracle Communications Session Border Controller set up as a SIP proxy, the Oracle Communications Session Border Controller forwards the INVITE to UAS Y (Y is not a session agent). Y then responds to the Oracle Communications Session Border Controller with a 3xx response (redirection message) with the same URI that was in the original request. This indicates to the Oracle Communications Session Border Controller that if it receives any future requests directed toward Y, that it should automatically redirect the request directly to Y. The Oracle Communications Session Border Controller then recurses, or repeatedly sends subsequent incoming messages to the Contact URI specified in the Header of the 3xx responses.

When the redirect-action parameter is set to recurse-305-only, if the Oracle Communications Session Border Controller receives a 305 SIP redirect response (Use Proxy) on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 305 response. All other 3xx responses are sent back to the previous hop.

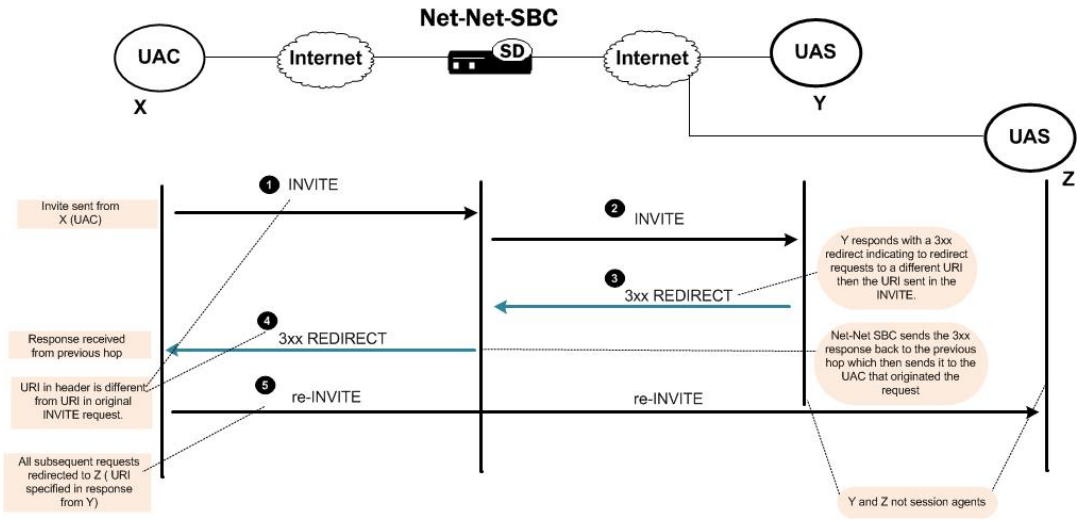
When the UAS is a session agent, the Oracle Communications Session Border Controller can send the SIP redirect response back to the UAC using the value in the session agent's redirect action field. If there are too many UASs to define as individual session agents, or if the UASs are Hosted NAT Traversal (HNT) endpoints, and SIP redirect responses need to be proxied for UASs that are not session agents, you can set the behavior at the SIP interface level.

Redirect-Action Set to Proxy

The following occurs if you set the **redirect-action** parameter to proxy on the SBC:

1. X (UAC) sends an INVITE to the SBC.
2. The SBC forwards the INVITE to Y (UAS).
3. Y sends the 3xx REDIRECT response to the SBC with a different URI in the message header.
4. The SBC forwards the 3xx REDIRECT response to the previous hop. X receives the 3xx REDIRECT response from the previous hop.
5. X redirects all subsequent requests to the URI in the message header received from Y.

The following illustration shows an example of a dialog between X, Y, Z, and the SBC during a redirect-action session set to proxy.

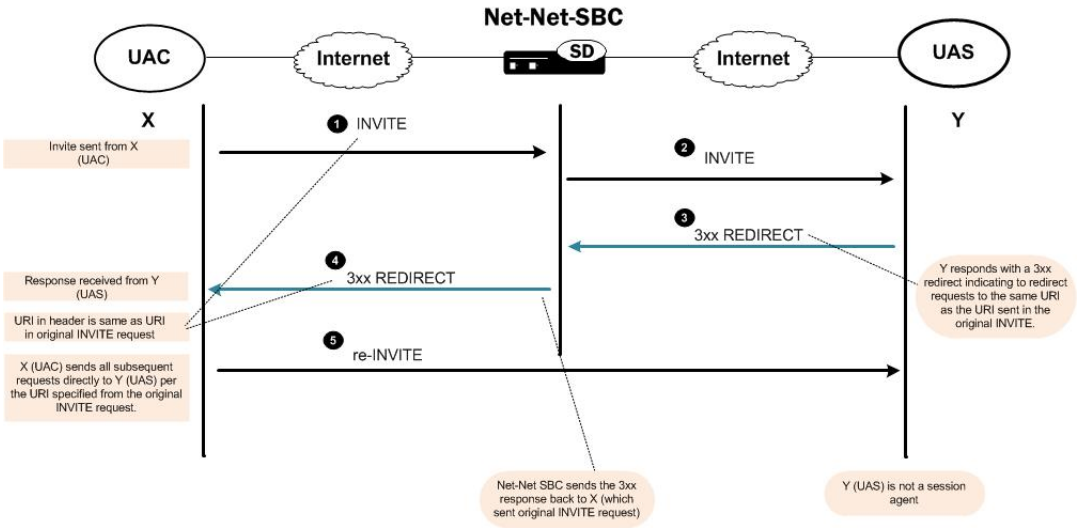


Redirect-Action Set to Recurse

The following occurs if you set the **redirect-action** parameter to recurse on the Oracle Communications Session Border Controller:

1. X (UAC) sends an INVITE to the Oracle Communications Session Border Controller.
2. The Oracle Communications Session Border Controller forwards the INVITE to Y (UAS).
3. Y sends the 3xx REDIRECT response to the Oracle Communications Session Border Controller with the same URI as the URI sent in the original request.
4. The Oracle Communications Session Border Controller forwards the 3xx REDIRECT response to X (UAC).
5. X (UAC) sends all subsequent requests directly to Y (UAS) per the URI specified from the original INVITE request.

   The following illustration shows an example of a dialog between X, Y, and the Oracle Communications Session Border Controller during a redirect-action session set to recurse.

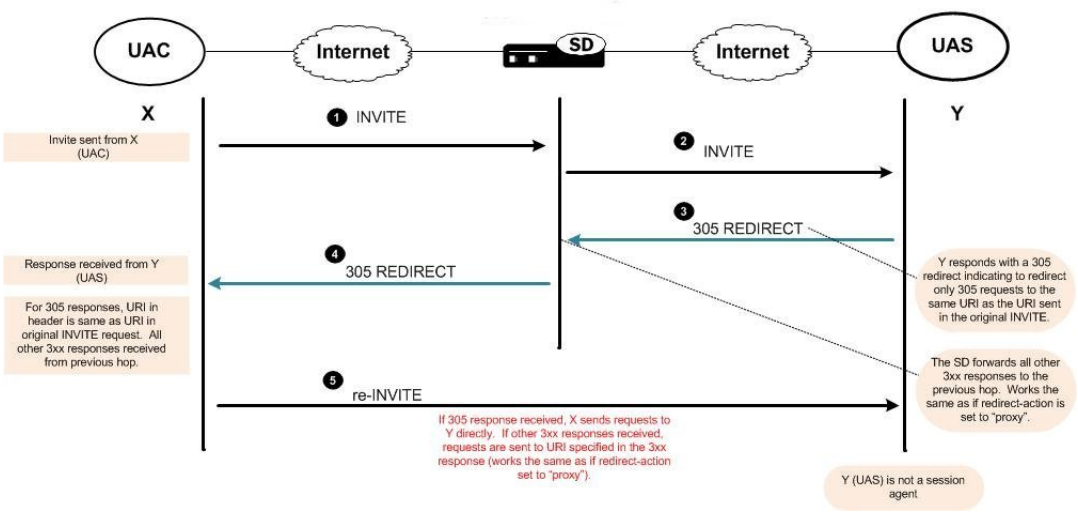


Redirect-Action Set to Recurse-305-Only

The following occurs if you set the **redirect-action** parameter to **recurse-305-only** on the SBC:

1. X (UAC) sends an INVITE to the SBC.
2. The SBC forwards the INVITE to Y (UAS).
3. Y sends a 305 REDIRECT response to the SBC with the same URI as the URI sent in the original request.
4. The SBC forwards the 305 REDIRECT response to X (UAC).
5. If 305 response received, X sends requests to Y directly. If other 3xx responses received, requests are sent to URI specified in the 3xx response (works the same as if redirect-action set to proxy).

   The following illustration shows an example of a dialog between X, Y, and the SBC during a redirect-action session set to recurse-305-only.



Redirect Configuration for SIP Interface

You can configure the Oracle Communications Session Border Controller to redirect requests from a UAC to a UAS using the URI in 305 responses only. You can use the ACLI at the paths **session-router**, **sip-interface** or **session-router**, **session-agent**.

To configure the redirect-action feature on the SIP interface on the Oracle Communications Session Border Controller:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type **sip-interface** and press Enter to access the SIP interface-related configurations. The system prompt changes to let you know that you can begin configuring individual parameters for this object.

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

4. **Enter redirect-action followed by the following value:**
   - recurse-305-only

```
ORACLE(sip-interface)# redirect-action recurse-305-only
```

When the Oracle Communications Session Border Controller receives a 305 SIP redirect response (Use Proxy) on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 305 response. All 3xx responses other than 305 responses are sent back to the previous hop.

To disable this feature, enter **redirect-action** and press Enter without entering a value.

```
ORACLE(sip-interface)# redirect-action
```

Redirect Configuration for Session Agent

To configure the redirect-action feature for a session agent on the Oracle Communications Session Border Controller:

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 configurations.

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

3. Type **session-agent** and press Enter to access the SIP session agent-related configurations. The system prompt changes to let you know that you can begin configuring individual parameters for this object.

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

4. **Enter redirect-action followed by the following value:**
   - recurse-305-only

```
ACMEPACKET(session-agent)# redirect-action recurse-305-only
```

When the Oracle Communications Session Border Controller receives a 305 SIP redirect response (Use Proxy) on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 305 response. All 3xx responses other than 305 responses are sent back to the previous hop.

To disable this feature, enter redirect-action and press Enter without entering a value.

```
ACMEPACKET(sip-interface)# redirect-action
```

Embedded Routes in Redirect Responses

When the Oracle Communications Session Border Controller recurses as the result of a redirect (3xx) response, the server might need to specify one or more intermediate hops. These hops are reflected in the Contact header for the 3xx response using embedded route headers and look like this:

```
Contact: <sip:touser@server.example.com?Route=%3Cproxy.example.com%Blr%3E>
```

The Contact header shows that the request should be sent to server.example.com using proxy.example.com.

You can configure your Oracle Communications Session Border Controller to specify that embedded headers in 3xx Contact headers are to be included in new requests such that they are tied to a session agent representing the new target (server.example.com). This behavior requires you to set the **request-uri-headers** parameter.

However, you can also use the **use-redirect-route** in global SIP configuration's options parameter so that the embedded Route header is used as the next hop to receive the new request.

When you configure this new option, the Oracle Communications Session Border Controller constructs a new request using the redirect Contact, and the SIP URI from the Contact becomes the Request-URI. Then, the system inserts the embedded routes as Route headers in the, using the same order in which they appeared in the redirect Contact. Afterward, the Oracle Communications Session Border Controller determines the next hop in the same way it does with any other request. If the first route is a loose route (i.e., it has the lr URI parameter), then the Oracle Communications Session Border Controller sends a request to host indicated in the first route. Otherwise, strict routing applies, and the Oracle Communications Session Border Controller sends the request to the host indicated in the Request-URI.

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name.

```
ORACLE(sip-config)# options use-redirect-route
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Selecting SDP within Multi-Dialog Call Scenarios

By default, the Oracle Communications Session Border Controller saves SDP presented in a series of early dialogs using To tags to differentiate between dialogs. If the session continues with a 200OK that does not include SDP, the Oracle Communications Session Border Controller refers to the To tag to identify the dialog from which the Oracle Communications Session Border Controller selects the SDP for the media flow. This complies with 3GPP TS 24.628, TS 24.182 and RFC 5009 behavior for sessions supporting early media.

Consider a SIP dialog with early media that proceeds by establishing call scenarios in which the final 200 OK does not include any SDP. This messaging may include multiple 183 (Session Progress) messages with SDP that differ from each other and include different To tags, establishing multiple early dialogs. In these scenarios, the Oracle Communications Session Border Controller uses the saved SDP from the dialog that matches the dialog indicated in the 200 OK's To tag to anchor the media. If the 200 OK includes SDP, the Oracle Communications Session Border Controller uses that SDP to anchor the media.

> **Note:**
>
> The user can disable this behavior, for example, to use the functional behavior in Oracle Communications Session Border Controller version S-CZ7.2.0 and below, which is to use the last SDP seen as the source for SDP. Deployments that rely on this behavior must revert to it using the **sip-config**'s **dont-save-early-dialog-sdp** option parameter.

```
ORACLE(sip-config)# options +dont-save-early-dialog-sdp
```

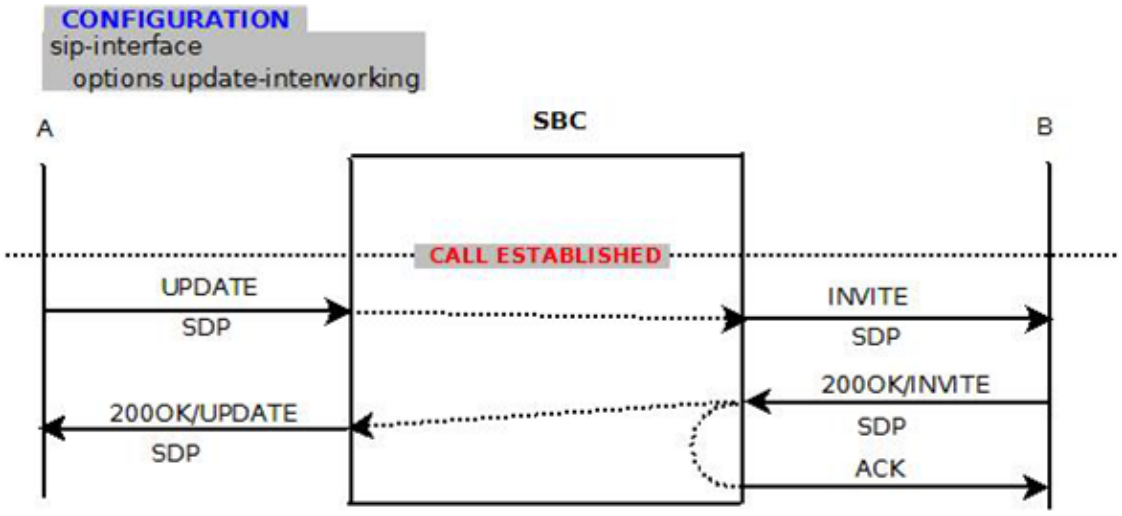
UPDATE Interworking

The Oracle Communications Session Border Controller can be configured to convert UPDATE methods (with or without SDP) to INVITE methods inside a dialog that has already been established. SDP is inserted when the UPDATE message doesn't have it. The method is modified from UPDATE to INVITE for the duration of an UPDATE based transaction and the SBC creates an ACK message to acknowledge the INVITE response.

You must specify the new **sip-interface** option **update-interworking** on the ingress SIP interface to enable the UPDATE Interworking feature. The three possible scenarios are described as follows:

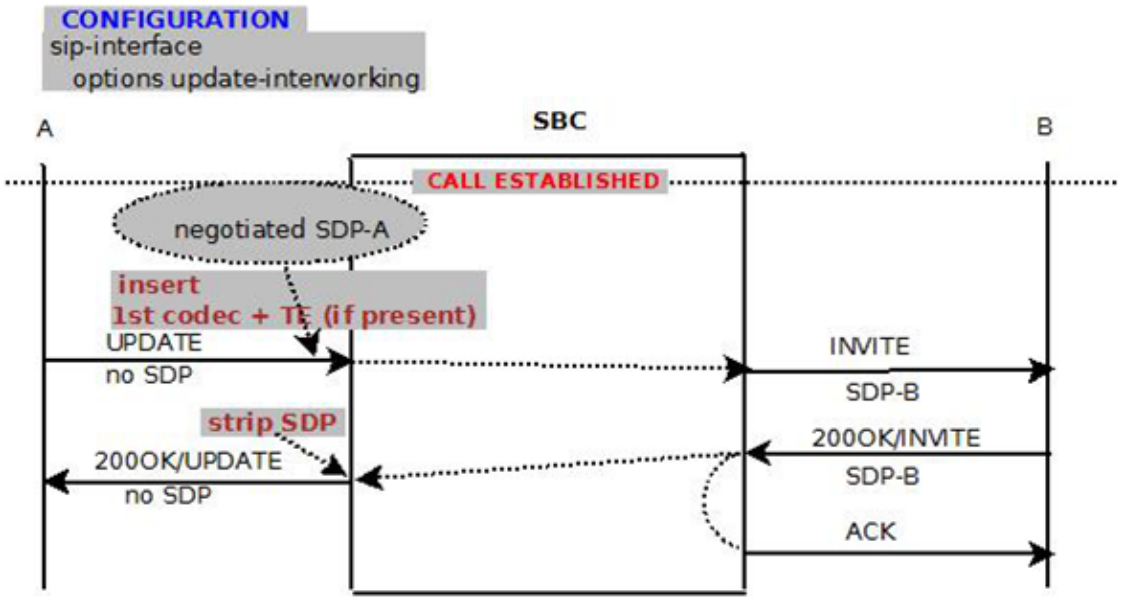
Successful UPDATE with SDP

In this example the original UPDATE contains SDP, and the INVITE is sent with SDP.



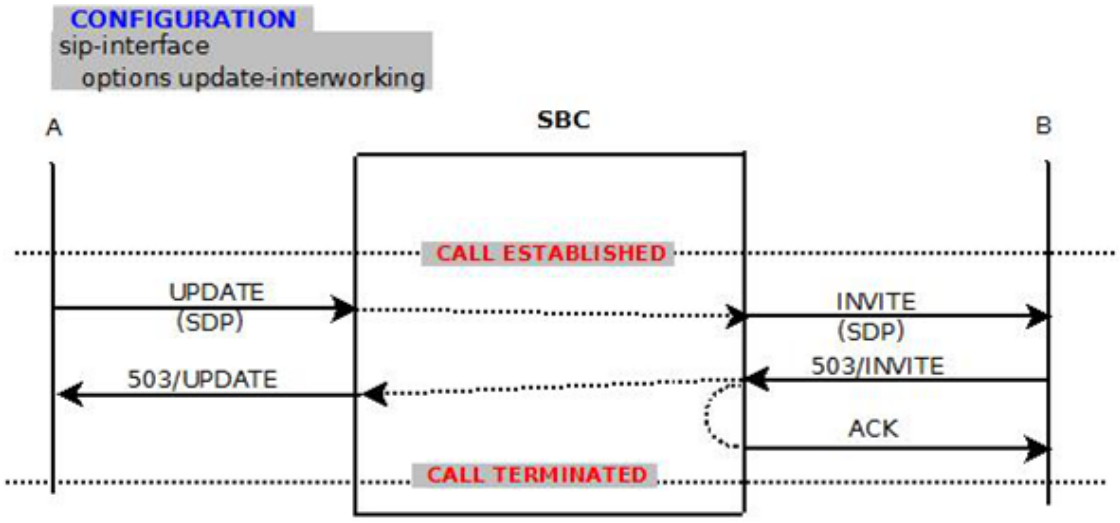
Successful UPDATE without SDP

In this example the original UPDATE does not contain SDP, SDP is inserted and the INVITE is sent with SDP. SDP is stripped from 200OK/UPDATE on the “A” (ingress) side.



UPDATE Failure

In this example the original UPDATE contains SDP, and the INVITE is sent with SDP.



UPDATE Interworking Configuration

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

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

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **options** — Set this parameter by typing **options**, a Space, the option name **update-interworking** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-interface)# options +update-interworking
```

   If you type options and then the option value without the plus sign, you will overwrite any previously configured options. In order to append the new option to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

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

SIP PRACK Interworking

When you configure your Oracle Communications Session Border Controller with PRACK interworking for SIP, you enable it to interwork between endpoints that support RFC 3262, *Reliability of Provisional Responses in the Session Initiation Protocol*, and those that do not.

As its title indicates, RFC 3262 defines a reliable provisional response extension for SIP INVITEs, which is the 100rel extension tag. While some endpoints do not support the RFC, other SIP implementations require compliance with it. A session setup between two such endpoints fails. However, you can configure your Oracle Communications Session Border Controller to supply the provisional response on behalf of endpoints that do not support it—and thereby enable sessions between those endpoints and the ones requiring RFC 3262 compliance.

You need to configure PRACK interworking for a SIP interface associated with the endpoints that need RFC 3262 support. To enabled the feature, you set the **100rel-interworking** option. The Oracle Communications Session Border Controller applies PRACK interworking for either the UAC or the UAS. TheOracle Communications Session Border Controller checks to see whether or not it needs to apply PRACK interworking when an INVITE arrives at the ingress or egress SIP interface with the option enabled. First, it checks the Require header for the 100rel tag; if not found there, it checks the Supported header.

Since there is a slight difference in the application of this feature between the UAC and UAS, this section explains both.

> **Note:**
>
> If SDP is included in a PRACK request sent to a SIP interface where PRACK interworking is enabled, it will not be responded to, nor will any SDP be included in the locally-generated 200 OK to that PRACK.

UAC-Side PRACK Interworking

The Oracle Communications Session Border Controller applies PRACK interworking on the UAC side when:

- An incoming SIP INVITE contains the 100rel tag in a Require header
- The ingress SIP interface is enabled with the **100rel-interworking** option
- The UAS fails to send reliable provisional responses

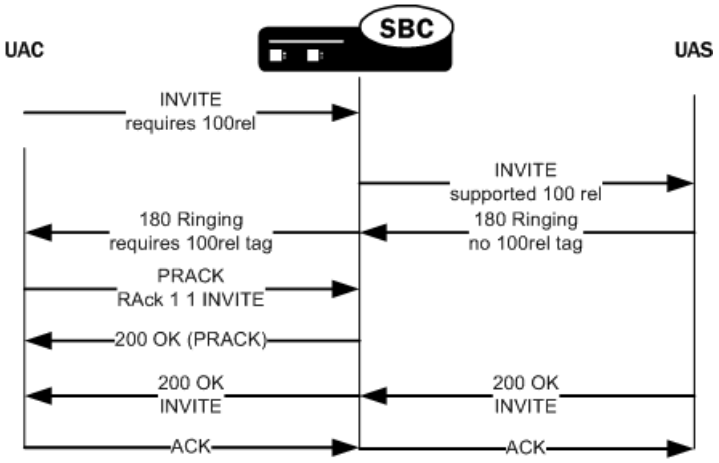
When it is to forward a non-reliable response to a UAC that requires RFC 3262 support, the Oracle Communications Session Border Controller converts the non-reliable response to a reliable one by adding the 100rel tag to the Require header and adding an Rseq header to the response. Further, the Oracle Communications Session Border Controller adds a Require header (complete with the100rel tag) if there is not one already in the response, and then also adds Rseq header.

Note that the Oracle Communications Session Border Controller sets the value of the Rseq header as 1 for the first provisional response, and then increments it by 1 for each subsequent provisional response. It also adds the PRACK method to the Allow header when that header appears.

The Oracle Communications Session Border Controller retransmits the converted reliable provisional response in accordance with RFC 3262, until it receives a PRACK request. For the initial timeout for retransmission, the Oracle Communications Session Border Controller uses the value you set in the **init-timer** parameter in the global SIP configuration. It stops retransmitting when either it receives a transmission, or when the ingress SIP interface's trans-expire timer elapses.

If it never receives a PRACK, the Oracle Communications Session Border Controller does not generate an error response to the INVITE, relying instead on the downstream UAS to produce a final response.

The call flow for this application looks like this:



UAS-Side PRACK Interworking

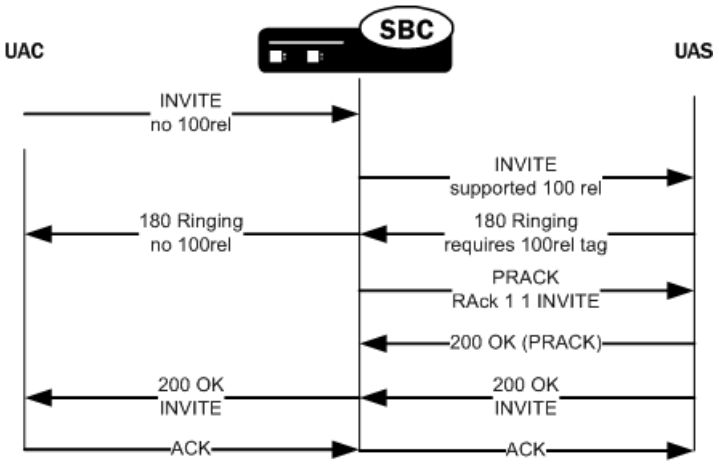
The Oracle Communications Session Border Controller applies PRACK interworking on the UAS side when:

- An incoming SIP INVITE does not contain the 100rel tag in a Require or Supported header
- The egress SIP interface is enabled with the **100rel-interworking** option
- The UAS does send reliable provisional responses

When the UAC does not support RFC 3262, the Oracle Communications Session Border Controller generates a PRACK request to acknowledge the response. It also converts the response to non-reliable by removing the 100 rel tag from the Require header and removing the RSeq header from the response.

In the case of the UAS, the Oracle Communications Session Border Controller matches the PRACK to a converted reliable provisional response using the PRACK's RAck header. If it finds a matching response, the Oracle Communications Session Border Controller generates a 200 OK to the PRACK. And if it finds no match, then it generates a 481 Call Leg/Transaction Does Not Exist response. The Oracle Communications Session Border Controller generates a 400 Bad Request response if either the RAck is not in the PRACK request or it is not formatted properly.

The call flow for this application looks like this:



PRACK Interworking Configuration

You enable PRACK interworking for ingress and egress SIP interfaces. Be sure you know on what side, ingress or egress, you need this feature applied.

To configure PRACK interworking for a SIP interface:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-interface** and press Enter. If you are editing an existing configuration, select the one on which you want to enable this feature.

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

4. **options**—Set the options parameter by typing **options**, a Space, the option name **100rel-interworking** with a plus sign in front of it, and then press Enter.

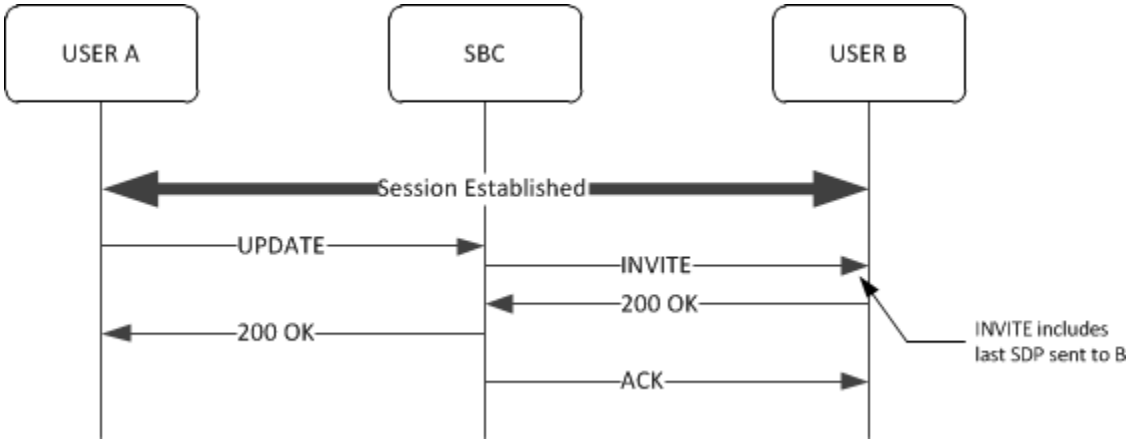
   ```
   ORACLE(sip-interface)# options +100rel-interworking
   ```

   If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to this** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

5. Save and activate your configuration.

After Dialog Establishment (INVITE transaction terminated)

Inside a dialog that has already been established, we should offer a way to convert UPDATE methods (with or without SDP) to INVITE methods. During the duration of this UPDATE based transaction, the method has to be modified from UPDATE to INVITE in all the appropriate places.



In the case that the original UPDATE does not contain SDP, the INVITE will include the last SDP sent to Peer B.

During dialog establishment (INVITE transaction not yet terminated)

There are six different use cases here depending on the following factors:

- Caller/called requirements of reliable responses
- Delayed/early offer
- SDP insertion in delayed offer or not

Use Case	100rel Required by	Delayed offer	SDP injection
Use Case 1	Callee	No	N/A
Use Case 2	Callee	Yes	Yes
Use Case 3	Callee	Yes	No
Use Case 4	Caller	No	N/A
Use Case 5	Caller	Yes	Yes
Use Case 6	Caller	Yes	No

PRACK IWF on ingress side:

- When the 100rel-interworking (or equivalent CLI) option is activated on the ingress realm, if an INVITE comes with a 100rel tag in a Require header, the INVITE will be forwarded with the tag only in the Supported header.
- If the response is that the UAS do not support 100rel, the function will be triggered. If the response comes with a supported 100rel, the function will not be triggered.

PRACK IWF on egress side:

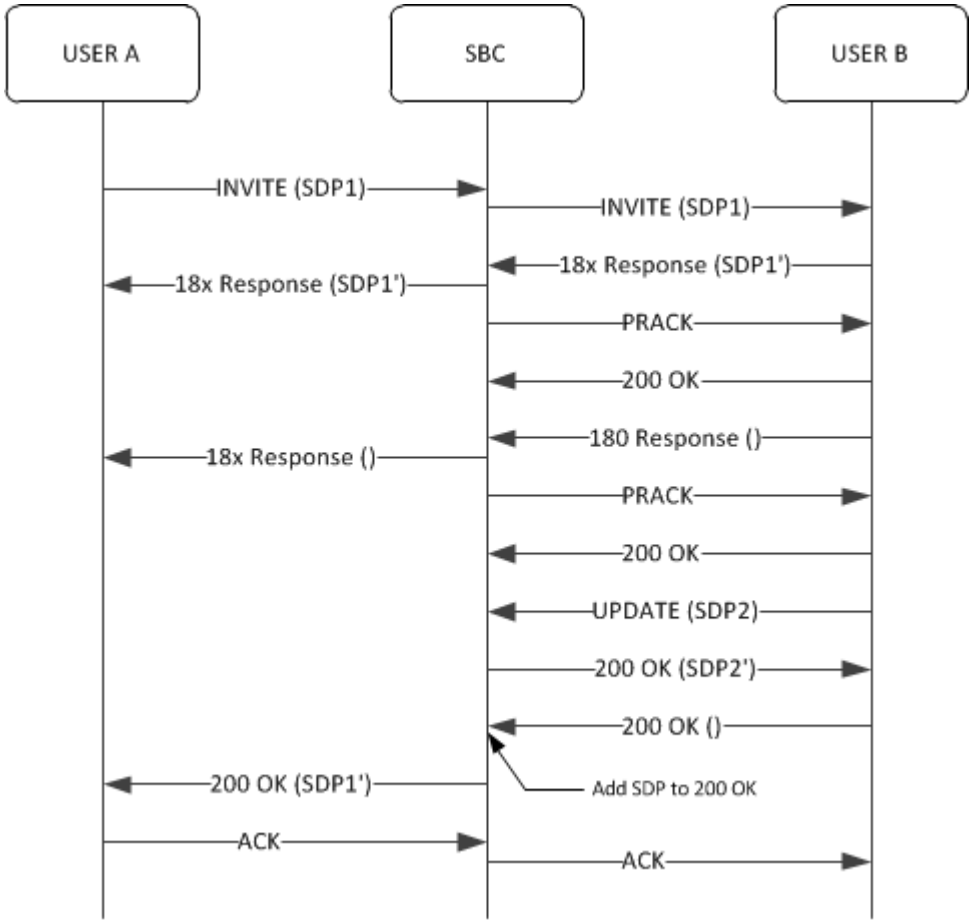
When the 100rel-interworking option is activated on the egress realm:

- If an INVITE comes with a 100rel tag in a Require header, the function will not be triggered.

- If an INVITE comes with no 100rel tag, it will be included in the Supported header.
  - If the response includes 100rel requirements, the function will be triggered

Use Case 1

In this use case the caller leg does not support reliable responses and the called leg requires them. The initial INVITE comes with an SDP offer.



Actions:

- Upon reception of a response with SDP (it can only happen once)
  - Forward it to A side
  - Send PRACK and wait for 200OK
  - Keep the SDP of the first provisional response (SDP1').
    NOTE: The SDP answer must come in a reliable response. The 200OK of a PRACK is not reliable
- Upon reception of a response with no SDP
  - Forward it to A side
  - Send PRACK and wait for 200OK
- Upon reception of an UPDATE with SDP from B after the SDP answer,
  - Answer locally the UPDATE in the called side. The SDP of the 200OK will contain:
    * The first codec of the codec list present in the new offer that complies with the configured codec-policies.

    * The tel-events and image codecs from the new offer.

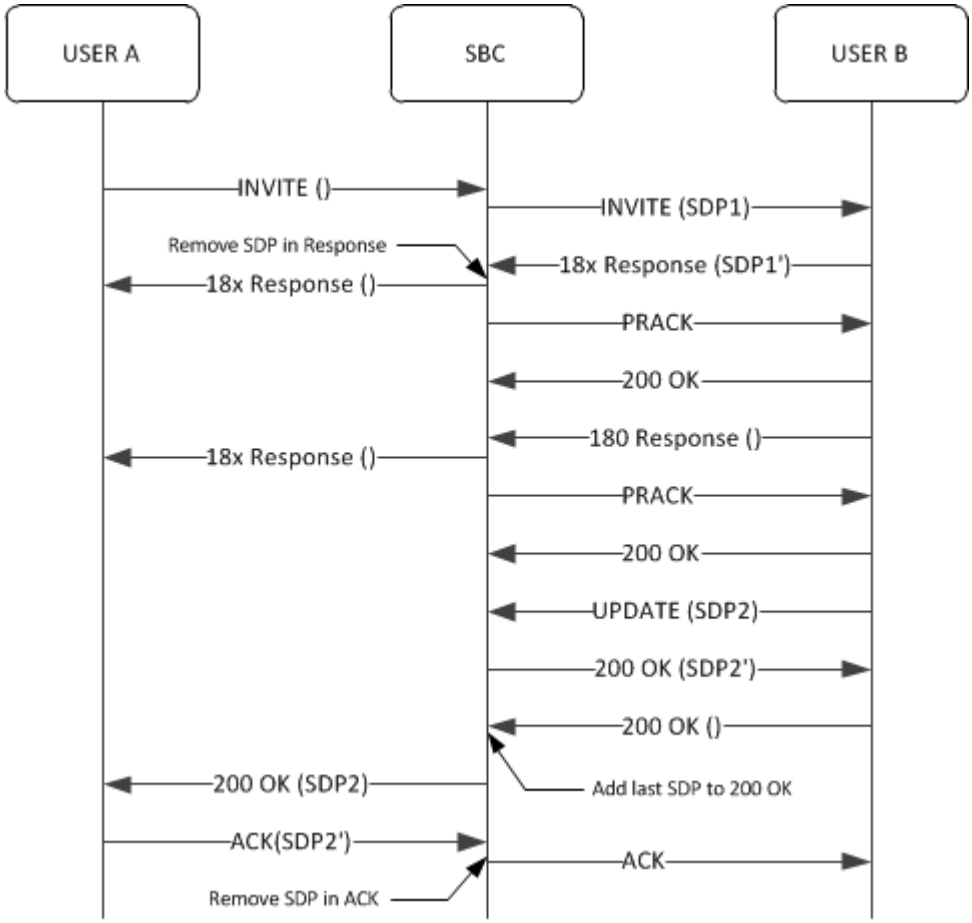
- Upon reception of an UPDATE with SDP from B before the SDP answer, reject it with 488 (INVITE is still ongoing)
- Upon reception of any additional response with no SDP
  - Forward it to A side
  - Send PRACK and wait for 200OK
- 200OK for INVITE should come with no SDP. We'll include the same SDP as in the previous message in this side (SDP1').
- Depending of the codecs offered and answered in both sides transcoding may be triggered.

Use Case 2 (Not applicable to Session Router)

In this use case the caller leg does not support reliable responses and the called leg requires them.

The initial INVITE comes without SDP offer, and the SBC is adding SDP in the outbound INVITE

This use case is not applicable in a Session Router installation since we don't have a way to indicate the proper IP address in the generated SDP.



Actions:

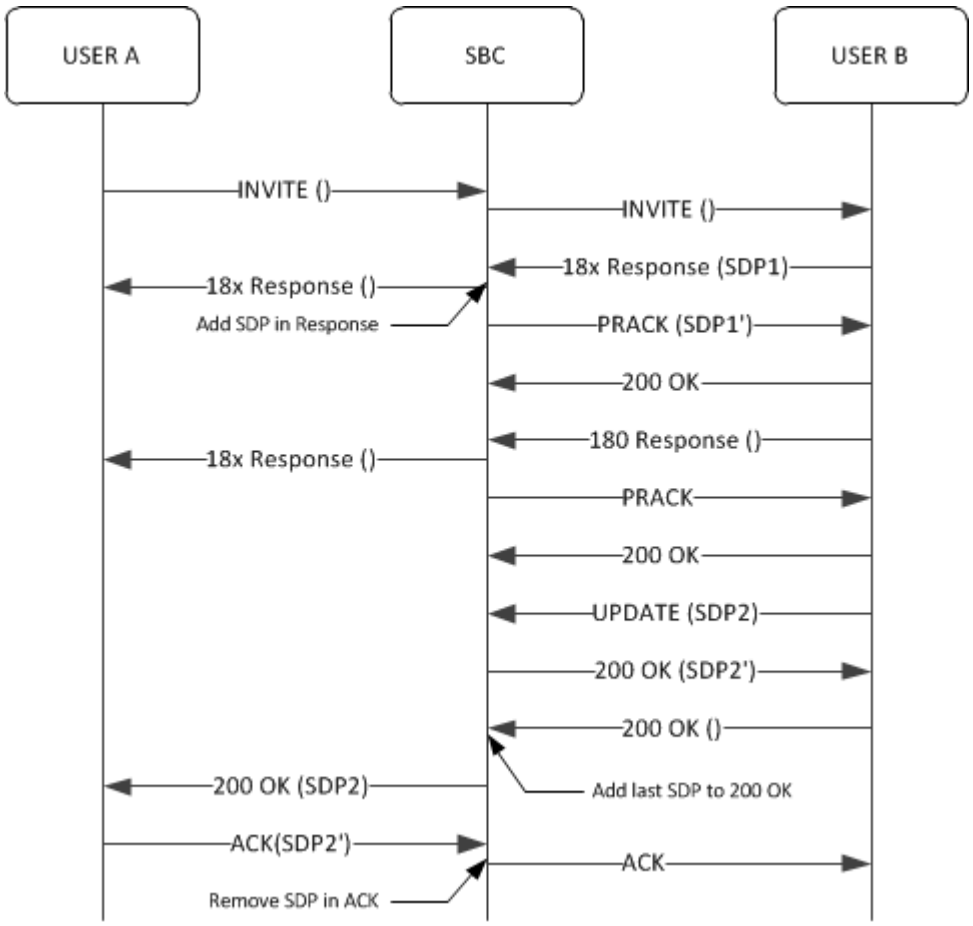
- Upon reception of offer-less INVITE, and add-SDP is used:

  - Add SDP in the egress INVITE (as today).
- Upon reception of a response with no SDP
  - Forward it to A side
  - Send PRACK and wait for 200OK
- Upon reception of a response with SDP (it can only happen once)
  - Remove SDP and forward it to A side
  - Send PRACK and wait for 200OK
  - Keep the SDP.
    NOTE: The SDP answer must come in a reliable response. The 200OK of a PRACK is not reliable.
- Upon reception of an UPDATE with SDP from B after the SDP answer,
  - Keep the SDP (don't need the previous kept SDP)
  - Answer locally the UPDATE in the called side. The SDP of the 200OK will contain:
    * The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    * The tel-events and image codecs from the new offer.
- Upon reception of an UPDATE with SDP from B before the SDP answer, reject it with 488 (INVITE is still ongoing)
- Upon reception of any additional response with no SDP
  - Forward it to A side
  - Send PRACK and wait for 200OK
- 200OK for INVITE should come with no SDP. We'll include the last SDP received from the B side..
- The ACK from the calling side will come with SDP. Remove it before relaying the ACK to the other side.
- Depending of the codecs offered and answered in both sides transcoding may be triggered.

Use Case 3 (Not applicable to Session Router)

In this use case the caller leg does not support reliable responses and the called leg requires them. The initial INVITE comes without SDP offer, and the SBC is not adding SDP in the outbound INVITE

This use case is not applicable in a Session Router installation since we don't have a way to indicate the proper IP address in the generated SDP.



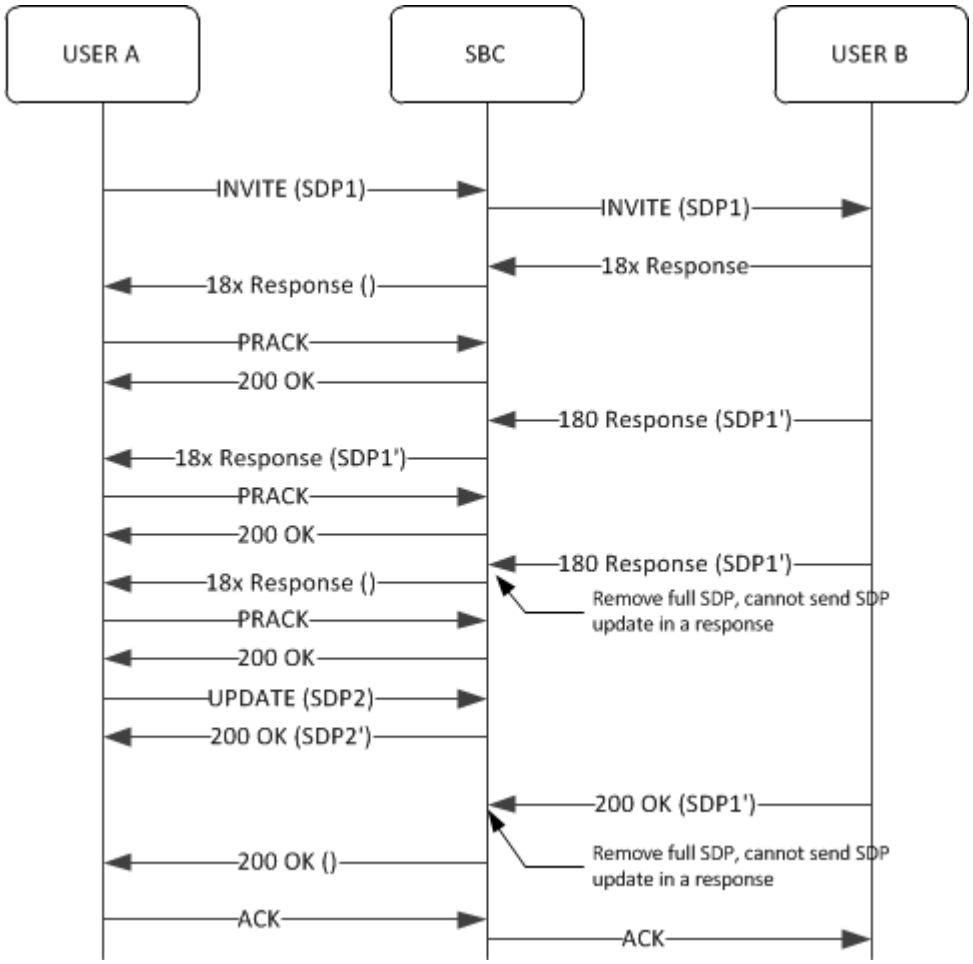
Actions:

- Upon reception of a response with no SDP (Illegal, but should we care?)
  - Forward it to A side
  - Send PRACK and wait for 200OK
- Upon reception of a response with SDP (it can only happen once)
  - Keep the SDP.
  - Remove SDP and forward it to A side
  - Send PRACK with SDP. Include a SDP answer to the PRACK that acknowledges the provisional response with SDP offer. This SDP will contain:
    - The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    - The tel-events and image codecs from the new offer.
  - Wait for 200OK for PRACK
- Upon reception of an UPDATE with SDP from B after the SDP answer,
  - Keep the SDP (don't need the previous kept SDP)
  - Answer locally the UPDATE in the called side. The SDP of the 200OK will contain:
    - The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    - The tel-events and image codecs from the new offer.

- Upon reception of an UPDATE with SDP from B before the SDP answer, reject it with 488 (INVITE is still ongoing). This is completely illegal, remove it if required.
- Upon reception of any additional response with no SDP:
  - Forward it to A side
  - Send PRACK and wait for 200OK
- 200OK for INVITE should come with no SDP. We'll include the last SDP received from the B side.
- The ACK from the calling side will come with SDP. Remove it before relaying the ACK to the other side.
- Depending of the codecs offered and answered in both sides transcoding may be triggered

Use Case 4

In this use case the caller leg does require reliable responses and the called side does not support them. The initial INVITE comes with SDP offer.



Actions:

- Upon reception of a response with no SDP:
  - Forward it to A side
  - Answer locally the PRACK with 200OK
- Upon reception of a response with SDP (it can only happen once):

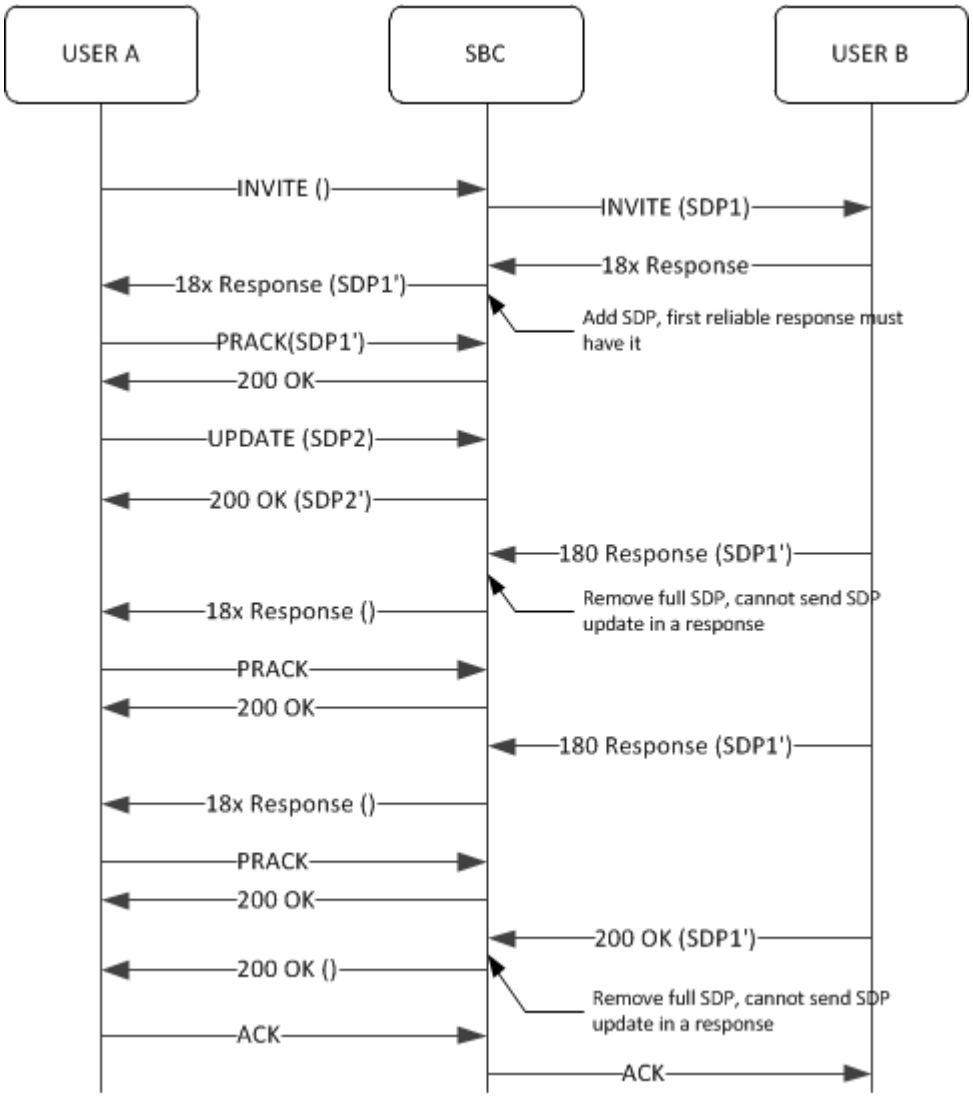
  - Keep the SDP of the first provisional response (SDP1').
  - Forward it to A side
  - Wait for PRACK.
- If PRACK comes with SDP, handle it as an UPDATE (see below)
- If PRACK comes with no SDP, just answer it as today.
- Upon reception of a new response with SDP (new response with SDP again):
  - Remove the SDP
  - Forward it to A side
  - Wait for PRACK.
    * If PRACK comes with SDP, handle it as an UPDATE (see below)
    * If PRACK comes with no SDP, just answer it as today.
- Upon reception of an UPDATE with SDP from A after the SDP answer:
  - Answer locally the UPDATE. The SDP of the 200OK will contain:
    * The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    * The tel-events and image codecs from the new offer.
- Upon reception of an UPDATE with SDP from A before the SDP answer, reject it with 488 (INVITE is still ongoing)
- Upon reception of any additional response with no SDP:
  - Forward it to A side
  - Answer locally the PRACK with 200OK
- 200OK for INVITE should come with SDP:
  - Remove SDP
  - Forward 200OK to A side
- Depending of the codecs offered and answered in both sides transcoding may be triggered.

Use Case 5 (Not applicable to Session Router)

In this use case the caller leg does require reliable responses and the called leg does not support them. The initial INVITE comes without SDP offer, and we have the SBC configured to add the SDP to the initial INVITE.

Also, the first provisional response comes with no SDP, so the SBC will add SDP to the first response.

This use case is not applicable in a Session Router installation since we don't have a way to indicate the proper IP address in the generated SDP.

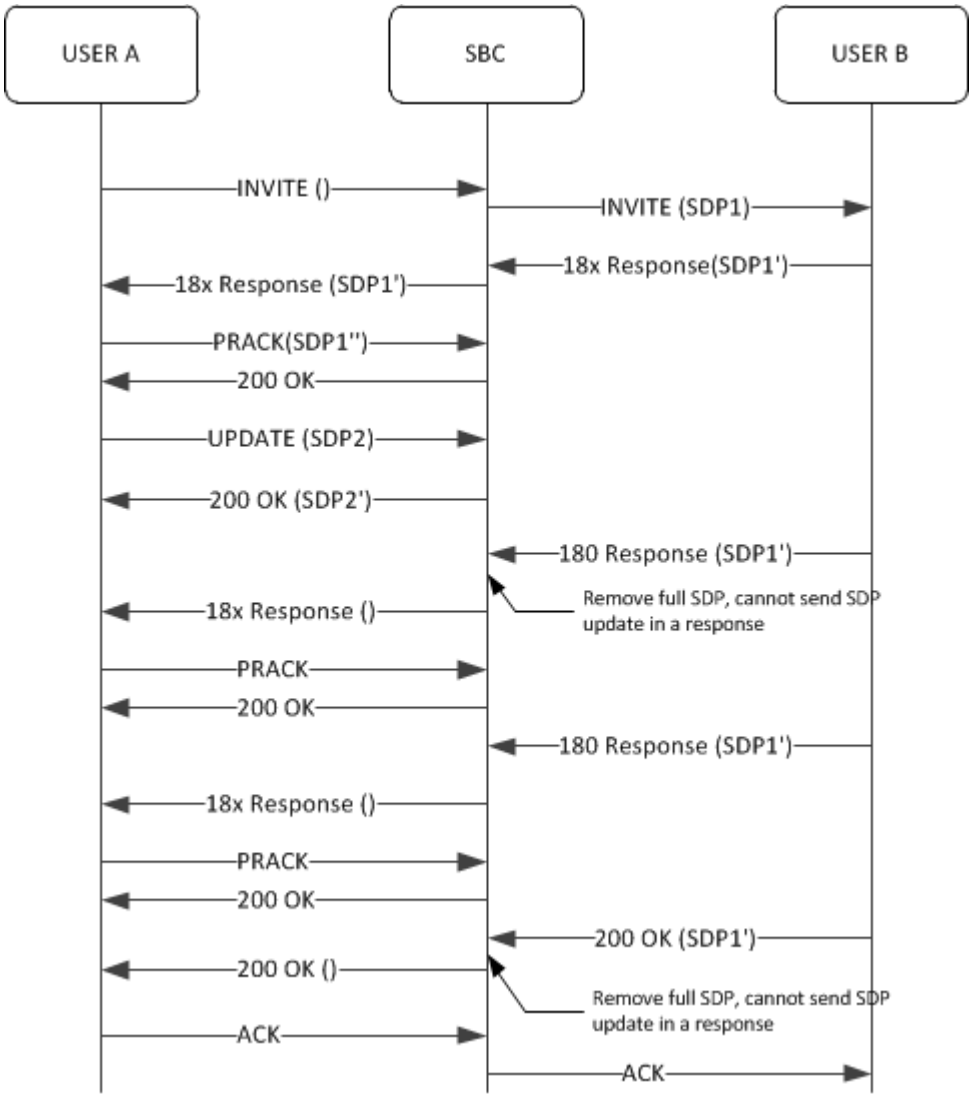


Actions:

- Add SDP in the egress INVITE.
- The first provisional response comes without SDP, so the SBC will add a predefined (media-profile) to the SDP to the response.
  - Forward it to A side
  - Wait for PRACK.
    - Will come with SDP1''
    - And answer it
- Upon reception of a new response with SDP:
  - Remove the SDP
  - Forward it to A side
  - Wait for PRACK.
    - If PRACK comes with SDP, handle it as an UPDATE (see below)
    - If PRACK comes with no SDP, just answer it as today.
- Upon reception of an UPDATE with SDP from A after the SDP answer:

  - Answer locally the UPDATE. The SDP of the 200OK will contain:
    * The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    * The tel-events and image codecs from the new offer.
- Upon reception of an UPDATE with SDP from A before the SDP answer, reject it with 488 (INVITE is still ongoing)
- Upon reception of any additional response with no SDP:
  - Forward it to A side
  - Wait for PRACK:
    * If PRACK comes with SDP, handle it as an UPDATE (see previous step)
    * If PRACK comes with no SDP, just answer it as today.
- 200OK for INVITE should come with SDP:
  - Remove SDP
  - Forward 200OK to A side
- Depending of the codecs offered and answered in both sides transcoding may be triggered

As an alternative to use-case 5 we have the case where the first provisional response already comes with SDP:



Actions:

- Add SDP in the egress INVITE.
- The first provisional response comes with SDP:
  - Forward it to A side
  - Wait for PRACK.
    - Will come with SDP1''
    - And answer it
- Upon reception of a new response with SDP (new response with SDP again):
  - Remove the SDP
  - Forward it to A side
  - Wait for PRACK.
    - If PRACK comes with SDP, handle it as an UPDATE (see below)
    - If PRACK comes with no SDP, just answer it as today.
    -

- Upon reception of an UPDATE with SDP from A after the SDP answer:
  - Answer locally the UPDATE. The SDP of the 200OK will contain:
    - The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    - The tel-events and image codecs from the new offer.
- Upon reception of an UPDATE with SDP from A before the SDP answer, reject it with 488 (INVITE is still ongoing)
- Upon reception of any additional response with no SDP:
  - Forward it to A side
  - Wait for PRACK.
    - If PRACK comes with SDP, handle it as an UPDATE (see previous step)
    - If PRACK comes with no SDP, just answer it as today.
- 200OK for INVITE should come with SDP:
  - Remove SDP
  - Forward 200OK to A side
- Depending of the codecs offered and answered in both sides transcoding may be triggered.

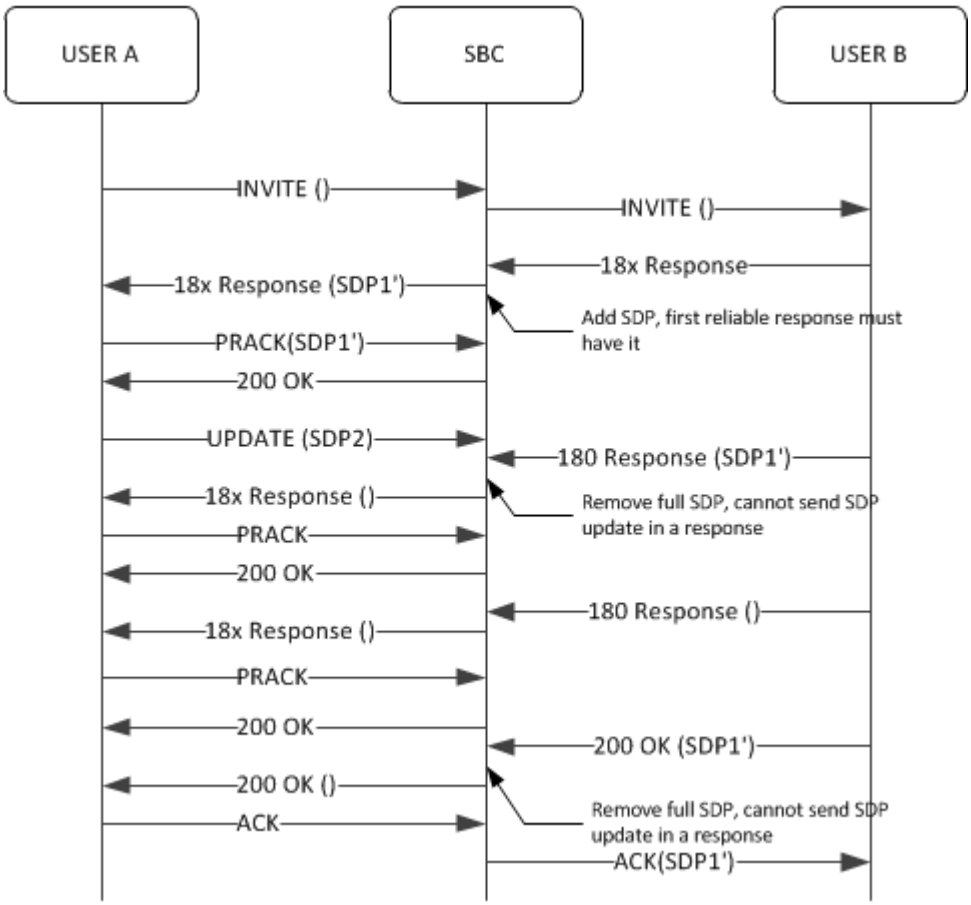
Use Case 6 (not applicable to Session Router)

In this use case the caller leg does require reliable responses and the called leg does not support them. The initial INVITE comes without SDP offer, and the SBC is not configured to add the SDP to the initial INVITE.

This use case is not applicable in a Session Router installation since we don't have a way to indicate the proper IP address in the generated SDP.

From the IETF point of view, the first SDP offer should come in the first reliable response from the UAS, in this case the 200OK. Any other case would be illegal. However, in order to protect for possible use cases, and leveraging the previous use case, we'll offer also this one:

** LRBT is currently planned for a later release of SBC and so any interactions with LRBT will not be implemented as part of this PRACK IWF feature in 7.3 M2



Actions:

- The first provisional response comes without SDP:
  - o Add a predefined (media-profile) to the SDP in the response.
  - o Forward it to A side
  - o Wait for PRACK.
    - Will come with SDP1''
    - And answer it
- Upon reception of a new response with SDP:
  - Remove the SDP
  - Forward it to A side
  - Wait for PRACK.
    - If PRACK comes with SDP, handle it as an UPDATE (see below)
    - If PRACK comes with no SDP, just answer it as today.
- Upon reception of an UPDATE with SDP from A after the SDP answer:
  - Answer locally the UPDATE. The SDP of the 200OK will contain:
    - The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    - The tel-events and image codecs from the new offer.

- Upon reception of an UPDATE with SDP from A before the SDP answer, reject it with 488 (INVITE is still ongoing)
- Upon reception of any additional response with no SDP:
  - Forward it to A side
  - Wait for PRACK.
    - If PRACK comes with SDP, handle it as an UPDATE (see previous step)
    - If PRACK comes with no SDP, just answer it as today.
- 200OK for INVITE should come with SDP:
  - Remove SDP
  - Forward 200OK to A side
- In the ACK insert SDP. The SDP will contain:
  - The first codec of the codec list present in the new offer (SDP1') that complies with the configured codec-policies.
  - The tel-events and image codecs from the new offer (SDP1').
- Depending of the codecs offered and answered in both sides transcoding may be triggered.

Alternative case where the first reliable response comes with SDP:

Actions:

- The first provisional response comes with SDP:
  - Forward it to A side
  - Wait for PRACK.
    - Will come with SDP1''
    - And answer it
- Upon reception of a new response with SDP (new response with SDP again):
  - Remove the SDP
  - Forward it to A side
  - Wait for PRACK.
    - If PRACK comes with SDP, handle it as an UPDATE (see below)
    - If PRACK comes with no SDP, just answer it as today.
- Upon reception of an UPDATE with SDP from A after the SDP answer:
  - Answer locally the UPDATE. The SDP of the 200OK will contain:
    - The first codec of the codec list present in the new offer that complies with the configured codec-policies.
    - The tel-events and image codecs from the new offer.
- Upon reception of an UPDATE with SDP from A before the SDP answer, reject it with 488 (INVITE is still ongoing)
- Upon reception of any additional response with no SDP:
  - Forward it to A side
  - Wait for PRACK.

    * If PRACK comes with SDP, handle it as an UPDATE (see previous step)
    * If PRACK comes with no SDP, just answer it as today.
- 200OK for INVITE should come with SDP:
  - Remove SDP
  - Forward 200OK to A side
- In the ACK insert SDP. The SDP will contain:
  - The first codec of the codec list present in the new offer (SDP1') that complies with the configured codec-policies.
  - The tel-events and image codecs from the new offer (SDP1').
- Depending of the codecs offered and answered in both sides transcoding may be triggered.

Global SIP Timers

This section explains how to configure SIP retransmission and expiration timers.

> **Note:**
>
> you can also set timers and counters per SIP interface.

Overview

SIP timers define the transaction expiration timers, retransmission intervals when UDP is used as a transport, and the lifetime of dynamic TCP connections. The retransmission and expiration timers correspond to the timers defined in RFC 3261.

- init timer: is the initial request retransmission interval. It corresponds to Timer T1 in RFC 3261.
  This timer is used when sending requests over UDP. If the response is not received within this interval, the request is retransmitted. The retransmission interval is doubled after each retransmission.
- max timer: is the maximum retransmission interval for non-INVITE requests. It corresponds to Timer T2 in RFC 3261.
  The retransmission interval is doubled after each retransmission. If the resulting retransmission interval exceeds the max timer, it is set to the max timer value.
- trans expire: is the transaction expiration timer. This value is used for timers B, D, F, H and J as defined in RFC 3261.
- invite expire: defines the transaction expiration time for an INVITE transaction after a provisional response has been received. This corresponds to timer C in RFC 3261.
  If a final response is not received within this time, the INVITE is cancelled. In accordance with RFC 3261, the timer is reset to the invite expire value when any additional provisional responses are received.
- Inactive dynamic conn timer defines the idle time of a dynamic TCP connection before the connection is torn down. Idle is defined as not transporting any traffic. There is no timer in RFC 3261 corresponding to this function.

Timers Configuration

To configure timers:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

4. **init-timer**—Enter the initial timeout value in milliseconds for a response to an INVITE request, and it applies to any SIP request in UDP. In RFC 3261, this value is also referred to as TIMER_T1. The default is **500**. The valid range is:
   - Minimum—0
   - Maximum—4294967295
5. **max-timer**—Enter the maximum transmission timeout (T2) for SIP in milliseconds.

   When sending SIP over UDP, a re-transmission timer is used. If the timer expires and the message is re-transmitted, the re-transmission timer is then set to twice the previous value (but will not exceed the maximum timer value). Using the default values of 500 milliseconds and 4000 milliseconds, the re-transmission timer is 0.5, then 1, 2, and finally 4. The incrementing continues until the transmission expire timer activates. The default is **4000**. The valid range is:
   - Minimum—0
   - Maximum—4294967295
6. **trans-expire**—Enter the transaction expire timeout value (Timer B) in seconds to set the time for SIP transactions to live. The same value is used for Timers D, F, H and J. The default is **32**. The valid range is:
   - Minimum—0
   - Maximum—2147473
7. **invite-expire**—Enter the invite expire timeout value (Timer C) in seconds to indicate the time for SIP client transaction will live after receiving a provisional response. The default is **180**. The valid range is:
   - Minimum—0
   - Maximum—2147473
8. **inactive-dynamic-conn**—Enter the inactive dynamic connection value in seconds to set the time limit for inactive dynamic connections.

   If the connection between the SIP proxy and a session agent is dynamic (for example, through dTCP), and the connection has been idle for the amount of time specified here, the

SIP proxy breaks the connection. Idle is defined as not transporting any traffic. The default value is **32**. The valid range is:

- Minimum—0
- Maximum—4294967295

> **Note:**
>
> Setting this parameter to 0 disables this parameter.

The following example shows SIP config timer values for a peering network. Some parameters are omitted for brevity.

```
sip-config
        state                           enabled
        operation-mode                  dialog
dialog-transparency        disabled
home-realm-id                   acme
        egress-realm-id
        nat-mode                        Public
        registrar-domain
        registrar-host
        registrar-port                  0
        init-timer                      500
        max-timer                       4000
        trans-expire                    32
        invite-expire                   180
        inactive-dynamic-conn           32
```

SIP Timers Discreet Configuration

Previous releases controlled various SIP timers with a single ACLI command, **trans-expire**, available in both sip-config and sip-interface modes. When executed in sip-config mode, the command essentially established a global default transaction expiration timer value. Executed at the sip-interface level, the command established a local, interface-specific value that overrode the global default.

Specific timers controlled by **trans-expire** are as follows:

Timer B, the INVITE transaction timeout timer, defined in Section 17.1.1.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer D, the Wait-Time for response retransmitals timer, defined in Section 17.1.1.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer F, the non-INVITE transaction timeout timer, defined in Section 17.1.2.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer H, the Wait-Time for ACK receipt timer, defined in Section 17.2.1 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer J, the Wait-Time for non-INVITE requests timer, defined in Section 17.2.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

A new ACLI command (**initial-inv-trans-expire**) that enables user control over SIP Timer B for initial INVITE transactions. Other timers, namely B for non-initial INVITEs, D, F, H, and J remain under the control of **trans-expire**.

Use **initial-inv-trans-expire** in the sip-config configuration mode, to establish a global, default transaction timeout value (expressed in seconds) used exclusively for initial INVITE transactions.

```
ORACLE(sip-config)# initial-inv-trans-expire 4
ORACLE(sip-config)#
```

Allowable values are integers within the range 0 (the default) through 999999999. The default value, 0, indicates that a dedicated INVITE Timer B is not enabled. Non-default integer values enable a dedicated Timer B and set the timer value.

The default value retains compatibility with previous operational behavior in that Timers B, D, F, H, and J all remain subject to the single timer value set by **trans-expire**. However, when **initial-inv-trans-expire** is et to a supported non-zero value, SIP Timer B as it applies to initial INVITEs, assumes that value rather than the value assigned by **trans-expire**. This functionality is available in both sip-config and in sip-interface objects.

If a dedicated Timer B is enabled at the sip-config level, you can use **initial-inv-trans-expire** in the sip-interface configuration mode, to establish a local interface-specific Timer B timeout value that overrides the global default value.

```
ORACLE(sip-interface)# initial-inv-trans-expire 8
ORACLE(sip-interface)#
```

Session Timer Support

The Oracle Communications Session Border Controller partially supports RFC4028 by establishing session timers without participating in the session timer negotiation.

When a 2xx response to a Session Refresh Request arrives, the Oracle Communications Session Border Controller will start a new timer or refresh the existing timer using the value of the Session-Expires header. When the session timer expires, the Oracle Communications Session Border Controller will send a BYE to both the upstream and downstream endpoints.

When accounting is configured, the Oracle Communications Session Border Controller will also send a RADIUS STOP record with Acct-Terminate-Cause=Session-Timeout.

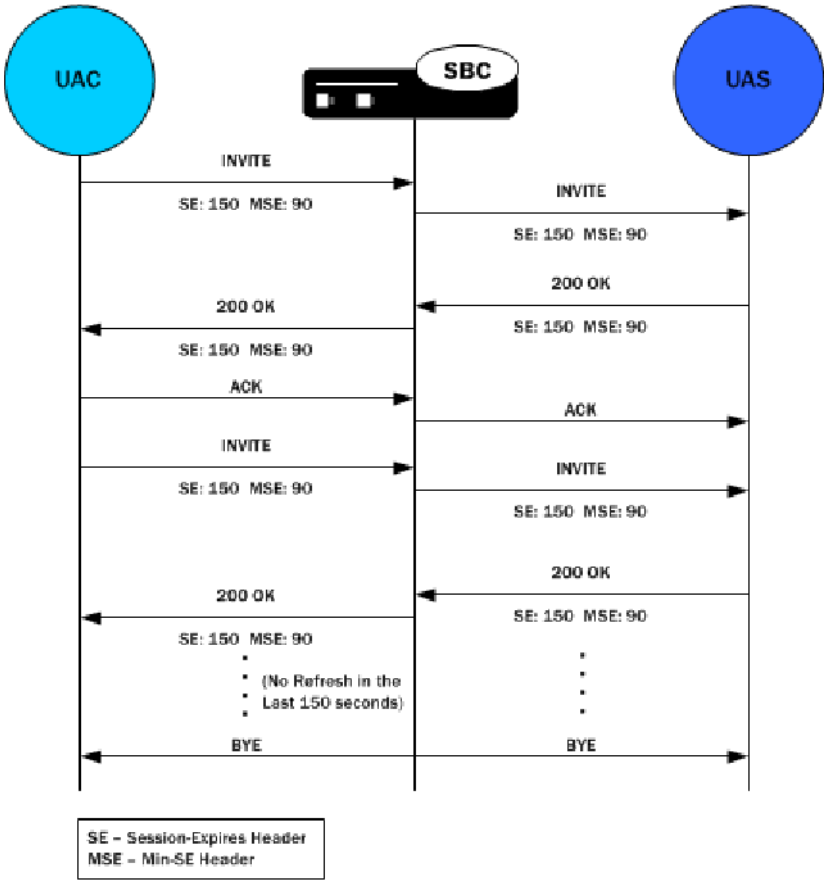
Call Flow Example

The UAS obtains the value from the Session-Expires header field in a 2xx response to a session refresh request that it sends.

Proxies and UACs obtain this value from the Session-Expires header field in a 2xx response to a session refresh request that they receive.

Once the session timer runs out, the Oracle Communications Session Border Controller sends a BYE to both the UAC and the UAS to clear the session.

Enable this feature by adding the **session-timer-support** option to the sip config.



SIP Per-User CAC

The Oracle Communications Session Border Controller's call admission control (CAC) supports an enhanced degree of granularity for SIP sessions.

Without this feature enabled, the Oracle Communications Session Border Controller performs call admission control (CAC) based on:

- Bandwidth limits configured in realms and nested realms
- Number of media flows available through the steering pool per realm
- Number of inbound sessions configured for a SIP session agent
- Number of total sessions (inbound and outbound) per SIP session agent
- Use of the Oracle Communications Session Border Controller's support for common open policy service (COPS), allowing the Oracle Communications Session Border Controller to perform CAC based on the policies hosted in an external policy server

These methods provide a basic level of call admission control in order to ensure that a SIP session agent's capacity is not exceeded. You can also ensure that signaling and media bandwidth capacities are not exceeded for physical trunks and peers.

With this feature enabled, the Oracle Communications Session Border Controller changes behavior so that it will only allow the configured number of calls or total bandwidth to and from each user in a particular realm. The overall realm bandwidth and steering pool limits still apply, and as before, the Oracle Communications Session Border Controller still rejects users who

might be within their CAC limitations if accepting them with exceed the bandwidth limitations for parent or child realms and steering pools.

For SIP sessions, the Oracle Communications Session Border Controller now keeps track of the amount of bandwidth a user consumes and the number of active sessions per address of record (AoR) or per IP address, depending on the CAC mode you select (either aor or ip). When an endpoint registers with the Oracle Communications Session Border Controller, theOracle Communications Session Border Controller allots it a total amount of bandwidth and total number of sessions.

This section describes the details of how SIP per user CAC works.

You should note that the functionality this section describes only works if you enable registration caching on your Oracle Communications Session Border Controller.

For SIP sessions, the Oracle Communications Session Border Controller now keeps track of the amount of bandwidth a user consumes and the number of active sessions per address of record (AoR) or per IP address, depending on the CAC mode you select (either aor or ip). When an endpoint registers with the Oracle Communications Session Border Controller, the Oracle Communications Session Border Controller allots it a total amount of bandwidth and total number of sessions.

Per User CAC Modes

There are three modes that you can set for this feature, and each has an impact on how the other two per-user-CAC parameters are implemented:

- none—No per user CAC is performed for users in the realm.
- aor—The Oracle Communications Session Border Controller performs per user CAC according to the AoR and the contact associated with that AoR for users in the realm.
- ip—The Oracle Communications Session Border Controller performs per user CAC according to the IP address and all endpoints that are sending REGISTER messages from the IP address for users in the realm.

Per User CAC Sessions

You can set the number of CAC for sessions per user in the realm configuration. Depending on the CAC mode you set, the sessions are shared between contacts for the same AoR or the endpoints behind the same IP address.

When it receives an INVITE, the Oracle Communications Session Border Controller determines the registration entry for the calling endpoint and the registration for the called endpoint. It then decides if session can be established between the two. If it can, the Oracle Communications Session Border Controller establishes the session and changes the active session count for the calling and called endpoints. The count is returned to its original value once the session is terminated.

Per User CAC Bandwidth

You can set the per user CAC bandwidth in realm configuration, too, and it is handled much the same way that the sessions are handled. That is, depending on the CAC mode you set, the bandwidth is shared between contacts for the AoR or the endpoints behind the same IP address. All endpoints must be registered with the Oracle Communications Session Border Controller.

When it receives a Request with SDP, the Oracle Communications Session Border Controller checks to see if there is enough bandwidth for the calling endpoint and for the called endpoint. The Oracle Communications Session Border Controller assumes that the bandwidth usage is symmetric, and it uses the maximum bandwidth configured for the codec that it finds in the Request. In the event that there are multiple streams, the Oracle Communications Session Border Controller determines the total bandwidth required for all of the streams. If the required bandwidth exceeds what is available for either endpoint, the Oracle Communications Session Border Controller rejects the call (with a 503 error response). If the amount of available bandwidth is sufficient, then the used bandwidth value is increased for both the registered endpoints: calling and called. Any mid-session requests for changes in bandwidth, such as those caused by modifications in codec use, are handled the same way.

The Oracle Communications Session Border Controller also keeps track of the bandwidth usage on a global level. When the call terminates, the bandwidth it was consuming is returned to the pool of available bandwidth.

Notes on HA Nodes

This feature has been implemented so that a newly active system is able to perform SIP per user CAC. The standby Oracle Communications Session Border Controller is updated with the appropriate parameters as part of the SIP session update.

SIP per User CAC Configuration

Note that you must enable registration caching for this feature to work.

To configure SIP per user CAC:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

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

4. Select the realm where you want to want to add SIP per user CAC.

   ```
   ORACLE(realm-config)# select
   ```

5. **user-cac-mode**—Set this parameter to the per user CAC mode that you want yo use. The default value is **none**. The valid values are:
   - **none**—No user CAC for users in this realm
   - **aor**—User CAC per AOR
   - **ip**—User CAC per IP
6. **user-cac-sessions**—Enter the maximum number of sessions per user for dynamic flows to and from the user. The default is **0**. Leaving this parameter set to its means that there is

unlimited sessions, meaning that the per user CAC feature is disabled in terms of the constraint on sessions. The valid range is:

7. Minimum—0
8. Maximum—999999999
9. **user-cac-bandwidth**—Enter the maximum bandwidth per user for dynamic flows to and from the user. The default is **0** and leaving this parameter set to the default means that there is unlimited bandwidth, meaning that the per user CAC feature is disabled in terms of the constraint on bandwidth. The valid range is:
   - Minimum—0
   - Maximum—999999999

SIP Per-Realm CAC

Building on the Oracle Communications Session Border Controller's pre-existing call admission control methods, CAC can be performed based on how many minutes are being used by SIP or H.323 calls per-realm for a calendar month.

In the realm configuration, you can now set a value representing the maximum number of minutes to use for SIP and H.323 session using that realm. Although the value you configure is in minutes, the Oracle Communications Session Border Controller performs CAC based on this value to the second. When you use this feature for configurations with nested realms, the parent realm will have the total minutes for all its child realms (i.e., at least the sum of minutes configured for the child realms).

The Oracle Communications Session Border Controller calculates the number of minutes used when a call completes, and counts both call legs for a call that uses the same realm for ingress and egress. The total time attributed to a call is the amount of time between connection (SIP 200 OK) and disconnect (SIP BYE), regardless of whether media is released or not; there is no pause for calls being placed on hold.

If the number of minutes is exhausted, the Oracle Communications Session Border Controller rejects calls with a SIP 503 Service Unavailable message (including additional information "monthly minutes exceeded). In the event that the limit is reached mid-call, the Oracle Communications Session Border Controller continues with the call that pushed the realm over its threshold but does not accept new calls. When the limit is exceeded, the Oracle Communications Session Border Controller issues an alarm and sends out a trap including the name of the realm; a trap is also sent when the alarm condition clears.

> **Note:**
>
> The Oracle Communications Session Border Controller does not reject GETS/NSEP calls based on monthly minutes CAC.

You can change the value for minutes-based CAC in a realm configuration at any time, though revising the value downward might cause limits to be reached. This value resets to zero (0) at the beginning of every month, and is checkpointed across both system in an HA node. Because this data changes so rapidly, however, the value will not persist across and HA node if both systems undergo simultaneous failure or reboot.

You can use the ACLI **show monthly minutes <realm-id>** command (where **<realm-id> is** the realm identifier of the specific realm for which you want data) to see how many minutes are

configured for a realm, how many of those are still available, and how many calls have been rejected due to exceeding the limit.

SIP per Realm CAC Configuration

This section shows you how to configure minutes-based CAC for realms and how to display minutes-based CAC data for a specific realm.

Enabling Realm-Based CAC

Note that setting the new monthly-minutes parameters to zero (0), or leaving it set to its default of 0, disables this feature.

To configure minutes-based CAC:

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

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

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

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

4. Select the realm where you want to want to add SIP per user CAC.

   ```
   ORACLE(realm-config)# select
   ```

5. **monthly-minutes**—Enter the number of minutes allowed during a calendar month in this realm for SIP and H.323 calls. By default, this parameter is set to zero (0), which disabled monthly minutes-based CAC. You can enter a value as high as 71582788.
6. Save and activate your configuration.

Viewing Realm-Based CAC Data

Use the ACLI show monthly-minutes command to see the following information:

- How many minutes are configured for a realm
- How many of those are still available
- How many calls have been rejected due to exceeding the limit
  To view information about SIP per user CAC using the IP address mode:

- In either User or Superuser mode, type **show monthly-minutes <realm-id>**, a Space, and the IP address for which you want to view data. Then press Enter. The **<realm-id>** is the realm identifier for. the realm identifier of the specific realm for which you want data

```
ORACLE# show monthly-minutes private_realm
```

SIP Options Tag Handling

This section explains how to configure SIP options on a global or per-realm level and how to specify whether the feature treatment applies to traffic inbound to or outbound from a realm, or both.

SIP extensions that require specific behavior by UAs or proxies are identified by option tags. Option tags are unique identifiers used to designate new options (for example, extensions) in SIP. These option tags appear in the Require, Proxy-Require, and Supported headers of SIP messages.

Option tags are compatibility mechanisms for extensions and are used in header fields such as Require, Supported, Proxy-Require, and Unsupported in support of SIP.

The option tag itself is a string that is associated with a particular SIP option (i.e., an extension). It identifies this option to SIP endpoints.

Overview

The SIP specification (RFC 3261) requires that the Oracle Communications Session Border Controller B2BUA reject any request that contains a Require header with an option tag the Oracle Communications Session Border Controller does not support. However, many of these extensions operate transparently through the Oracle Communications Session Border Controller's B2BUA. You can configure how SIP defines the Oracle Communications Session Border Controllers B2BUA treatment of specific option tags.

Also, there might be certain extensions that an endpoint indicates support for by including the option tag in a Supported header. If you do not want a given extension used in your network, the you can configure SIP option tag handling to remove the undesired option tag from the Supported header. You can also specify how option tags in Proxy-Require headers are to be treated.

Configuration Overview

You configure the SIP feature element to define option tag names and their treatment by the Oracle Communications Session Border Controller when the option tag appears in a Supported header, a Require header, and a Proxy-Require header. If an option tag is encountered that is not configured as a SIP feature, the default treatments apply. You only need to configure option tag handling in the SIP feature element when non-default treatment is required.

You can specify whether a SIP feature should be applied to a specific realm or globally across realms. You can also specify the treatment for an option based on whether it appears in an inbound or outbound packet. Inbound packets are those that are coming from a realm to the Oracle Communications Session Border Controller and outbound packets are those which are going from the Oracle Communications Session Border Controller to the realm.

The following tables lists the SIP option tag parameters you must configure.

Parameter	Description
name	SIP feature tag name
realm	Realm name with which the feature will be associated. To make the feature global, leave the field empty.
support mode inbound	Action for tag in Supported header in an inbound packet.
require mode inbound	Action for tag in Require header in an inbound packet
proxy require mode inbound	Action for tag in Proxy-Require header in an inbound packet
support mode outbound	Action for tag in Supported header in an outbound packet
require mode outbound	Action for tag in Require header in an outbound packet
proxy require mode outbound	Action for tag in Proxy-Require header in an outbound packet

SIP Option Tag Handling Configuration

To configure SIP option tag handling:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can configure SIP option tags parameters. To view all sip-feature parameters, enter a **?** at the system prompt.

4. **name**—Enter a name for the option tag that will appear in the Require, Supported, or Proxy-Require headers of inbound and outbound SIP messages.

   You must enter a unique value.

   > **Note:**
   >
   > Valid option tags are registered with the IANA Protocol Number Assignment Services under Session Initiation Protocol Parameters. Because option tags are not registered until the SIP extension is published as a RFC, there might be implementations based on Internet-Drafts or proprietary implementations that use unregistered option tags.

5. **realm**—Enter the name of the realm with which this option tag will be associated. If you want to apply it globally across realms, leave this parameter blank.
6. **support-mode-inbound**—Optional. Indicate the support mode to define how the option tag is treated when encountered in an inbound SIP message's Supported header. The default value is **pass**. Valid values are:

- **pass**—Indicates the B2BUA should include the tag in the corresponding outgoing message.
- **strip**—Indicates the tag should not be included in the outgoing message. Use strip if you do not want the extension used.

7. **require-mode-inbound**—Optional. Indicate the require mode to define how the option tag is treated when it is encountered in an inbound SIP message's Require header. The default value is **reject**. The valid values are:
   - **pass**—Indicates the B2BUA should include the tag in the corresponding outgoing message.
   - **reject**—Indicates the B2BUA should reject the request with a 420 (Bad Extension) response. The option tag is included in an Unsupported header in the reject response.
8. **support-mode-outbound**—Optional. Indicate the support mode to define how the option tag is treated when encountered in an outbound SIP message's Supported header. The default value is pass. Valid values are:
   - **pass**—Indicates the B2BUA should include the tag.
   - **strip**—Indicates the tag should not be included in the outgoing message. Use strip if you do not want the extension used.
9. **require-mode-outbound**—Optional. Indicate the require proxy mode to define how the option tag is treated when encountered in an outgoing SIP message's Proxy-Require header. The default value is **reject**. The valid values are:
   - **pass**—Indicates the B2BUA should include the tag.
   - **reject**—Indicates the B2BUA should reject the request with a 420 (Bad Extension) response. The option tag is included in an Unsupported header in the reject response.

   The following example shows SIP option tag handling configured for non-default treatment of option tags.

```
sip-feature
        name                            newfeature
        realm                           peer-1
        support-mode-inbound            Strip
        require-mode-inbound            Reject
        proxy-require-mode-inbound      Pass
        support-mode-outbound           Pass
        require-mode-outbound           Reject
        proxy-require-mode-outbound     Reject
        last-modified-date              2004-12-08 03:55:05
```

Replaces Header Support

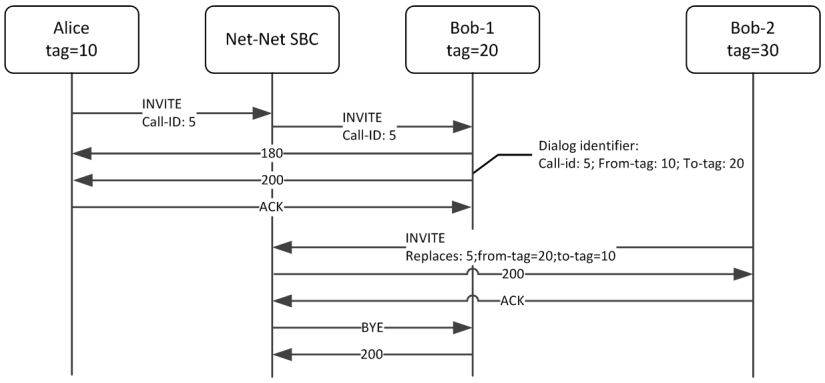
The Oracle Communications Session Border Controller (SBC) supports the Replaces: header in SIP messages according to RFC 3891. The header, included within SIP INVITE messages, provides a mechanism to replace an existing early or established dialog with a different dialog. The different dialog can be used for services such as call parking, attended call transfer and various conferencing features.

The Replaces: header specifies the dialog to replace by containing the corresponding dialog identifier. The identifier includes the from tag, to tag, and call id. The orientation of endpoint-created tags, as from-tag and to-tag, matches each of the two dialogs for a standard call. For

example, the Replaces: header from an endpoint that specifies assuming the dialog between the SBC and Bob-1 appears as follows:

```
Replaces:5555;from-tag=20;to-tag=10
```

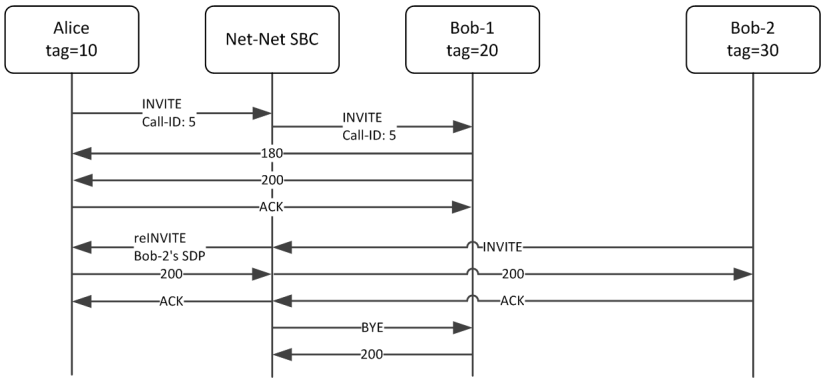
The SBC validates the dialog identifier by matching an existing dialog, tries to install the new endpoint, and tries to remove the old endpoint by gracefully ending that dialog with a BYE. The replaces INVITE must come from an endpoint in the same realm as the endpoint it is replacing. If the UA sending the Replaces header is in a different realm as the original call leg (or indicates such architecture from a malformed Replaces: header), the SBC replies to the Replaces: endpoint with a 481 Missing Dialog. Refer to the following diagram for the standard scenario.



Note that when the endpoints are in the same realm, you must enable the **mm-in-realm** parameter in realm-config or the SBC cannot generate the SDP for the 200 OK. Without the SDP, the call is unsuccessful.

New SDP Parameters in INVITE with Replaces

When an INVITE with Replaces: header is received, the media parameters in the new SDP are compared against the SDP of the dialog to be replaced. If any portion of the SDPs (excluding the session-origin line) is different, then the media must be renegotiated. The SBC sends a re-INVITE with the new SDP to the dialog opposite of the one being replaced as shown below. If the re-INVITE fails for any reason, then the original dialogs will remain.

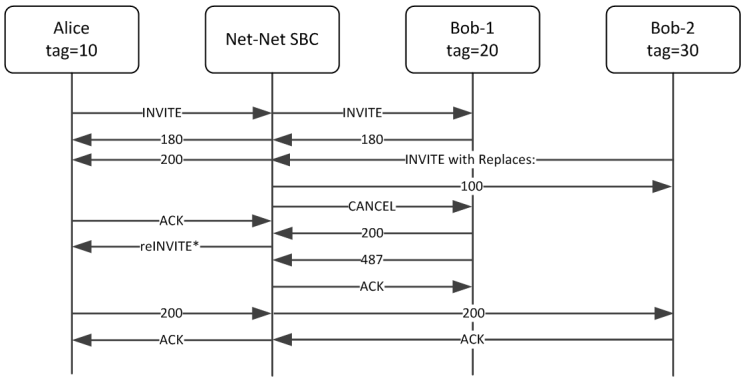


It is important to note that, if the SDP matches, the SBC suppresses this re-INVITE. As inferred above, when an INVITE with a REPLACES header arrives, the SBC notes the matching dialog

and then makes SDP changes, including codec manipulation, codec-policy application and, in the case of SRTP to RTP sessions, crypto removal. It is only after these changes that the SBC compares everything in the dialog's most recent SDP with the new SDP, excepting the SDP version and whitespace. If the SDP does match, the SBC suppresses the re-INVITE to the original client and proceeds with the replaced dialog.

Early Dialog Replacement

An INVITE with Replaces: header can replace an early dialog. That is, a dialog where the final 2xx class response to INVITE request has not arrived yet. The SBC completes the originating side of the call with a 200 OK. The original dialog with the terminator is cancelled. SDP from the new terminator can be renegotiated if it changes.



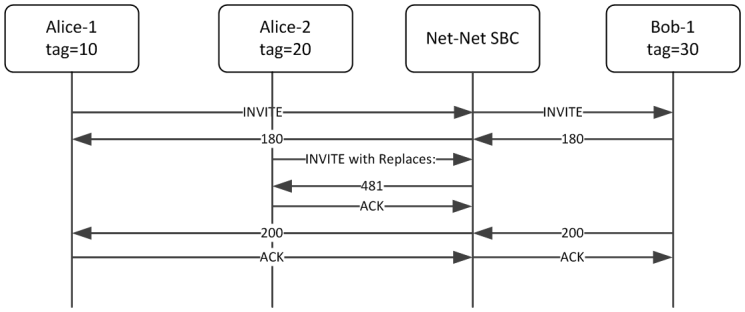
The SDP from the original 183 response is used for the 200 response back to the originator if present to complete the early transaction. If reliable provisional messages are used, then no SDP is included in the 200 response.

If no SDP is present in any of the provisional messages, then the SBC constructs it from the original offer and modifying the IP port information for each c= and m= line with information from the INVITE with Replaces: header. If there are more m= lines in the original offer than ports from the INVITE with Replaces: header, then the extra ports are disabled with port value of 0. If no SDP was offered in the original INVITE, then the SDP from the INVITE with Replaces: header is used as the offer in the 200 OK.

If the SDP media parameters were compatible between the replaced and replacing SDPs, then media does not need to be renegotiated and no re-INVITE is created. If the re-INVITE fails, the original dialogs are torn down using a BYE for the original server dialog.

INVITE with Replaces in Early Dialog Server Side

The Oracle Communications Session Border Controller does not support replacing an early server dialog. It replies with a 481 (Dialog/Transaction does not exist) response to the endpoint requesting the replace.



Replace Header Configuration

Replaces: header support is configured in the session-agent, realm, or sip-interface via the sip-profile configuration element. sip-profiles are defined once and attached to a chosen interface, realm or session-agent.

The replace-dialogs parameter is set to either **enabled** or **disabled**. In addition, you may set this parameter to **inherit** which uses the next lower order of precedence object. If there are no sip-profiles referenced in the higher ordered object, or if all the replace-dialogs parameters are set to **inherit**, then the feature is disabled.

To configure Replaces: header support:

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

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

2. Type **session-router** and press Enter.

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

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

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

4. **replace-dialogs**—Set this parameter to **enabled** to enable Replaces: header support. A replaces parameter is also inservted in the Supported: header as sent into the realm where this sip profile is applied. You may also set this parameter to **inherit** for the element which this sip-profile is applied to, to inheret the value from the next-lower element.
5. Type **done** to save your work and continue.

Debugging

show sipd status

Includes Replaced Dialogs counts to show successfully replaced dialogs:

```
# show sipd status
17:41:38-142
```

```
SIP Status                -- Period -- -------- Lifetime --------
                  Active     High   Total      Total  PerMax    High
Replaced Dialogs      -        -       1          1       1
```

show sipd errors

Includes Replace Dialog Fails to show failed dialog replacements. This counter is incremented only when the dialog replacement attempt actually occurred but failed to successfully complete.

```
# show sipd errors
17:58:04-181
SIP Errors/Events             ---- Lifetime ----
                        Recent      Total  PerMax
Replace Dialog Fails         0          0       0
```

SIP Options

This section explains how you can configure a limited list of specialized SIP features and/or parameters called options. The options described here were developed to meet specific needs not addressed by the standard SIP configuration parameters. Not all users have a need for these options.

> **Note:**
>
> Oracle recommends checking with your Oracle representative before applying any of these options.

Overview

You can configure options for the SIP configuration and SIP interface. Both elements include a parameter (options) that you use to configure the options.

Global SIP Options

The following table lists the SIP options supported by the Oracle Communications Session Border Controller (SBC).

Option	Description
add-error-to-tag=no	If present (even when set to no), suppresses the addition of an Acme tag on 3xx-6xx responses.
add-prov-to-tag=no	Prevents the SBC from adding a tag parameter to the To header (to-tag) to non-100 provisional responses to INVITE requests. Used when a provisional (101-199) response is received from the UAS on a client transaction without a to-tag. By default, the SBC adds the tag cookie in the response (as though it had a tag) sent back to the UAC for the associated server transaction. When you include this option in the SIP configuration, and the response from the UAS does not have a to-tag, the response forwarded to the UAC will not have a to-tag.

Option	Description
add-reg-expires	Causes an Expires header to always be included in a REGISTER response with the registration caching and HNT traversal functions of the SBC. Use for endpoints that do not understand the Expires parameter in the Contact header.
add-ruri-user=<methods>	Causes a userinfo portion to be added to a Request-URI when one is not present. Used to support the OKI phone, which registers a Contact of just an IP-Address but rejects initial INVITEs if the Request_URI does not have a userinfo part. <methods> is a comma-separated list of methods to which the option should apply. If more than one method is listed, the list must be enclosed in quotes. This option only applies to out-of-dialog requests (no tag parameter in the To header). However, if ACK is listed, it will apply to all ACK requests because an ACK is always supposed to have a to-tag.
allow-notify-no-contact	Prevents the SBC from rejecting NOTIFYs with a 400 Bad Request response.NOTIFY requests without Contact header are allowed to pass through the SBC instead.
call-id-host=<host>	Causes the SBC to include a host part (ID@host) in the Call-ID it generated. <host> is the hostname (or IP address) that is to appear in the host part of the Call-ID. If not specified, the SIP port address is used.
contact-endpoint=<param-name>	Defines a URL parameter to report the real Contact address of an endpoint in a REGISTER message forwarded to a registrar, when the SBC is caching registration. (plain or HNT). If <param-name> is not specified, the default value endpoint is used. This parameter is added as a URL parameter in the Contact on the REGISTER message. In order for the registration cache to work properly, the softswitch/ registrar is expected to include the endpoint parameter in the Request-URI of a SIP request it forwards to the address-of-record.
contact-firewall=<param-name>	Defines a URL parameter to report the NAT between the SBC and the real Contact address of an endpoint in a REGISTRAR message forwarded to a registrar when the SBC is doing registration caching for NHT. If <param-name> is not specified, the default value firewall is used. This parameter will be added as a URL parameter in the Contact on the REGISTER message. In order for the registration cache to work properly, the softswitch/ registrar is expected to include the endpoint parameter in the Request-URI of any SIP request it forwards for the address-of-record.
disable-privacy	Prevents the change of the P-Preferred-Identity to P-Asserted-Identity and lets the P-Preferred-Identity go through unchanged.
drain-sendonly	Causes the SBC to examine the SDP attributes and change sendonly mode to sendrecv. This causes the endpoint receiving the SDP to send RTP, which is required for HNT traversal endpoints to work with media servers. The SBC sets up the flow so that RTP coming from the endpoint are dropped to prevent the UA that sent the sendonly SDP from receiving packets. See the option video-sbc-session also.

Option	Description
encode-contact=<prefix>	Causes the SBC to encode Contact addresses into the userinfo part of the URI. It applies only to Contact address that usually get the maddr parameter. Use when the SBC needs requests sent to the URI in the Contact sent instead to the SBC. The host part of the URI will have the SBC's address. The <prefix> serves as a place between the original userinfo and the encoded address. If a <prefix> is specified, a default of +SD is used. Without this option, the SBC adds a maddr parameter.
fix-to-header	For requests that have the SBC address in both the Request-URI and the To-URI, it sets the hostport of the To-URI to a local policy's next hop target on out-of-dialog requests (no to-tag). This is the default IWF behavior, even without this option configured.
forward-reg-callid-change	Addresses the case when an endpoint reboots and performs a third party registration before its old registration expires. During this re-registration, the contact header is the same as it was pre-reregistration. As a consequence of the reboot, the SIP Call-ID changes. In this situation, the SBC does not forward the REGISTER to the registrar, because it believes the endpoint is already registered, based on a previous registration from the same Contact: header URI. To remedy this problem, the SBC now keeps track of the Call-ID in its registration cache. A new option in the SIP interface configuration element forces the SBC to forward a REGISTER message to the registrar when the Call-ID header changes in a REGISTER message received from a reregistering UAC.
global-contact	Addresses interoperability in the Dialog and Presence event packages that are used in hosted PBX and IP Centrex offerings. This option enables persistent URIs in the Contact headers inserted into outgoing SIP messages. If this option is not used, URIs placed in the Contact header of outgoing messages are only valid within the context of the dialog to which the message is associated.
ignore-register-service-route-oos	Prohibits a Register message from using a service route if that service route is an out-of-service session agent.
load-limit=<cpu percentage>	Defines the CPU usage percentage at which the SBC should start rejecting calls. Default value is 90%.
lp-sa-match=<match strategy>	Changes the ways local policies and session agents match; accounts for realm in matching process. Strategy choices are: all, realm, sub-realm, interface, and network.
max-register-forward=<value>	Defines a limit (as assigned in the value field) of REGISTER refreshes to be forwarded to the registrar. During each second, the sipd counts how many REGISTER refreshes have been sent to the registrar. It checks the threshold when it receives a REGISTER refresh from the UA and determines that less than half the real registration lifetime is left. If the number of REGISTER refreshes forwarded (new and updates) in the current second exceeds the configured threshold, it will respond to the UA from the cache.

Option	Description
max-register-refresh=<value>	Defines the desired limit of REGISTER refreshes from all the UAs. Each second of time, sipd counts the number of REGISTER/200-OK responses sent back. When the threshold is exceeded, it increments the expire time (based on NAT interval) by one second and resets the count. By default no threshold is applied. The recommended value is somewhat dependent on the SBC hardware used, but 300 can be used as an initial value.
max-routes=<number of routes>	Restricts the number of routes through which the sipd will iterate from a local policy lookup. For example, setting this option to 1 causes the SBC to only try the first, best, route. Setting this option to 0, or omitting it, lets the SBC use all of the routes available to it (with the priority scheme for route matching). When you test a policy using the test-policy ACLI command, this option is not recognized and all options that match the criteria are displayed.
max-udp-length=<maximum length>	Setting this option to zero (0) forces sipd to send fragmented UDP packets. Using this option, you override the default value of the maximum UDP datagram size (1500 bytes; sipd requires the use of SIP/TCP at 1300 bytes). You can set the global SIP configuration's max-udp-length=x option for global use in your SIP configuration, or you can override it on a per-interface basis by configuring this option in a SIP interface configuration.
media-release=<header-name>[;<header-param>]	Enables the multi-system media release feature that encodes IP address and port information for the media streams described by SDP. It lets another SBC decode the data to restore the original SDP, which allows the media to flow directly between endpoints in the same network (that is serviced by multiple SBCs). The media release information can appear in the following places: • SIP header P-Media-Release: <encoded-media-interface-information> • Header parameter on a SIP header Contact: <sip:1234@abc.com> ; acme-media=<encoded-media-interface-information> • SDP attribute in the message body a=acme-media: <encoded-media-interface-information> Option includes the following: • <header-name> is SIP header in which to put the information or the special value sdp, which indicates the information should be put into the SDP. • <header-param> is the header parameter name in which to put the information or in the case of the special header name value sdp, it is the SDP attribute name in which to put the information. They identify to where the encoded information is passed. If you do not specify a header, P-Media-Release is used.

Option	Description
no-contact-endpoint-port	Enables the SBC to add a URL parameter (defined as an argument to the contact-endpoint option) to the Contact headers of REGISTER messages that it forwards to the registrar when it performs registration caching. The value of the contact-endpoint URL parameter is the real address of the endpoint; and if the endpoint is behind a NAT, this includes the IP address and a port number. However, not all network entities can parse that port number, which is included unconditionally. This feature allows you to configure the exclusion of the port number. Despite the fact that you set this parameter in the global SIP configuration, it is applied only to SIP interfaces. However, you can set a contact-endpoint option in the realm configuration, on which this new parameter has no effect.
refer-to-uri-prefix=<prefix>	Defines a prefix to be matched against the userinfo part of Contact headers (config=), of which the SBC should create a B2BUA map. This ensures that outgoing messages include the correct userinfo value. This option is used to enable add-on conferencing. 1. On the SBC, set **refer-to-uri-prefix=<string>**, for example, **refer-to-uri-prefix="conf="**. 2. When the SBC receives either an INVITE or 200 OK, it stores the Contact:sip:conf=<ID@IP:port> contained in the SIP message. 3. When the SBC receives a REFER in a separate call session, the config=<ID>@ IP2:port2 in this message is different. 4. The SBC searches for the original conf=<ID> map, replaces the IP2:port2 with the stored IP:port, and forwards the updated REFER message.
reg-cache-mode=<mode>	Affects how the userinfo part of Contact address is constructed with registration caching. <mode> values are: • none: userinfo from the received (post NAT) Contact is retained • from: userinfo from the From header is copied to the userinfo of the forwarded Contact header • append: append the UA's Contact address into a cookie appended to the userinfo from the original Contact userinfo. For HNT, the NAT/firewall address is used. • append-from: takes userinfo from the From header and appends the encrypted address to the userinfo from the original Contact userinfo. For HNT, the NAT/firewall address is used. The from mode is used with softswitches that do not use the cookies used by the SBC. It also helps limit the number of bytes in the userinfo; which might create duplicate contacts. For example, if the SBC address is 1.2.3.4, both 1234@5.6.7.8 and 1234@4.3.2.1 will result in a SBC contact of 1234@5.6.7.8.

Option	Description
reg-contact-user-random	Support the SIP random registered-contact feature. Gives the SBC the ability to support endpoints that randomly change their contact usernames every time they re-register. Only applicable to operators who need to support the Japan TTC standard JJ-90.22 in specific applications. Applies to cases when an endpoint re-registers with a different contact username, but with the same hostname/IP address and the same address of record (AoR). Without this feature enabled, the SBC forwards every re-registration to the registrar with the new contact information without it being considered a registration refresh. The SBC forwards it to the Registrar using the same sd-contact as the previous registration. When you set this option, the SBC does treat such a re-registration as a registration refresh when it is received prior to the half-life time for the specific contact. The SBC also uses the new contact username for the Request-URI in requests it sends to the UA, and verifies that the UA uses the correct one when that SBC is set to allow-anonymous registered mode. NOTE: The registration cache mode is set using the option reg-cache-mode, but regardless of how you configure it, the registration cache mode will be set to contact when SIP random registered-contact feature is enabled.
register-grace-timer	Makes the grace time for the SIP Registration configurable. You can configure the grace timer in seconds.
reinvite-trying=[yes | no]	When set to "yes", the SBC sends a 100 Trying in response to Re-Invites. Whet set to "no", it does not. If you enter the option but omit the value, the option takes the value "yes". The default SBC behavior (the option not set) works differently on standalone and HA systems. For standalone, the default is to send the 100 Trying for Re-Invites. For HA, it does not. The difference between default behaviors is because of the timing involved with switching over. An active SBC can receive the re-INVITE before receiving a 200 OK. In that case, a new active SBC handles the re-INVITE properly after a switchover. But, if the SBC sends a 100 Trying before the switchover, the new active is not aware of the 200 OK transaction, and therefore discards the re-INVITE as a stray message.
reject-interval=<value>	Acts as a multiplier to increase the value presented to the UAC in the Retry-After field. For example, if reject-interval=5 (reject interval is set to 10); at a 90% rejection rate the SBC sends Retry-After: 45. When rejecting calls because of CPU load limiting, the SBC adds a Retry-After parameter to the error response (typically 503 Service Unavailable). By default the SBC sets the Retry-After value to be 1/10th of the current rejection rate.
reject-register=[no | refresh]	Allows REGISTER messages through even during load limiting. By default, REGISTER messages are subject to load limiting.
response-for-not-found=<response code>	Change the 404 Not Found generated by the SBC to a different response code.

Option	Description
route-register-no-service-route	Controls how a UA is registered. Option can have three values: • route-register-no-service-route—This option prevents the use of the Service-Route procedure to route the Re-Register requests after the UA has initially registered. • route-register-no-service-route=all—Prevents the use of the Service-Route procedure to route the Re-Register requests for all messages, after the UA has initially registered. • route-register-no-service-route=refresh—Prevents the use of the Service-Route procedure to route the Re-Register requests for all refresh-register messages, but not de-register messages, after the UA has initially registered. Addition idle argument ensures that, when enabled, the SBC follows the previously defined rules for idle calls, where idle means not engaged in any INVITE-based sessions. Sample syntax: route-register-no-service-route=refresh;idle
sdp-insert-sendrecv	When a call is initiated, the SDP communicates between call offerer and call answerer to determine a route for the media. Devices can be configured to only send media (“a=sendonly”), to only receive media (“a=recvonly”), or to do both (“a=sendrecv”). Some devices, do not disclose this information. With this option configured, when either the offerer or answerer does not disclose its directional attribute, the SBC automatically inserts a sendrecv direction attribute to the media session.
set-inv-exp-at-100-resp	Set Timer C when a 100 Trying response is received (instead of waiting until 1xx (> 100) is received). If the SBC does not receive a 100 Trying response within Timer B, the call should be dropped because there is a problem communicating with the next hop.
strip-domain-suffix-route	Causes sipd to strip any Router headers from the inbound messages coming to the external address of a SIP NAT; if the message contains a FQDN that matches the configured domain suffix for that SIP NAT.
video-sbc-session	Use with drain-sendonly for conference floor support. When configured with drain-sendonly and when the SBC receives an SDP, the SBC proxies the m=control and its related a= and c= unchanged. Although media streams are allocated for this m line, an actual flow is not set up. SDP received with the following: m=video a=sendonly is sent out as the following: m=video a=sendonly a=X-SBC-Session
session-timer-support	This option enables the SBC to start the session timer for session refreshes coming from the UAC. The SBC determines whether or not a session is active based on session refreshes or responses. It terminates the session when no session refreshes occur within the session timer interval.
session-timer-support	Enables RFC4028 session timer support.
inmanip-before-validate	Enables SIP Header Pre-processing for HMR.
proccess-implicit-tel-URI	Correctly appends coodie in REGISTER message when user=phone does not exist.
offerless-bw-media	Reserves appropriate bandwidth for an INVITE with no SDP.

SIP Interface Options

The following table lists the SIP interface options supported by the Oracle Communications Session Border Controller (SBC).

Option	Description
100rel-interworking	Enables RFC 3262, *Reliability of Provisional Responses in the Session Initiation Protocol* support.
change-contact-user	The SBC alters the **user-uri** portion of the contact header. This option works only when the **user-uri** contact header is different than the FROM header. 1. In **sip-config**, **options** apply `reg-cache-mode=from`. 2. In the ingress **sip-interface**, **options** apply `change-contact-user`. When these settings are applied, the contact header's user-uri is changed to the user-uri of the FROM header. This feature works for out of dialog INVITE and in-dialog messages (ACK, BYE, UPDATE,REFER,INFO,SUBSCRIBE,NOTIFY,PUBLISH,MESSAGE,PRACK), however, does not work for REGISTER messages.
contact-endpoint=<endpoint name>	The SBC inserts the endpoint IP address and port into the Contact headers as messages egress using that SIP interface. The inserted data is the same as the information received in the Request or Response being forwarded. If the endpoint name is not specified, the default value endpoint is used.
contact-firewall=<firewall name>	The SBC inserts the firewall IP address and port into the Contact headers as messages egress using that SIP interface. The inserted data is the same as the information received in the Request or Response being forwarded. If the endpoint name is not specified, the default value firewall is used.
contact-vlan=<VLAN/realm name>	The SBC inserts the realm and VLAN ID into the Contact headers as messages egress using that SIP interface. The inserted data is the same as the information received in the Request or Response being forwarded. If the endpoint name is not specified, the default value vlan is used.

Option	Description
dropResponse	The SBC drops responses by specified status codes. The option value can contain one or more status codes separated by colons. Response code ranges can also be entered. If any of the response codes matches, then a response is not sent. If the dropResponse option is set in both the sip-interface and the session-agent elements, the session-agent setting takes precedence. **Note:** This feature screens all messages passing through SBC against the dropResponse list. If there is a hit, the response message is dropped. However, for error messages generated by the SBC, only the response messages generated after the initial routing is complete are screened against the dropResponse list.
max-udp-length=<maximum length>	Sets the largest UDP packers that the SBC will pass. Packets exceeding this length trigger the establishment of an outgoing TCP session to deliver the packet; this margin is defined in RFC 3261. The system default for the maximum UDP packet length is 1500. You can set the global SIP configuration's max-udp-length=x option for global use in your SIP configuration, or you can override it on a per-interface basis by configuring this option in a SIP interface configuration.
response-for-not-found=<response code>	Change the 404 Not Found generated by the SBC to a different response code.
strip-route-headers	Causes the SBC to disregard and strip all route headers for requests received on a SIP interface.
upd-fallback	When a request needs to be sent out on the SIP interface for which you have configured this option, the SBC first tries to send it over TCP. If the SIP endpoint does not support TCP, however, then the SBC falls back to UDP and tries the request again.
via-header-transparency	Enables the SBC to insert its Via header on top of the top-most Via header received from user equipment (UE). It then forwards it on to the IP Multimedia Subsystem (IMS) core with the original Via header now located as the bottom-most Via header. The SBC still replaces the Contact and other header addresses with its own, and does not pass on the core's Via headers in outbound requests.
use-redirect-route	Use Route parameter in Contact header as next-hop as received in a 3xx response.
reg-via-proxy	Enables your SBC to support endpoints that register using an intervening proxy.
lmsd-interworking	Enables 3GPP2 LMSD Interworking.
suppress-reinvite	Enables reINVITE supression.

SIP Session Agent Options

The following table lists the SIP session agent options supported by the Oracle Communications Session Border Controller.

Option	Description
dropResponse	The Oracle Communications Session Border Controllerdrops responses by specified status codes. The option value can contain one or more status codes separated by semicolons. Error ranges can also be entered. If any of the response codes matches then a response is not sent. If the dropResponse option is set in both the sip-interface and the session-agent elements, the session-agent setting takes precedence.
trans-timeouts=<value>	Defines the number of consecutive non-ping transaction timeouts that will cause a session agent to be out of service. When the session agent is configured, i.e. when the PING options are defined, the value is 10. If not defined, the default value is 5. A Value of 0 prevents the session agent from going out of service because of a non-ping transaction timeout.
via-origin=<parameter-name>	Causes a parameter to be included in the top Via header of requests sent to the session agent. The parameter indicates the source IP address of the corresponding request received by the Oracle Communications Session Border Controller. <parameter-name> defines the name of the parameter. If not specified, the default value origin is used.
refer-reinvite	Enables SIP REFER with Replaces.

SIP Realm Options

The following table lists the SIP session agent options supported by the Oracle Communications Session Border Controller.

Option	Description
number-normalization	Applies to the SIP To URI. (Currently the Oracle Communications Session Border Controller supports number normalization on From and To addresses for both inbound and outbound call legs.) Number normalization includes add, delete, and replace string functions that result in consistent number formats. Number normalization occurs on ingress traffic, prior to the generation of accounting records or local policy lookups. (also applies for H.323 to SIP calls.)
refer-reinvite	Enables SIP REFER with Replaces.

SIP Realm Options Configuration

To configure options:

Labels enclosed in <> indicate that a value for the option is to be substituted for the label. For example, <value>. In order to change a portion of an options field entry, you must re-type the entire field entry.

1. Navigate to the options parameter in the SIP configuration or SIP interface elements.

2. Enter the following:

   ```
   options Space <option name>=”<value>”
   ```

   For example, if you want to configure the refer-to-uri-prefix option (the add-on conferencing feature):

   Type options, followed by a Space.

   Type refer-to-uri-prefix, followed by an equal sign (=).

   Type the opening quotation mark (“) followed by conf, another equal sign and the closing quotation mark.

   Press Enter.

   For example:

   ```
   options refer-to-uri-prefix=conf=
   ```

   If the feature value itself is a comma-separated list, it must be enclosed in quotation marks.

Configuring Multiple Options

You can enter a list of options for this field:

1. Type **options** followed by a space.
2. Within quotation marks, enter the feature names and values of the parameters you need. Separate each one with a comma.
3. Close the quotation marks.
4. Press Enter.

   For example:

   ```
   ACMEPACKET(sip-config)# options “refer-to-uri-prefix=“conf=”,encode-
   contact=“+SD”,add-ruri-user=INVITE,ACK”
   ```

Adding an Entry

Enter the new entry with a preceding plus (+) sign. For example:

```
options +response-for-not-found
```

This format allows previously configured options field values to remain intact without requiring re-entry of the entire field value.

SIP Security

This section provides an overview of Oracle Communications Session Border Controller’s security capability. Oracle Communications Session Border Controller security is designed to provide security for VoIP and other multi-media services. It includes access control, DoS attack, and overload protection, which help secure service and protect the network infrastructure (including the Oracle Communications Session Border Controller). In addition,

Oracle Communications Session Border Controller security lets legitimate users to still place calls during attack conditions, protecting the service itself.

Oracle Communications Session Border Controller security includes the Net-SAFE framework's numerous features and architecture designs. Net-SAFE is a requirements framework for the components required to provide protection for the Session Border Controller (SBC), the service provider's infrastructure equipment (proxies, gateways, call agents, application servers, and so on), and the service itself.

Denial of Service Protection

The Oracle Communications Session Border Controller Denial of Service (DoS) protection functionality protects softswitches and gateways with overload protection, dynamic and static access control, and trusted device classification and separation at Layers 3-5. The Oracle Communications Session Border Controller itself is protected from signaling and media overload, but more importantly the feature allows legitimate, trusted devices to continue receiving service even during an attack. DoS protection prevents the Oracle Communications Session Border Controller host processor from being overwhelmed by a targeted DoS attack from the following:

- IP packets from an untrusted source as defined by provisioned or dynamic ACLs
- IP packets for unsupported or disabled protocols
- Nonconforming/malformed (garbage) packets to signaling ports
- Volume-based attack (flood) of valid or invalid call requests, signaling messages, and so on.
- Overload of valid or invalid call requests from legitimate, trusted sources

Levels of DoS Protection

The multi-level Oracle Communications Session Border Controller Denial of Service protection consists of the following strategies:

- Fast path filtering/access control: involves access control for signaling packets destined for the Oracle Communications Session Border Controller host processor as well as media (RTP) packets. The SBC accomplishes media filtering using the existing dynamic pinhole firewall capabilities. Fast path filtering packets destined for the host processor require the configuration and management of a trusted list and a deny list for each Oracle Communications Session Border Controller realm (although the actual devices can be dynamically trusted or denied by the Oracle Communications Session Border Controller based on configuration). You do not have to provision every endpoint/device on the Oracle Communications Session Border Controller, but instead retain the default values.
- Host path protection: includes flow classification, host path policing and unique signaling flow policing. Fast path filtering alone cannot protect the Oracle Communications Session Border Controller host processor from being overwhelmed by a malicious attack from a trusted source. The host path and individual signaling flows must be policed to ensure that a volume-based attack will not overwhelm the Oracle Communications Session Border Controller's normal call processing; and subsequently not overwhelm systems beyond it. The Oracle Communications Session Border Controller must classify each source based on its ability to pass certain criteria that is signaling- and application-dependent. At first each source is considered untrusted with the possibility of being promoted to fully trusted. The Oracle Communications Session Border Controller maintains two host paths, one for each class of traffic (trusted and untrusted), with different policing characteristics to ensure that fully trusted traffic always gets precedence.

- Host-based malicious source detection and isolation – dynamic deny list. Malicious sources can be automatically detected in real-time and denied in the fast path to block them from reaching the host processor.

Configuration Overview

NAT table entries are used to filter out undesired IP addresses (deny list). After the packet from an endpoint is accepted through NAT filtering, policing is implemented in the Traffic Manager based on the sender's IP address. NAT table entries are used to distinguish signaling packets coming in from different sources for policing purposes.

You can configure deny rules based on the following:

- ingress realm
- source IP address
- transport protocol (TCP/UDP)
- application protocol (SIP)

You can configure guaranteed minimum bandwidth for trusted and untrusted signaling paths.

You can configure signaling path policing parameters for individual source addresses. Policing parameters include:

- peak data rate in bits per second
- average data rate in bits per second
- maximum burst size

SIP Unauthorized Endpoint Call Routing

The Oracle Communications Session Border Controller (SBC) can route new dialog-creating SIP INVITEs from unauthorized endpoints to a session agent or session agent group; then rejection can occur based on the allow-anonymous setting for the SIP port. This type of provisional acceptance and subsequent rejection applies only to INVITEs; the SBC continues to reject all other requests, such as SUBSCRIBE.

You might enable this feature if you have a network in which unauthorized SIP endpoints continually try to register even if the Oracle Communications Session Border Controller has previously rejected them and never will accept them. For instance, the user account associated with the endpoint might have been removed or core registrars might be overloaded.

SIP Unauthorized Endpoint Call Routing Configuration

You enable the routing of unauthorized endpoints to session agents and session agent groups that will reject them in the SIP interface configuration.

To enable SIP unauthorized endpoint call routing:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-interface** and press Enter.

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

   If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **route-unauthorized-calls**—Enter the name (or IP address) of the session agent or session agent group to which you want calls from unauthorized endpoints routed. This parameter is blank by default, meaning the SIP unauthorized call routing feature is disabled.

   Remember your settings in the **allow-anonymous** parameter in the SIP port configuration provide the basis for rejection.

5. Save and activate your configuration.

SIP NAT Function

This section explains how to configure the optional SIP NAT function. You can configure the SIP NAT function if you need to translate IP address and UDP/TCP port information. The SIP NAT function also prevents private IP addresses in SIP message URIs from traveling through an untrusted network.

Overview

TheOracle Communications Session Border Controller is an intermediary device that provides NAT functions between two or more realms. It translates IP addresses between untrusted and trusted networks using NAT. A trusted network is inside the NAT, and a untrusted network is outside the NAT. A NAT also lets a single IP address represent a group of computers.

For SIP, the SIP NAT function on the Oracle Communications Session Border Controller does the following:

- routes SIP packets between the Oracle Communications Session Border Controller's SIP proxy (B2BUA) and external networks (or realms), including the translation of IP address and UDP/TCP port information.
- prevents private IP addresses in SIP message URIs from traveling through the untrusted network. SIP NAT either translates the private address to one appropriate for an untrusted address or encrypts the private address into the URI.

Packets arriving on the external address (at port 5060) are forwarded to the Oracle Communications Session Border Controller's SIP proxy with the source address changed to the home address (at port 5060). When the Oracle Communications Session Border Controller's SIP proxy sends packets to the home address (at port 5060), they are forwarded to the external proxy address (and external proxy port), with the source address changed to the external address (at port 5060).

> **Note:**
>
> The SIP config's NAT mode parameter works in conjunction with the SIP NAT function configuration. It identifies the type of realm in which the SIP proxy is located (public or private) and affects whether IPvr addresses in SIP messages are encoded.

The translation of URIs in the actual SIP message occurs as messages are received and sent from the Oracle Communications Session Border Controller's SIP proxy. For the messages being sent to the external network, the contents of the SIP message are examined after the translation to determine if the destination needs to be changed from the external proxy address to an address and port indicated by the SIP message. This process takes place so the request is sent to where the Request-URI or the Route header indicates, or so the response is sent to where the Via indicates.

NAT Modes

The specific addresses used in translating URIs in the SIP message depend on whether the Oracle Communications Session Border Controller is performing NAT functions for a trusted or untrusted network. This condition is determined by the NAT mode value you enter when you configure the SIP config element. The NAT modes are:

- **untrusted**—The SIP proxy is associated with an address for an untrusted network (the address value you entered when you configured the SIP interface's SIP port parameter), and the home address in the SIP NAT is the address of the external realm/network. When the URI contains the external address, it is translated to the SIP NAT's home proxy address (or to the SIP port address if the home proxy address field is empty). When a URI contains the external proxy address, it is translated to the home address.
  If the URI contains any other private address (matching the realm's address prefix, identified in the SIP NAT's realm ID), it is encrypted and the address is replaced with the home address value. If the URI contains a user part, a suffix consisting of the user NAT tag and the encrypted address is appended to the user part. For example, with a user NAT tag value of -private-, the private URI of sip@123192.169.200.17:5060 will become the public URI of sip:123-private-eolmhet2chbl3@172.16.0.15.

  If there is no user part, the host consists of the host NAT tag followed by the encrypted address and the domain suffix. A maddr parameter equal to the home address (or received in the case of a Via header) is added to the URI. For example, with a host NAT tag value of PRIVATE- and a domain suffix value of private.com, the private URI of sip:192.168.200.17:5060 will become the public URI of sip:PRIVATE-eolmhet2chbl3.private.com:5060;maddr=172.16.0.15.
- **trusted**—The SIP proxy is on a trusted network (the address value you entered when you configured the SIP interface's SIP port parameter), and the SIP NAT's external address is the public address of the external realm/network. When the URI contains the home address value, it is translated to the value set for the external proxy address. When the URI contains the SIP proxy's address, it is translated to the external address. If the URI contains any other private address (matching the realm's address prefix, identified in the SIP NAT's realm ID), the private address is encrypted and the address is replaced with the external address.

> **Note:**
>
> Do not use the home proxy address value with private NAT functioning.

Adding a maddr Parameter to a URI

When you configure a SIP interface, you can configure the contact mode. The contact mode sets the contact header routing mode, which determines how the contact address from a trusted network is formatted. You set the contact mode to add a maddr parameter equal to the SIP proxy to the URI in the Contact header. For example, the URI from the prior example (sip:192.168.200.17:5060) becomes sip:123-trusted-eolmhet2chbl3@172.16.0.15;maddr=172.16.0.12.

> **Note:**
>
> For SIP elements that do not support the maddr parameter, configure a Contact mode as none.

You might require this encryption to cause other SIP elements in the untrusted network to send requests directly to the SIP proxy. Otherwise, the requests are sent to the home address. However, responses sent by the SIP proxy will have the SIP proxy's source address, rather than the home address. Some SIP elements might drop responses that come from a IP address different from the one to which the request is sent.

About Headers

You can specify which SIP headers you want effected by the SIP NAT function. The URIs in these headers are translated and encrypted, the encryption occurs according to the rules of this SIP NAT function.

You can enter header values by using either the full header name or its corresponding abbreviation, if applicable. The following table lists the available headers and their corresponding abbreviations.

Header	Abbreviation
Call-ID	i
Contact	m
From	f
Record-Route	none
Route	none
Ready-To	none
Replaces	none
Refer-To	r
To	t
Via	v

SIP sessions are terminated and re-originated as new sessions as they are routed through the Oracle Communications Session Border Controller. Among the actions performed, SIP headers are modified to prevent the transmission of IP address and route information.

Replacing Headers

In the SIP signaling message, any Via headers are stripped out and a new one is constructed with the Oracle Communications Session Border Controller's IP address in the sent-by portion.

If a Contact header is present, it is replaced with one that has the Oracle Communications Session Border Controller's IP address. All other headers are subject to NATing based on the following rules:

- The Request-URI is replaced with the next hop's IP or FQDN address.
- All other headers are replaced based on the two SIP NAT function SIP NAT function rules

Mapping FQDNs

The Oracle Communications Session Border Controller maps FQDNs that appear in the certain headers of incoming SIP messages to the IP address that the SBC inserts in outgoing SIP contact headers. The mapped FQDNs are restored in the SIP headers in messages that are sent back to the originator.

This feature is useful to carriers that use IP addresses in the SIP From address to create trunk groups in a softswitch for routing purposes. When the carrier's peer uses FQDNs, the carrier is forced to create trunk groups for each possible FQDN that it might receive from a given peer. Similarly, this can apply to SIP Contact and P-Asserted-Identity headers.

SIP NAT Function Cookies

Cookies are inserted to hide that information is coming from a realm external to the home realm. They are used when information needs to be placed into a given element of a SIP message that must also be seen in subsequent SIP messages within a flow. When forwarding a SIP message, the Oracle Communications Session Border Controller (SBC) encodes various information in the outgoing message, which is passed from one side to another in SIP transactions.

SIP NAT function cookies let the SBC hide headers, IPv4 addresses, and SIP URIs. These cookies are included when certain conditions are present in SBC SIP transactions.

Oracle's SIP NAT function cookies can be used in the userinfo, host, URL parameter, and tel URL parameter portions of the SIP message.

userinfo

The Oracle Communications Session Border Controller places a cookie in the userinfo portion of a SIP URI when a SIP header contains a SIP URI, and includes that header type in the list of headers to be hidden (encrypted) in the associated SIP NAT function. The cookie for the userinfo portion is the following:

```
[user nat tag][encrypted 13-byte host IP][encrypted 13 byte maddr IP (if
present)]
```

where:

- [user nat tag] refers to the SIP NAT function's original user NAT tag field.
- [encrypted 13-byte host IP] refers to the host IP encryption.
- [encrypted 13 byte maddr IP (if present)] refers to the maddr IP encryption, if it exists.

With a user NAT tag of -acme, the following SIP-URI:

```
sip:6175551212@192.168.1.100
```

might be translated into:

```
sip:6175551212-acme-pfi1s7n2pstna@172.16.1.10
```

> **Note:**
>
> Multiple additional cookies might be appended with each hop (for example, from the external proxy to the home proxy and back).

host

When hiding IP addresses in a SIP message, the SIP NAT function generates the following cookie for a SIP-URI with no userinfo portion:

```
[host nat tag][encrypted 13-byte host IP][encrypted 13 byte maddr IP (if
present)][domain suffix]
```

where:

- [host nat tag] refers to the SIP NAT function's host NAT tag.
- [encrypted 13-byte host IP] refers to the host IP encryption.
- [encrypted 13 byte maddr IP (if present)] refers to the maddr IP encryption, if it exists.
- [domain suffix] refers to the SIP NAT function's domain suffix field.

With a SIP NAT function's host tag of ACME- and a domain suffix of .acme.com, the following SIP header:

Via: SIP/2.0/UDP 192.168.1.100:5060

might be translated into the following:

```
Via: SIP/2.0/UDP ACME-pfi1s7n2pstna.acme.com
```

URL Parameter

If the SIP NAT function's use url parameter field has a value of from-to or all, the SIP NAT function places all cookies generated to hide SIP URIs in a custom tag appended to the header. Setting the use url parameter field to:

- from-to only affects the behavior of the SIP NAT function's cookies in the From and To headers.
- all affects all SIP headers processed by the SIP NAT function

The cookie is the following:

[;url-parameter]=[host nat tag][encrypted 13-byte host IP][encrypted 13-byte maddr IP]

where:

- [;url-parameter] refers to the SIP NAT function's parameter name field.
  This cookie type is associated with the all and from-to field value options of the SIP NAT function's use url parameter field.

- [host nat tag] refers to the SIP NAT function's host NAT tag field.
- [encrypted 13-byte host IP] refers to the host IP encryption.
- [encrypted 13 byte maddr IP (if present)] refers to the maddr IP encryption, if it exists.

With a host NAT tag of ACME- and a parameter name of acme_param, the following SIP-URI:

```
sip:6175551212@192.168.1.100
```

might be translated into the following:

```
sip:6175551212@172.16.1.10;acme_param=ACME-pfi1s7n2pstna.
```

tel URL

The SIP NAT function cookie is used when devices in your network are strict about the context portion of SIP messages regarding the conversion of tel URLs. This cookie for the tel URL parameter portion of a SIP message is the following:

```
tel URL parameter-[13-byte host IP][13 byte optional maddr IP]domain suffix
```

where:

- tel URL parameter refers to the SIP NAT function's use url parameter.
  This cookie type is associated with the use url parameter's phone field value for the SIP NAT.
- [13-byte host IP] refers to the host IP encryption.
- [13 byte optional maddr IP] refers to the maddr IP encryption, if it exists.
- domain suffix refers to the SIP NAT function's domain suffix field.

Configuration Overview

Configuring the SIP NAT function falls into two areas, the SIP NAT interface parameters and the SIP NAT policies.

SIP NAT Interface

The following tables lists the SIP NAT function interface parameters you need to configure on the Oracle Communications Session Border Controller (SBC).

Parameter	Description
realm ID	Name of the external realm. The realm ID must be unique; no two SIP NATs can have the same realm ID. This realm ID must also correspond to a valid realm identifier entered when you configured the realm.
external proxy address	IPv4 address of the SIP element (for example, a SIP proxy) in the external network with which the SBC communicates. Entries must follow the IP address format.
external proxy port	UDP/TCP port of the SIP element (for example, a SIP proxy) in the external network with which the SBC communicates. Minimum value is 1025, and maximum value is 65535. Default is 5060.

Parameter	Description
external address	IPv4 address on the media interface in the external realm. Enter a value that ensures any packet with an external address value as its destination address is routed to the SBC through the media interface connected to or routable from the external realm. Entries must follow the IP address format. To specify whether the external realm referenced in this field is private or public, configure the SIP config's NAT mode.
home address	IPv4 address on the media interface in the home realm. Enter a value that ensures any packet with a home address value as its destination address must be routed to the SBC through the media interface connected to or routable from the home realm. Entries must follow the IP address format. The value entered in this field must be different from the IP address value of the home realm's network interface element. The home realm network interface is associated with this SIP NAT by its realm ID and the realm's identifier and network interface value you entered when you configured the realm. The realm's network interface identifier value corresponds to this SIP NAT's realm ID, the SIP config's home realm ID, and the media manager's home realm ID.
home proxy address	Sets the IP address for the home proxy (from the perspective of the external realm). By default, this field is empty. An empty home proxy address field value signifies that there is no home proxy, and the external address will translate to the address of the SBC 's SIP proxy. Entries must follow the IP address format.
home proxy port	Sets the port number for the home realm proxy. Value can be set to zero (0). Minimum is 1025 and maximum is 65535. Default is 5060.
route home proxy	Whether to route all inbound requests for the SIP NAT to the home proxy. enabled adds route if Request-URI is not the SBC disabled does not route inbound requests to the home proxy forced always adds route

SIP NAT Function Policies

The following tables lists the SIP NAT function policy parameters you need to configure on the Oracle Communications Session Border Controller (SBC).

Parameter	Description
domain suffix	Domain name suffix of the external realm. The domain name suffix refers to and must conform to the hostname part of a URI. In combination with the user NAT tag and host NAT tag values, this value is used to help the SBC identify an encoded URI that it needs to translate when moving between public and private realms. This suffix is appended to encoded hostnames that the SIP NAT function creates. For example, if the encoded hostname is ACME-abc123 and the domain-suffix value is .netnetsystem.com, the resulting FQDN will be ACME-abc123.netnetsystem.com.
address prefix	Defines which IPv4 address prefixes from incoming messages require SIP-NAT encoding (regardless of the realm from which these messages came).

Parameter	Description
tunnel redirect	Controls whether Contact headers in a 3xx Response message received by the SBC are NATed when sent to the initiator of the SIP INVITE message.
use url parameter	Establishes whether SIP headers will use the URL parameter entered in the parameter name for encoded addresses that the SIP NAT function creates. Also, if SIP headers will be used, which type of headers will use the URL parameter. For example, all headers or just the From and To headers. Enumeration field.
parameter name	Indicates the name of the URL parameter when use url applies. This field value will be used in SIP NAT encoding addresses that have a use url parameter value of either from-to or all.
user NAT tag	Identifies the prefix used when an address is encoded into the username portion of user@host;name=xxxx; where name = parameter name. The user NAT tag values can consist of any characters that are valid for the userinfo part of a URI. In combination with the domain suffix and host NAT tag field values, this value is used to help the SBC identify an encoded URI that it needs to translate when moving between public and private realms.
host NAT tag	Identifies the prefix used when encoding an address into the hostname part of the URI or into a URL parameter. The host NAT tag values refer to domain labels and can consist of any characters that are valid for the hostname part of a URI. In combination with the domain suffix and user NAT tag values, this value is used to help the SBC identify an encoded URI that it needs to translate when moving between public and private realms.
headers	Lists the SIP headers to be affected by the SBC SIP NAT function. The URIs in these headers will be translated and encrypted, and encryption will occur according to the rules of this SIP NAT.

SIP NAT Function Configuration

To configure the SIP NAT function on an Oracle Communications Session Border Controller (SBC):

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. **realm-ID**—Enter the name of the realm you want to identify as the external realm.

   The name you use as the realm ID must be unique. No two SIP NAT functions can have the same realm ID. Also, this value must correspond to a valid identifier entry already configured for the realm.

5. **domain-suffix**—Enter the domain suffix to identify the domain name suffix of the external realm. The domain suffix must begin with a (.) dot.

   The domain name suffix refers to and must conform to the hostname part of a URI. For example:

   ```
   .netnetsystem.com
   ```

   The domain suffix is appended to encoded hostnames that the SIP NAT function creates. For example, if the encoded hostname is ACME-abc123, the resulting FQDN is ACME-abc123.netnetsystem.com.

6. **external-proxy-address**—Enter the external proxy address to identify the IPv4 address of the SIP element (for example, a SIP proxy) in the external network with which the SBC communicates.

   Enter the value in the IP address format. For example:

   ```
   192.168.11.200
   ```

7. **external-proxy-port**—Enter the external proxy port value to identify the UDP/TCP port of the SIP element (for example, a SIP proxy) in the external network with which the SBC communicates. The default is **5060**. The valid range is:

   - Minimum—1025
   - Maximum—65535

8. **external-address**—Enter the external address, which is an IPv4 address on the media interface in the external realm.

   Enter the value in the IP address format. For example:

   ```
   192.168.11.101
   ```

   This value must be such that any packet with an external address value as its destination address is routed to the SBC through the media interface connected to or routable from the external realm.

9. **home-address**—Enter the home address, which is an IPv4 address on the network interface in the home realm. This value must be such that any packet with a home address value as its destination address must be routed to the SBC through the media interface connected to or routable from the home realm.

   Enter the value in the IP address format. For example:

   ```
   127.0.0.10
   ```

   The value entered in this field **must be different** from the IP address value of the home realm's network interface element.

   The home realm network interface is associated with this SIP NAT by its realm ID and the realm's identifier and network interface value you entered when you configured the realm. The realm's network interface identifier value corresponds to this SIP NAT's realm ID, the SIP config's home realm ID, and the media manager's home realm ID.

10. **home-proxy-address**—Enter the home proxy address to set the IP address for the home proxy (from the perspective of the external realm).

By default, this field is empty. No home proxy address entry signifies there is no home proxy, and the external address will translate to the address of the SBC's SIP proxy.

Enter the value in the IP address format. For example:

```
127.1.0.10
```

11. **home-proxy-port**—Enter the home proxy port to set the port number for the home realm proxy. The default value is **0**. The valid range is:
    - Minimum—0, 1025
    - Maximum—65535
12. **route-home-proxy**—Optional. Enable or disable requests being routed from a given SIP-NAT to the home proxy. The default value is **disabled**. The valid values are:
    - **enabled**—All inbound requests for a specific SIP NAT are routed to the home proxy
    - **disabled**—All inbound requests are not routed through the home proxy.
    - **forced**—The Request is forwarded to the home proxy without using a local policy.
13. **address-prefix**—Optional. Indicate the IPv4 address prefix from incoming messages that requires SIP NAT function encoding (regardless of the realm from which these messages came).

    > **Note:**
    >
    > This value overrides the value set in the realm's address prefix field.

    This field's format incorporates an IPv4 address and number of bits in the network portion of the address. For example, a Class C address has a 24-bit network part. The address prefix for 101.102.103.x would be represented as 10.102.103.0/24.

    The default value is an asterisk (*****). When you enter this value or do not enter a value, the realm's address prefix value is used.
14. **tunnel-redirect**—Set to one of the following values to indicate whether certain headers in a 3xx Response message received by the SBC are NATed when sent to the initiator of the SIP INVITE message. The default is **disabled**. The valid values are:
    - **enabled**—Certain headers in a 3xx Response message are NATed.
    - **disabled**—Certain headers in a 3xx Response message are not NATed.
15. **use-url-parameter**—Establish whether SIP headers will use the URL parameter (configured in the next step) for encoded addresses created by the SIP NAT function. If SIP headers will be used, this value identifies which types of headers will use the URL parameter. The default value is **none**. The available values include:
    - **none**—No headers will use the URL parameter for address encoding.

      The following example illustrates the functionality of an SBC using a use url parameter value of none:

      ```
      sip: 1234@1.2.3.4 is translated into sip: 1234-acme-xxxx@5.6.7.8
      ```

      where -acme-xxxx is a cookie and xxxx is the encoded version of 1.2.3.4.
    - **from-to**—From and To headers will use the URL parameter for address encoding

The following example illustrates the functionality of a SBC using a use url parameter value of none:

```
sip: 1234@1.2.3.4 is translated into sip: 1234@5.6.7.8; pn=acme-xxxx
```

where -acme-xxxx is a cookie and xxxx is the encoded version of 1.2.3.4.

- **all**—All headers will use the URL parameter for address encoding. Oracle recommends not using this values because other SIP elements or implementations (other than the Oracle Communications Session Border Controller) might not retain the URL parameter in subsequent SIP messages that they send to the Oracle Communications Session Border Controller.
- **phone**—

  If this field is set to either from-to or all, the SBC puts the encoded address of the SIP NAT into a URL parameter instead of using the encoding name inside the userinfo part of the address.

16. **parameter-name**—If you have configured the **use-url-parameter** with the from-to or all value, you need to indicate the hostname prefix.

    The parameter name value is used in SIP NAT encoding addresses that have the use url parameter values of from-to or all.

17. **user-NAT-tag**—Enter a value to identify the username prefix used for SIP URIs. The values you can use can include any characters valid for the userinfo part of a URI. This should be made unique for each realm and SIP NAT function.

    The default value is **-acme-**.

    In combination with the domain suffix and host NAT tag values, this value is used to help the SBC identify an encoded URI that it needs to translate when moving between public and private realms.

18. **host-NAT-tag**—Enter a value for the host NAT tag field to identify the hostname prefix used for SIP URIs. The value refers to domain labels and can include any characters valid for the hostname part of the URI. This should be made unique for each realm and SIP NAT function.

    The default value is **ACME-**.

    In combination with the domain suffix and user NAT tag values, this value is used to help the SBC identify an encoded URI that it needs to translate when moving between public and private realms.

19. **headers**—List the SIP headers you want affected by the SIP NAT function. The URIs in these headers are translated and encrypted, and encryption occurs according to the SIP NAT function rules.

    To enter the full default list, type headers, followed by a Space and -d, then press Enter.

    You can also insert the following tags in SIP NAT headers if you want to replace FQDNs with next hop or SIP interface IP addresses:

    - fqdn-ip-tgt: replaces the FQDN with the target address
    - fqdn-ip-ext: replaces the FQDN with the SIP NAT external address

      Enter the tag using the following format:

      <header-name>=<tag>

      For example:

To=fqdn-ip-tgt

The FQDN in a To header is replaced with the target IP address.

You can insert the following tags to apply NAT treatment to a From header in an INVITE when the gateway sends it into the home realm.

- ip-ip-tgt: replaces any IP address in the From header with the next hop target
- ip-ip-ext: replaces any IP address in the From header with the SBC's external address

To view all SIP NAT function parameters, enter a **?** at the system prompt. The following example shows SIP NAT configuration for peering network.

```
sip-nat
        realm-id                        peer-1
        domain-suffix                   .p1.acme.com
        ext-proxy-address               192.168.11.200
        ext-proxy-port                  5060
        ext-address                     192.168.11.101
        home-address                    127.0.0.10
        home-proxy-address              127.1.0.10
        home-proxy-port                 5060
        route-home-proxy                enabled
        address-prefix                  *
        tunnel-redirect                 disabled
        use-url-parameter               none
        parameter-name
        user-nat-tag                    -p1-
        host-nat-tag                    P1-
        headers                         Call-ID Contact From Join Record-
Route
                                        Refer-To Replaces Reply-To Route

To Via
                                        f i m r t v
```

SIP Realm Bridging

This section explains how to configure the internal routing among realms known as realm bridging. Realm bridging lets you cross-connect SIP interfaces. You can use one of the following two methods for bridging realms:

- local policy bridging: use this method to enable dynamic internal routing between realms if your SIP interfaces do not have the SIP NAT function applied.
- SIP NAT bridging: use this method if your SIP interfaces have the SIP NAT function applied.

About SIP NAT Bridging

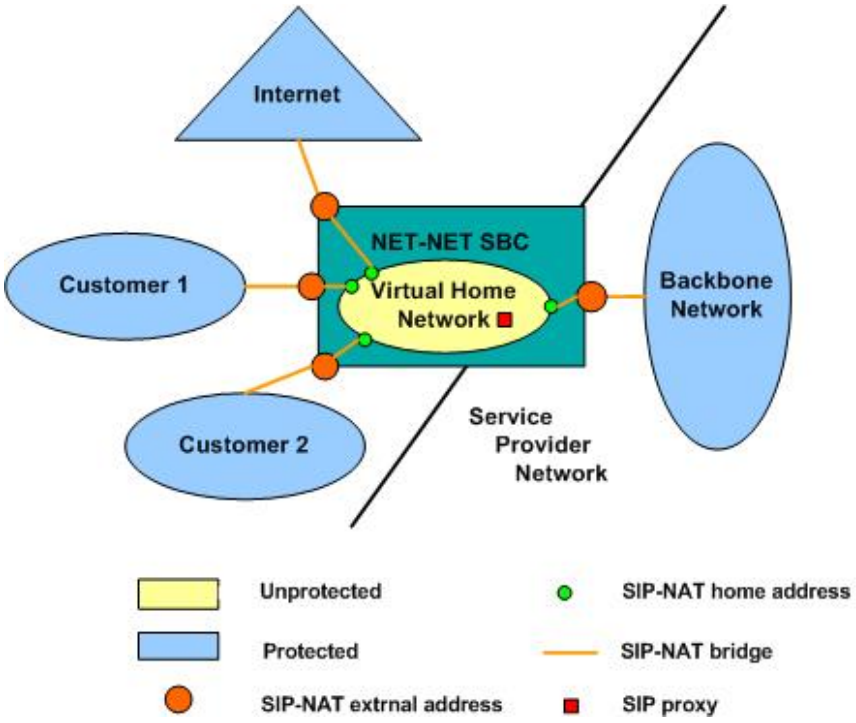
Each SIP NAT has a presence in two realms, trusted and untrusted. The SIP NAT bridge is the conduit for packages in and out of the home realm. It creates a bridge between realms by providing address translations; removing all references to the original IP addressing from the packets sent to the destination network.

With the SIP NAT bridge, an untrusted (or public) home network can reside within the Oracle Communications Session Border Controller, while the other entities (the backbone network, the Internet, or customer networks) are all trusted (or private). One of the primary functions of the

SIP NAT bridge is to protect networks from one another so that address bases can remain hidden. Using a SIP NAT bridge, no one network has direct access to the data of other networks.

Establishing a SIP NAT bridge lets you route every SIP Request message through the backbone. Without using this functionality, it would appear as though all messages/sessions were coming from the Oracle Communications Session Border Controller's SIP proxy (the SIP server that receives SIP requests and forwards them on behalf of the requestor).

The following diagram illustrates this unprotected (or public) and protected (or private) division.



SIP NAT Bridge Configuration Scenarios

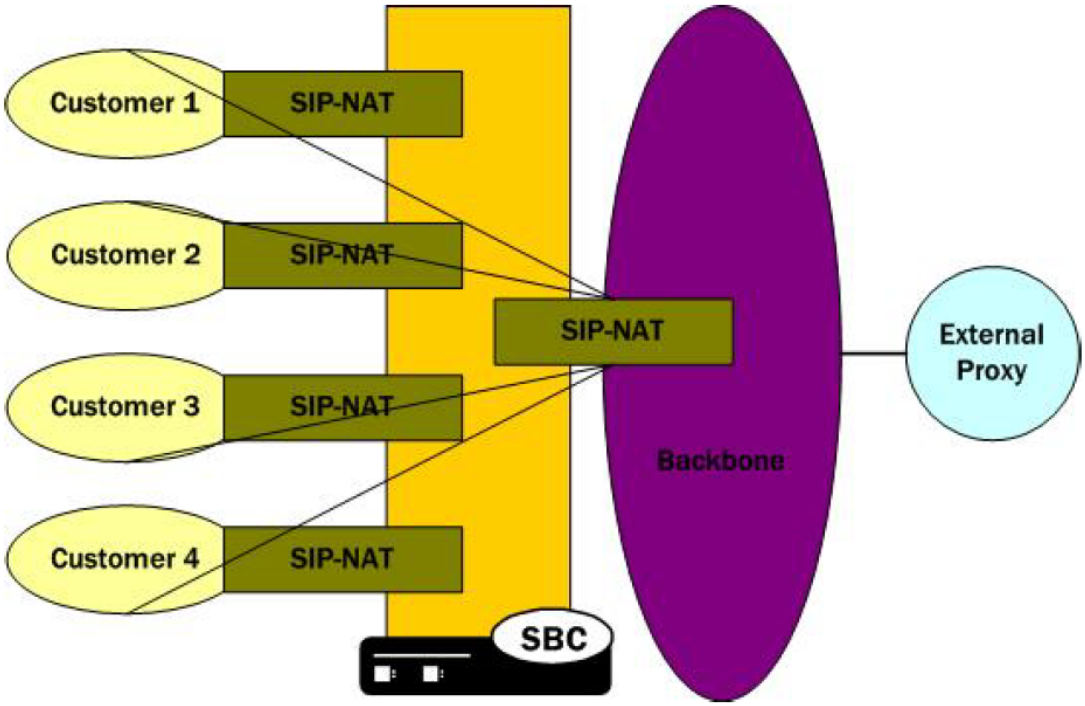
You can configure the SIP NAT bridge functionality in a many-to-one or a one-to-one relationship. For example, multiple customer SIP NATs can be tied to a single backbone SIP NAT, or a single customer SIP NAT can be tied to a single backbone SIP NAT.

You might need to use several SIP NATs on the customer side while using only one on the backbone side in a many-to-one relationship. Or you might configure one SIP NAT on the backbone side for every one that you configure on the customer side in a one-to-one relationship.

You can route all customer side SIP NAT requests to the corresponding backbone SIP NAT regardless of the Request URI. If a request arrives from the customer network with a Request URI that does not match the customer SIP NAT external address or the local policy that would route it to the backbone SIP NAT; the route home proxy value is used.

Many to One Configuration

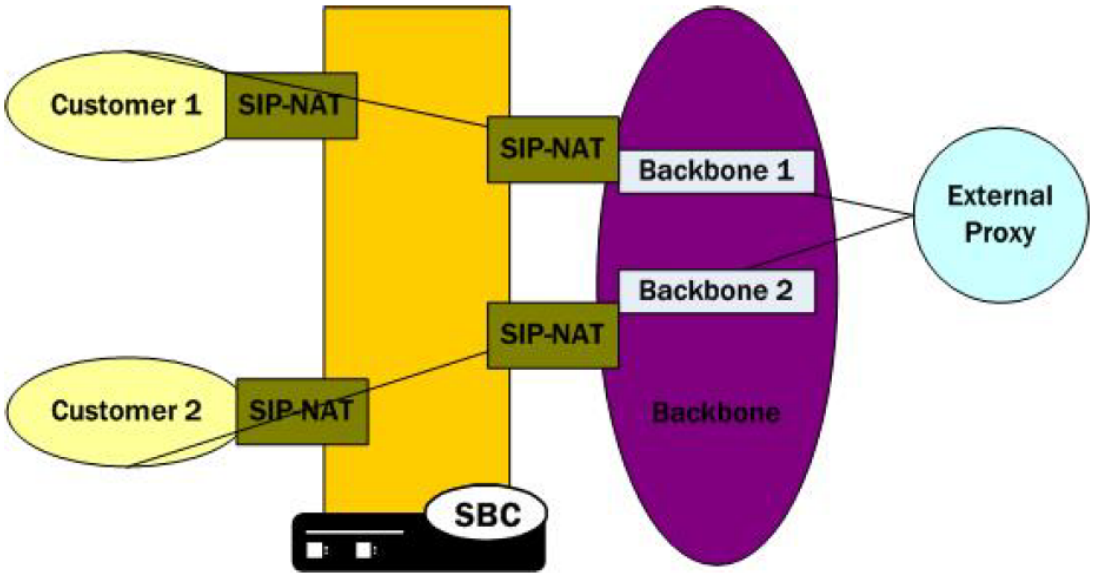
In the many-to-one scenario, multiple customer SIP NATs are tied to a single backbone SIP NAT. The following diagram illustrates the many-to-one SIP NAT bridge configuration.



One-to-One Configuration

In the one-to-one scenario, a single customer SIP NAT is tied to a single backbone SIP NAT. On the backbone SIP NAT side, you configure the home proxy address to match the home address of the customer SIP NAT. On the customer side, you configure the home proxy address to match the home address of the backbone SIP NAT.

The following diagram illustrates the one-to-one SIP-NAT bridge configuration.



SIP NAT Bridge Configuration

You create a bridge between SIP NATs by pointing them at one another. You point the SIP NATs at each other by configuring the home address and home proxy address to create the bridge. In addition, you can configure the route home proxy on the customer's side of a SIP NAT to force all requests to be routed to the corresponding backbone SIP NAT, regardless of the Request URI. You need to force requests when elements in the customer's network send requests with a Request URI that does not match the customer's SIP NAT external address. Or when the Request URI does not match a local policy element that would route the requests to the backbone SIP NAT.

You also need a home network to create a SIP NAT bridge. If you do not have a real home network, you need to create a virtual one. You also need to configure instances of the SIP NAT to create the SIP NAT bridge within your network.

Creating a Virtual Home Network

A virtual home network is a home network that resides entirely within the Oracle Communications Session Border Controller, as does a real home network. The difference between the two is the real home network also has a physical connection to the Oracle Communications Session Border Controller.

The internal home realm/network is usually configured with addresses within the special loopback range (127.0.0.0/8) as described in RFC 3330. This applies to the SIP port addresses for the home realm's SIP interface, and all home addresses for SIP NATs. The address 127.0.0.1 should not be used because it conflicts with the default loopback interface setup by the system for inter-process communication.

To create a virtual home network:

1. Set the name and subport ID of the network interface associated with the home realm element to lo0:0.
2. To enable the SIP proxy to listen for messages on the virtual home realm, configure the home realm ID. It must correspond to the realm's identifier, in which you set the network interface subelement to point to the appropriate network interface element.

   The following table lists the field values you need to set when you are using SIP NAT bridge functionality and you do not have a real home network.

Configuration Element	Configuration Parameter	Sample Values
realm configuration	identifier	home
N/A	network interfaces	lo0:0
N/A	address prefix	127.0.0.0/8
SIP configuration	home realm ID	home
N/A	SIP ports address	127.0.0.100

Many-to-One Configuration

To configure many-to-one:

1. For the backbone SIP NAT, ensure the home proxy address field is blank.
2. For the customer side SIP NAT:

Set the home address to match the home address of the customer.

Set the home proxy address to match the backbone SIP NAT home address.

Set route home proxy to forced.

The following table lists the field values you need to set to create a many-to-one SIP NAT bridge.

SIP NAT Entity	Field	Sample Values
Backbone SIP NAT	home address	IPv4 address of the home realm. For example: 127.0.0.120
N/A	home proxy address	IPv4 address of the home proxy from the perspective of the external realm. For a backbone SIP NAT, leave blank.
Customer SIP NAT	home address	127.0.0.120
N/A	home proxy address	127.0.0.110
N/A	route home proxy	forced

One-to-One Configuration

In the one-to-one scenario, a single customer SIP NAT is tied to a single backbone SIP NAT. The home proxy address field value of the backbone SIP NAT must match the home address of the customer SIP NAT. On the customer side, the home address of the customer SIP NAT should be defined as the home address of the customer, the home proxy address field value should match the home address of the backbone SIP NAT, and route home proxy should be set to forced.

The following table lists the field values you need to set to create a one-to-one SIP NAT bridge.

SIP NAT Entity	Field	Sample Values
Backbone SIP NAT	home address	IPv4 address of the home realm. For example: 127.0.0.110
N/A	home proxy address	IPv4 address of the home proxy from the perspective of the external realm. 127.0.0.120
Customer SIP NAT	home address	127.0.0.120
N/A	home proxy address	127.0.0.110
N/A	route home proxy	forced

Shared Session Agent

Usually, the same set of servers (the external proxy) is used for all SIP NATs to the backbone network. In order to support redundant servers in the backbone of a SIP NAT bridge, the original egress realm as determined by the incoming Request URI needs to be retained after a local policy lookup.

When a request arrives at the Oracle Communications Session Border Controller, it determines the matching (target) session agent and, after the local policy is examined, sets the new outbound session agent to the one from the selected target.

If the target session agent's realm is set to *, the Oracle Communications Session Border Controller retains the original session agent's realm ID. Because the target session agent does not have a realm ID defined, the original egress realm is retained.

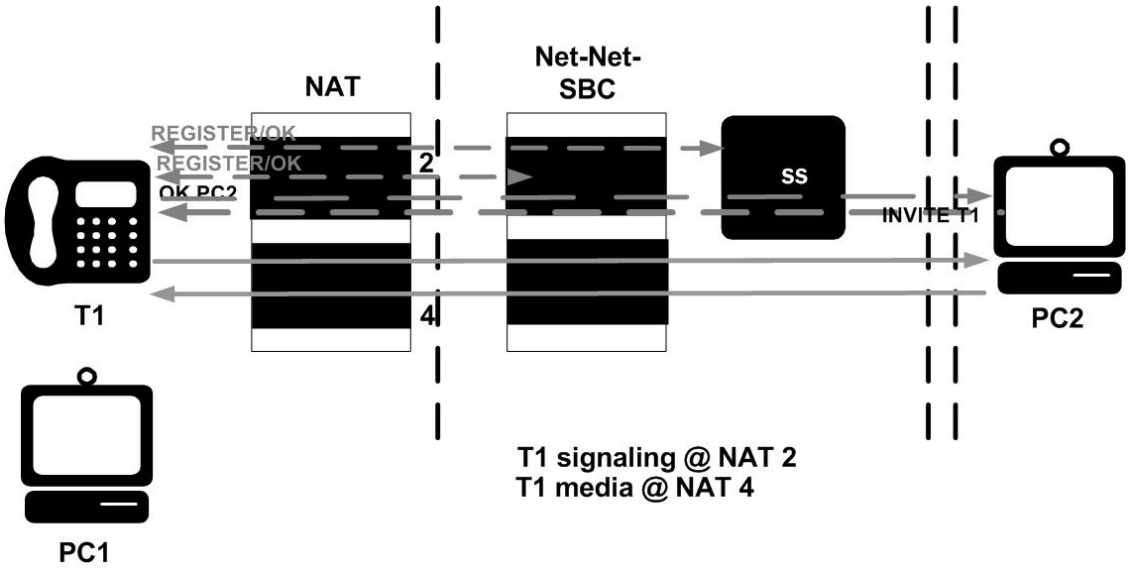
SIP Hosted NAT Traversal (HNT)

This section explains how to configure SIP Hosted Network Address Translation (HNT) traversal. SIP HNT lets endpoints behind a NAT/firewall device send and receive signaling and media using the Oracle Communications Session Border Controller as a relay.

About SIP HNT

SIP HNT is a technique the Oracle Communications Session Border Controller uses to provide persistent reachability for SIP UAs located in private Local Area Networks (LANs) behind Nat/firewall devices. It relies on frequent, persistent messaging to ensure that the binding on the intermediary NAT device is not torn down because of inactivity. HNT does not require support for the NAT in the SIP endpoint.

The following diagram illustrates SIP HNT traversal.



The SBC's HNT function allows endpoints located behind NATs to communicate; providing means to traverse NATs. The SBC interacts with endpoints (using SIP) to allow persistent inbound and outbound signaling and media communications through these NATs.

The SBC automatically detects when an intermediate NAT exists between the UA and the SBC by comparing the Layer 3 IP address of a REGISTER message with the IP address indicated within the UA. The SBC sends signaling responses to the address and port that the request came from, rather than the address and port indicated in the request. The Via header in the request message indicates where the response should be sent.

Using HNT with Existing NAT Device

For network architectures in which premise devices and endpoints reside behind an existing NAT device, the Oracle Communications Session Border Controller's HNT function allows these premise NATs to be traversed without requiring an upgrade to the premise equipment,

the deployment and management of additional premise-based hardware or software, or any NAT device configuration changes.

Registering Endpoints

The Oracle Communications Session Border Controller uses periodic endpoint registration messages to dynamically establish and maintain bindings in the NAT. These bindings keep a signaling port (port that is opened on a firewall to allow traffic to pass through it is a pinhole) open in the NAT that allows the inbound signaled communications to pass through. Using the endpoint registrations, the Oracle Communications Session Border Controller then maps the Layer 3 (OSI network layer that deals with switching and routing technologies for data transmission between network devices) IPv4 address/port information from the NAT device to the Layer 5 (OSI session layer that deals with session and connection coordination between applications) entity (for example, user name or phone number) behind the NAT so that when an incoming signaling message is received, the Oracle Communications Session Border Controller sends it to the appropriate address and port on the NAT for the called party.

Establishing Media Flows

During call setup, the ports for bidirectional media flows are established dynamically. Since the media flows also pass through the Oracle Communications Session Border Controller, it can identify the IPv4 address/port information on the NAT device used for the outgoing media coming from the user name/phone number. The Oracle Communications Session Border Controller then uses that same NAT's IPv4 address/port information to send incoming media to the correct user name/phone number behind the NAT device.

Prerequisites

In order to achieve HNT, the endpoints involved must be capable of:

- symmetric signaling: sending and receiving SIP messages from the same transport address (IP address or User Datagram Protocol/Transmission Control Protocol (UDP/TCP) port
- symmetric media: sending and receiving Real-Time Transport Protocol (RTP) messages from the same UDP port

These conditions are required to allow signaling and media packets back through the NAT (through the bound external address and port). These packets must come from the address and port to which the outbound packet that created the NAT binding was sent. The NAT sends these inbound packets to the source address and port of the original outbound packet.

When SIP HNT is used, the Oracle Communications Session Border Controller sends signaling responses to the address and port that the request came from rather than the address and port indicated in the request. The Via header in the request message indicates where the response should be sent.

Keeping the NAT Binding Open

Additional measures are also required to keep the NAT binding open because most NAT bindings are discarded after approximately a minute of inactivity. The SBC keeps the SIP NAT binding open by returning a short expiration time in REGISTER responses that forces the endpoint to send frequent REGISTER requests.

In order to keep the NAT binding open for SIP, the SBC maintains the registration state. When an endpoint first registers, the SBC forwards that REGISTER message on to the real registrar. You can define the real registrar using either of the following methods:

- Configure the SIP config registrar host and registrar port to indicate the real registrar.
- Map the SIP config registrar host and registrar port values to the SIP NAT home proxy address and home proxy port values. Then configure the SIP NAT's external proxy address and external proxy port values to correspond to the real registrar.

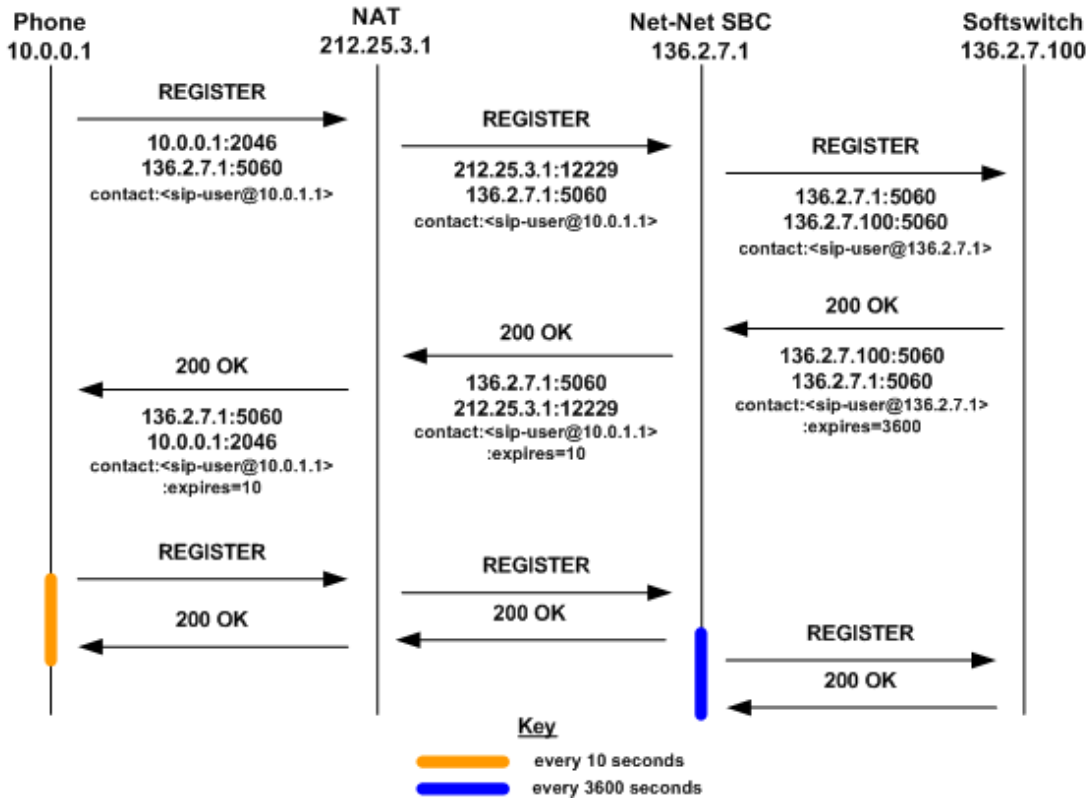
> **Note:**
>
> A registrar can be located in a SIP NAT realm.

When a successful response is received, the SBC caches the registration to memory. This cached registration lives for the length of time indicated by the expiration period defined in the REGISTER response message from the registrar. The response sent back to the endpoint has a shorter expiration time (defined by the SIP config's NAT interval) that causes the endpoint to send another REGISTER message within that interval. If the endpoint sends another REGISTER message before the cached registration expires, the SBC responds directly to the endpoint. It does not forward the message to the real registrar.

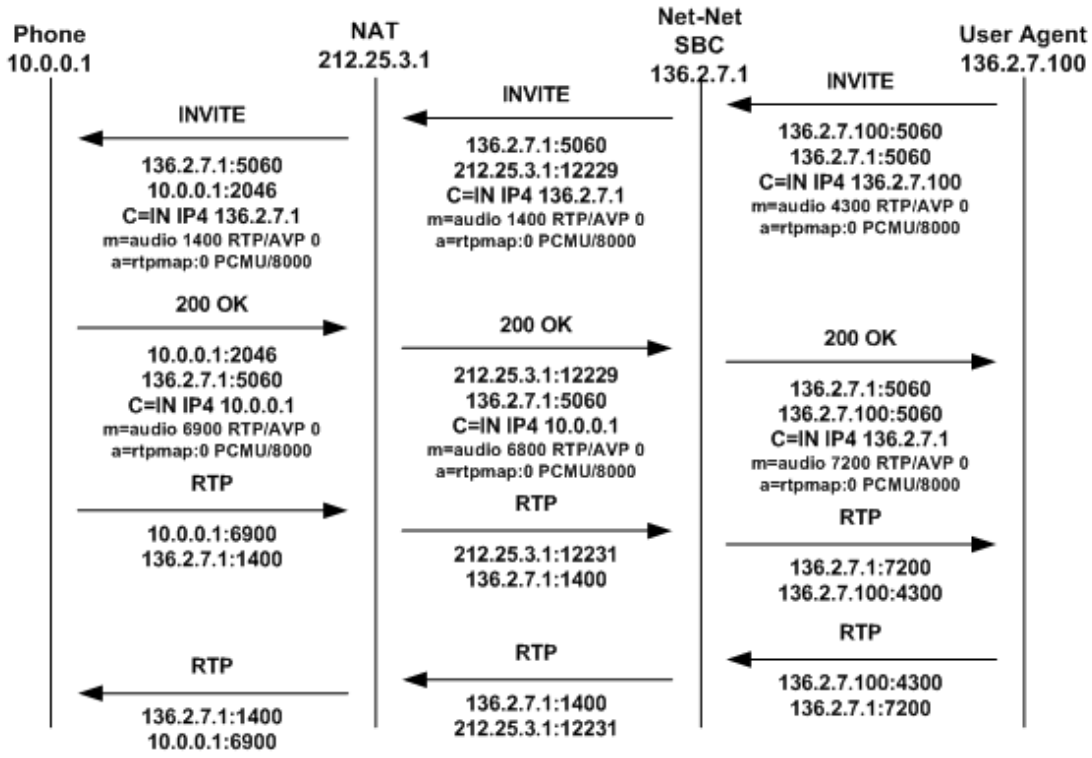
If the cached registration expires within the length of time indicated by the NAT interval, the REGISTER message is forwarded to the real registrar. If the SBC does not receive another REGISTER message from the endpoint within the length of time indicated by the NAT interval, it discards the cached registration.

The Contact Uniform Resource Identifier (URI) in the REGISTER message sent to the registrar by the SBC points at the SBC so that the proxy associated with the real registrar sends inbound requests to the SBC. This way, the inbound requests can be forwarded to the endpoint through the NAT binding.

The following example illustrates the SIP HNT registration call flow for the SIP HNT feature.



The following example illustrates the SIP HNT invitation call flow for the SIP HNT feature.



Working with Multiple Domains

You can use a wildcard (*) with the HNT feature to accommodate multiple domains and to allow the Oracle Communications Session Border Controller to cache all HNT endpoints. The wildcard functionality is enabled in the SIP config by entering an asterisk (*) in the registrar domain and registrar host fields.

The wildcard allows the use of either a local policy or Domain Name Service (DNS) to resolve the domain name to the correct registrar. Either method can be used to route the Fully Qualified Domain Name (FQDN) when the you enter an asterisk (*) for the register host. An FQDN consists of an unlimited number of domain labels (domain names), each separated by a dot (.). The FQDN can include the top level domain name (for example, acmepacket.com).

In the hostname acme-packet.domainlbl.example100.com, the syntax is as follows:

- acme-packet is a domain label
- domainlbl is a domain label
- example100 is a domain label
- com is the top label

The information configured in a local policy is used before DNS is used. If the next hop destination address (defined in the local policy's next hop field) is an IPv4 address, a DNS server is not needed. A DNS server is needed when the IPv4 address of the next hop destination address is a FQDN or cannot be determined from the Oracle Communications Session Border Controller's configuration. Even with a configured local policy, the next hop destination address might be an FQDN that requires a DNS lookup.

If the registrar host does not use the wildcard, the Oracle Communications Session Border Controller always uses the configured address. You can limit the number of endpoints that

receive the HNT function. For example, you can use a non-wildcarded registrar domain field value (like acme.com) with a wildcarded registrar host field value.

HNT Configuration Overview

To configure SIP HNT NAT traversal, you need to configure both the SIP interface and the SIP config.

SIP HNT Single Domain Example

The following example shows values entered for the SIP config and SIP interface elements to configure SIP HNT for a single domain and registrar.

- SIP config

Parameter	Sample Value
registrar domain	netnetsystem.com
registrar host	192.168.12.1
registrar port	5060

- SIP interface

Parameter	Sample Value
NAT traversal	always
NAT interval	60
minimum registration expire	200
registration caching	disabled
route to registrar	enabled

SIP HNT Multiple Domain Example

The following example shows values entered for the SIP config and SIP interface elements to configure SIP HNT for a multiple domains and multiple registrars.

- SIP config

Parameter	Sample Value
registrar domain	*
registrar host	*
registrar port	0

- SIP interface

Parameter	Sample Value
NAT traversal	always
NAT interval	60
minimum registration expire	200
registration caching	disabled
route to registrar	enabled

HNT Configuration

To configure a SIP interface on the Oracle Communications Session Border Controller (SBC):

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can configure sip-interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. **nat-traversal**—Define the type of HNT enabled for SIP. The default value is **none**. Available values include:
   - **none**—Disables the HNT feature for SIP (default value)
   - **rport**—SIP HNT function only applies to endpoints that include the rport parameter in the Via header and the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address.
   - **always**—SIP HNT applies to requests when the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address. (Even when the rport parameter is not present.)
5. **nat-interval**—Set the expiration time in seconds for the SBC's cached registration entry for an HNT endpoint. The default value is **30**. The valid range is:
   - Minimum—0
   - Maximum—4294967295

     Oracle recommends setting the NAT interval to one-third of the NAT binding lifetime. A NAT binding lifetime is the network connection inactivity timeout. The value is configured (or hardwired) in the NAT device (firewall). This timer is used to prevent the NAT device from keeping an unused port open.
6. **registration-caching**—Enable for use with all UAs, not just those that are behind NATs. By default, this field is set to **disabled**. If enabled, the SBC caches the Contact header in the UA's REGISTER request when it is addressed to one of the following:
   - SBC
   - registrar domain value
   - registrar host value

     The SBC then generates a Contact header with the SBC's address as the host part of the URI and sends the REGISTER to the destination defined by the registrar host value.

Whether or not SIP HNT functionality is enabled affects the value of the user part of the URI sent in the Contact header:

- **enabled**—The SBC takes the user part of the URI in the From header of the request and appends a cookie to make the user unique. A cookie is information that the server stores on the client side of a client-server communication so that the information can be used in the future.
- **disabled**—The user part of the Contact header is taken from the URI in the From header and no cookie is appended. This is the default behavior of the Oracle Communications Session Border Controller.

  When the registrar receives a request that matches the address-of-record (the To header in the REGISTER message), it sends the matching request to the SBC, which is the Contact address. Then, the v forwards the request to the Contact-URI it cached from the original REGISTER message.

7. **min-reg-expire**—Set the time in seconds for the SIP interface. The value you enter here sets the minimum registration expiration time in seconds for HNT registration caching. The default value is **300**. The valid range is:

   - Minimum—1
   - Maximum—999999999

   If, for example, an endpoint sends a REGISTER message with the "Expires:" field set to 60 seconds, then the SBC (when re-originating the REGISTER message to the registrar) changes that value to the min-reg-expire value if the latter is greater. This is because the SBC requires the value of the Expires field to be at least the value of the min-reg-expire parameter. If the value of the Expires field in the original message is equal to or greater than the value of the min-reg-expire parameter, the SBC will not change it.

   This value defines the minimum expiration value the SBC places in each REGISTER message it sends to the real registrar. In HNT, the SBC caches the registration after receiving a response from the real registrar and sets the expiration time to the NAT interval value.

   Some UAs might change the registration expiration value they use in subsequent requests to the value specified in this field. This change causes the SBC to send frequent registrations on to the real registrar.

8. **registration-interval**—Set the SBC's cached registration entry interval for a non-HNT endpoint. Enter the expiration time in seconds that you want the SBC to use in the REGISTER response message sent back to the UA. The UA then refreshes its registration by sending another REGISTER message before that time expires. The default value is **3600**. The valid range is:

   - Minimum—1

     A registration interval of zero causes the SBC to pass back the expiration time set by and returned in the registration response from the registrar.

   - Maximum—999999999

     If the expiration time you set is less than the expiration time set by and returned from the real registrar, the SBC responds to the refresh request directly rather than forwarding it to the registrar.

> **Note:**
>
> With registration caching, there is no NAT; therefore, a short registration interval causes the UA to send excess REGISTER messages.

Although the registration interval applies to non-HNT registration cache entries, and the loosely related NAT interval applies to HNT registration cache entries, you can use the two in combination. Using a combination of the two means you can implement HNT and non-HNT architectures on the same SBC. You can then define a longer interval time in the registration interval field to reduce the network traffic and load caused by excess REGISTER messages because there is no NAT binding to maintain.

9. **route-to-registrar**—Enable routing to the registrar to send all requests that match a cached registration to the destination defined for the registrar host; used when the Request-URI matches the registrar host value or the registrar domain value, not the SBC's address. Because the registrar host is the real registrar, it should send the requests back to the SBC with the SBC's address in the Request-URI. The default value is **disabled**. The valid values are:

   - enabled | disabled

     For example, you should enable routing to the registrar if your network uses a SBC and needs requests to go through its service proxy, which is defined in the registrar host field.

Global SIP Configuration

To configure the SIP configuration:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. **registrar-domain**—Optional. Define the domain to match against the host part of a URI to determine if a request is addressed to the registrar. If there is a match, the registration caching, NAT traversal, and route to registrar parameter values for the SIP interface are applied to the request. By default, this field remains empty. Available values are:

   - an asterisk (*) to specify the values apply to all requests.
   - any alphanumeric character or any combination of alphanumeric characters. For example, acme1.com.

A hostname consists of any number of domain labels, separated by dots (.), and one top label. A top label is the last segment of the hostname. It must start with an alphabetical character. After the first character, a top label can consist of any number or combination of alphanumeric characters, including those separated by dashes. The dash must be preceded and followed by alphanumeric characters. A single alphabetical character is the minimum requirement for a hostname field (for example, c to indicate .com).

When the REGISTER message's Request-URI has an FQDN, it is matched against the registrar domain's value to determine if the message needs to be forwarded to the registrar port on the registrar host. The registrar domain's value is also used when route to registrar is set to enabled, to determine if a request needs to be forwarded to the registrar.

Only the right-hand part of the domain name in the Request-URI needs to match the registrar domain value. For example, acme3.acmepacket.com matches acmepacket.com. However, the entire domain label within the domain name must match. For example, the domain label "acme3.acmepacket.com" would not match packet.com.

5. **registrar-host**—Define the address of the registrar for which requests for registration caching, NAT traversal, and router to registrar options apply. You can use a specific hostname, a IP address, or a wildcard (*):
   - an asterisk (*) indicates normal routing (local policy, DNS resolution, and so on) is used to determine the registrar's address.
   - hostname: can consist of any alphanumeric character or any combination of alphanumeric characters (for example, acme1.com). The hostname can consist of any number of domain labels, separated by dots (.), and one top label. You can use the minimum field value of a single alphabetical character to indicate the top label value (for example, c to indicate .com).
   - IPv4 address: must follow the dotted notation format. Each of the four segments can contain a numerical value between zero (0) and 255. For example, 192.168.201.2. An example of a invalid segment value is 256.

     By default, the registrar host field remains empty.
6. **registrar-port**—Set the SIP registrar port number. The SIP registrar server configured in this and the registrar host field is the real registrar. Or the values entered in those fields map to the home proxy address and home proxy port of the SIP NAT with external proxy address and external proxy port values that correspond to the real registrar. The default value is 0. The valid range is:
   - Minimum—0, 1025
   - Maximum—65535

   The following example shows the values for a single domain and registrar configuration.

```
sip-config
        state                         enabled
        operation-mode                dialog
dialog-transparency       disabled
        home-realm-id                 acme
        egress-realm-id
        nat-mode                      Public
        registrar-domain
        registrar-host
```

```
registrar-port                0
init-timer                    500
max-timer                     4000
trans-expire                  32
invite-expire                 180
inactive-dynamic-conn         32
red-sip-port                  1988
red-max-trans                 10000
red-sync-start-time           5000
red-sync-comp-time            1000
last-modified-date            2005-03-19 12:41:28
```

Endpoint-Initiated Keep-Alives

The Oracle Communications Session Border Controller provides three Keep-Alive algorithms designed to generate sufficient traffic flow to maintain NAT (Network Address Translation) bindings. Prior releases provided a single Keep-Alive algorithm that is described in the SIP Hosted NAT Traversal (HNT) section of the SIP Signaling Services chapter of the *ACLI Configuration Guide*. This method is SD-based and requires no explicit participation by the SIP endpoint. Instead the SD manipulates SIP registration requests and responses to spoof the endpoint — generating frequent and extraneous registration requests that generate a periodic traffic flow to maintain existing NAT bindings.

Unlike the current SD-centric method, the new methods require the active participation of the SIP endpoint. With these methods the SIP endpoint initiates a Keep-Alive negotiation with the SD that produces a periodic request/response message sequence which also generates sufficient traffic flow to maintain NAT bindings.

The new methods are based upon the following RFCs.

RFC 5626, *Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)*, which, in Section 3.5.2, provides a framework for the implementation of an endpoint-initiated Keep-Alive on both TCP and UDP connections and provides two approaches to generating endpoint-initiated Keep-Alive traffic flows. One approach described in Section 4.4.1 of the RFC is restricted to connection-oriented transport protocols such as TCP. It defines an end-point initiated ping which requires an SD pong in response. A second approach described in Section 4.4.2 of the RFC is restricted to connectionless transport protocols such as UDP. It defines an endpoint-initiated STUN binding request which requires as SD-originated STUN binding response.

RFC 6223, *Indication of support for Keep-Alive*, was initially published as an internet draft (most recently as draft-ietf-sipcore-keep-12.txt). RFC 6223 defines a procedure that enables a SIP endpoint to signal its capability and willingness to send and receive periodic Keep-Alive messages to a device referred to by the RFC as an edge proxy, a role performed by the SD. After receiving such a signal, the SD returns a response indicating its willingness to exchange Keep-Alives, and specifying the interval between Keep-Alive exchanges.

RFC 5389, *Session Traversal Utilities for NAT (STUN)*, defines STUN binding requests and responses in Section 6 of the RFC. Endpoints that support endpoint-initiated Keep-Alives over a UDP connection must be capable of constructing and transmitting a STUN binding request, and receiving and parsing a STUN binding response.

Endpoint-Initiated Keep-Alive Negotiation

Endpoint-initiated Keep-Alive negotiation requires the presence of the keep parameter in the Via header of Registration requests and responses issued and received by a SIP endpoint. As defined in RFC 6223, keep is

A SIP Via header field parameter that a SIP entity can insert in the topmost Via header field that it adds to the request, to explicitly indicate willingness to send keep-alives towards its adjacent downstream SIP entity. A SIP entity can add a parameter value to the keep parameter in a response to explicitly indicate willingness to receive keep-alives from its adjacent upstream SIP entity.

While RFC 6223 allows the presence of the keep parameter in either REGISTER or INVITE requests and responses, the current implementation, is registration-based. keep parameters contained in methods other than REGISTER are ignored.

As shown in the following sample registration exchange, the SIP endpoint signals its willingness to exchange Keep-Alive messages by placing an unvalued keep parameter in the SIP Via header of a REGISTER request.

```
REGISTER sip:512@172.16.101.23:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;keep
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
Call-ID:1-14400@172.16.101.38
CSeq: 1 REGISTER
Max-Forwards: 70
Contact: "512" <sip:512@172.16.101.38:5070>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
```

The SD strips the keep parameter from the Via header of the REGISTER request, and forwards the request to the Registrar.

```
REGISTER sip:512@192.168.7.32:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Max-Forwards: 69
Contact: "512" <sip:512@192.168.101.23:5060;transport=udp>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
```

The Registrar indicates successful registration with a 200 OK REGISTER response back to the SD.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512"
<sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
```

```
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@192.168.101.23:5060;transport=udp>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

The SD indicates its willingness to exchange Keep-Alives by assigning a value, which specifies the interval between Keep-Alives, to the keep parameter and re-inserting the parameter and its assigned value into the Via header of the REGISTER response.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;keep=20
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512"
<sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@172.16.101.38:5070>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

Connection Oriented Keep-Alives

After the endpoint-initiated Keep-Alive exchange has been negotiated for a TCP connection, the SIP endpoint transmits a periodic ping at intervals specified by the value of the keep parameter. As shown below, the ping consists of carriage return (CR) and line feed (LF) characters.

```
CRLFCRLF
CR = %x0D
LF = %x0A
```

The SD responds with a pong as shown below.

```
CRLF
CR = %x0D
LF = %x0A
```

The request ping and the response pong are transmitted between SIP messages, and cannot be sent in the middle of SIP message. If the TCP connection is secured by TLS, the Keep-Alive requests and responses are transmitted within the TLS encrypted stream.

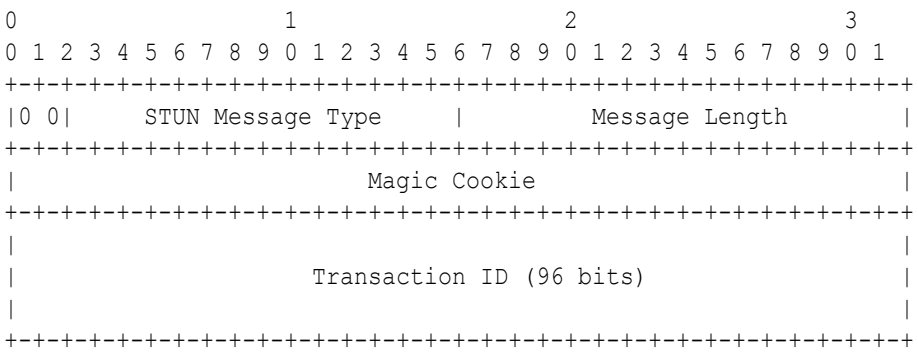
As specified in Section 4.4.1 of RFC 5626 the SD must respond to a ping within a 10-second interval. If the endpoint fails to receive a valid pong within that interval, it must treat the flow as failed.

Connectionless Keep-Alives

After the endpoint-initiated Keep-Alive exchange has been negotiated for an unreliable UDP connection, the SIP endpoint, acting as a STUN client, transmits a periodic STUN binding request so that the interval between each request is randomly distributed between 80 and 100 percent of the value of the keep parameter. Assuming a parameter value of 20 seconds, for

example, the SIP endpoint transmits a STUN binding request at random intervals between 16 and 20 seconds in length.

The STUN binding request has the following format.



The initial two bits of any STUN message are always 00

STUN Message Type

This 14-bit field contains the message class and method — for a STUN binding request, class is request and method is binding.

Message Length

This 16-bit field contains the length, in octets, of the Attribute section of the STUN binding request. Since the Attribute section is not present in a binding request, this field contains a value of 0.

Magic Cookie

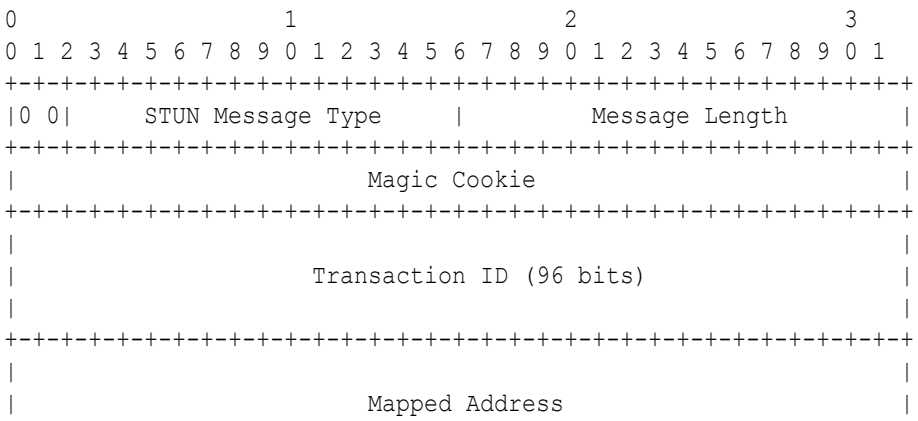
This 16-bit field always contains a value of 0x2112A442.

Transaction ID

This 96-bit field contains a random value that provides a transaction identifier. The value is generated by the STUN client and echoed in the STUN server response.

Upon receipt of a binding request, the SD, acting as a STUN server, transmits a STUN binding response. Receipt of the binding response by the endpoint confirms the UDP connection between the endpoint and the SD, and the viability of NAT bindings in the transmission path.

The STUN binding response has the following format.



```
|                                                                 |
|-----------------------------------------------------------------
```

The initial two bits of any STUN message are always 00.

STUN Message Type

This 14-bit field contains the message class and method — for a STUN binding response, class is response and method is binding.

Message Length

This 16-bit field contains the length, in octets, of the Attribute section of the STUN binding response. Since the binding response contains a single attribute, MAPPED-ADDRESS, the message length can be either 8 or 16 octets, depending on the IP address type.

Magic Cookie

This 16-bit field always contains a value of 0x2112A442.

Transaction ID

This 96-bit field contains a random value that provides a transaction identifier. The value is generated by the STUN client and echoed by the STUN server.

MAPPED-ADDRESS

This attribute contains the IP address and port number of the proximate NAT device, that is the address translator closest to the SD, that forwarded the STUN request toward the SD. Specific fields within the attribute identify the IP family (Version 4 or 6), the IP address, and port number. The attribute length is 8 octets if it contains an IPv4 address, or 16 octets if it contains an IPv6 address.

Once initiated, endpoint transmission of STUN binding requests and SD responses continue for the duration of the SIP Registration, 1 hour in the sample negotiation, or until the endpoint transmits a new REGISTER request. In the event of such a new request, the endpoint once again indicates its willingness to exchange STUN Keep-Alives with an unvalued keep parameter in the Via header. If endpoint-initiated Keep-Alive renegotiation is not successful, the endpoint must cease the transmission of Keep-Alive messages.

An endpoint failure to issue a timely STUN binding request is not fatal. In the absence of an expected request, the SD takes no action with regard to the UDP connection, or to current calls, nor does it remove entries from its registration cache. After reception of a tardy STUN request, the SD simply returns a STUN response.

In contrast, however, Section 4.4.2 of RFC 5626 states that an endpoint “considers the flow failed” if (1) it receives a malformed STUN bonding response from the SD, or (2) it fails to receive a timely binding response from the SD. If the endpoint receives a STUN binding response that contains a different MAP Address field from the previous response, It MUST treat this event as a failure on the flow.

Enpoint-Initiated Keep-Alives Configuration

You use the **registrar-keep-alive** attribute, available in SIP Interface configuration mode, to enable endpoint-initiated Keep-Alives on a SIP interface.

1. In Superuser mode, use the following ACLI command sequence to access SIP Interface configuration mode.

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

2. The **registrar-keep-alive** attribute enables endpoint-initiated Keep-Alive on the current SIP interface.

   none — (the default) disables endpoint-initiated Keep-Alive processing

   always — assuming that the endpoint has included the keep parameter in the Via REGISTER request header, forces the SD to place a keep parameter and an assigned in the Via REGISTER response — thus enabling endpoint-initiated Keep-Alive exchange with that endpoint

   bnat — assuming that the endpoint has included the keep parameter in the Via REGISTER request header, forces the SD to place a keep parameter and an assigned value in Via REGISTER response only if the requesting endpoint is identified as being behind an intervening NAT device (based on comparing source IP packet addresses with IP addresses extracted from the SIP request).

```
ORACLE(sip-interface)# registrar-keep-alive bnat
ORACLE(sip-interface)#
```

3. If Keep-Alive is enabled on the current SIP interface (**registrar-keep-alive** is always or bnat), use the **nat-interval** attribute to specify the value of the keep parameter provided by the SD to SIP endpoints served by a UDP connection.

   Allowable values are integers within the range 1 through 4294967295 (seconds).

   In the absence of an explicit assignment, this attribute defaults to a value of 30 seconds.

   The SIP endpoint transmits periodic STUN binding requests so that the interval between each request is randomly distributed between 80 and 100 percent of the value of the nat-interval attribute.

   Assuming the default value (30 seconds) the interval between STUN binding requests would vary from 24 to 30 seconds. This default value closely aligns to Section 4.4.2 of RFC 5626, which recommends that the time between each keep-alive request SHOULD be a random number between 24 and 29 seconds.

```
ORACLE(sip-interface)# nat-interval 20
ORACLE(sip-interface)#
```

4. If Keep-Alive is enabled on the current SIP interface (**registrar-keep-alive** is always or bnat), use the **tcp-nat-interval** attribute to specify the value of the keep parameter provided by the SD to SIP endpoints served by a TCP or TCP/TLS connection.

   Allowable values are integers within the range 1 through 999999999 (seconds).

   In the absence of an explicit assignment, this attribute defaults to a value of 90 seconds.

   The SIP endpoint transmits periodic pings at intervals specified by this attribute.

```
ORACLE(sip-interface)# tcp-nat-interval 120
ORACLE(sip-interface)#
```

5. Use **done**, **exit**, and **verify-config** to complete this configuration.
6. If necessary, repeat Steps 1 through 5 to configure Keep-Alives on additional SIP interfaces.

SD-originated Keep-Alive Negotiation

SD-originated Keep-Alives are useful on TCP connections with iOS (iPhone Operating System) clients, such as iPhones or iPads. iOS places inactive clients in a dormant state after a period of time. Once in the dormant state, clients cannot transmit date unless awakened by network traffic. SD-originated Keep-Alives are generated at a sufficient frequency to prevent iOS clients from entering the dormant state.

SD-originated Keep-Alive negotiation is similar to the earlier described negotiation, except for the parameter used. SD-originated Keep-Alive negotiation requires the presence of a well known proprietary parameter in the Via header of Registration requests and responses issued and received by a SIP endpoint. The proprietary parameter can be defined as follows:

A SIP Via header field parameter that a SIP endpoint can insert in the topmost Via header field that it adds to the request, to explicitly indicate willingness to receive Keep-Alives from its adjacent downstream SIP entity. This same SIP endpoint can assign a value to this proprietary parameter in the response to specify the keep-alive interval.

As shown in the following sample registration exchange, the SIP endpoint signals its willingness to receive Keep-Alive messages by placing an rkeep parameter in the SIP Via header of a REGISTER request; the assigned value of 20 specifies the interval between Keep-Alives.

```
REGISTER sip:512@172.16.101.23:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;rkeep=20
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
Call-ID:1-14400@172.16.101.38
CSeq: 1 REGISTER
Max-Forwards: 70
Contact: "512" <sip:512@172.16.101.38:5070>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
```

The SD strips the rkeep parameter from the Via header of the REGISTER request, and forwards the request to the Registrar.

```
REGISTER sip:512@192.168.7.32:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Max-Forwards: 69
Contact: "512" <sip:512@192.168.101.23:5060;transport=udp>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
```

The Registrar indicates successful registration with a 200 OK REGISTER response back to the SD. The expires parameter in the Contact header grants a registration period of 1 hour (3600 seconds).

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512"
<sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@192.168.101.23:5060;transport=udp>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

After inserting the rkeep parameter and assigned value in the Via header of the REGISTER response, the SD forwards the response to the endpoint The presence of the rkeep parameter signals the SD's willingness to transmit Keep-Alive messages, and the parameter value specifies the exchange frequency in seconds.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;rkeep=20
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512"
<sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@172.16.101.38:5070>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

SD-Originated Keep-Alive Format

After the Keep-Alive exchange has been negotiated for a TCP connection, the SD, transmits a periodic ping at a frequency specified by the proprietary parameter (rkeep in the example negotiation). As shown below, the ping consists of carriage return (CR) and line feed (LF) characters.

CRLFCRLF

CR = %x0D

LF = %x0A

pings are transmitted between SIP messages, and cannot be sent in the middle of SIP message. If the TCP connection is secured by TLS, pings are transmitted within the TLS encrypted stream.

SD-Initiated Keep-Alives Configuration

Use the following procedure to configure SD-initiated Keep-Alives.

1. In Superuser mode, use the following ACLI command sequence to access SIP Interface configuration mode.

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

2. The **options sbc-initiated-keep-alive-name** attribute enables SD-initiated Keep-Alive on the current SIP interface and specifies the name of the proprietary Via header parameter used during the SD-initiated Keep-Alive negotiation.

   Prefix the command with a plus (+) sign to avoid over-writing existing options configuration.

   The following example enables SD-initiated Keep-Alives and identifies the proprietary rkeep header as the header that contains negotiation information.

   > **Note:**
   >
   > Endpoints require prior knowledge of this proprietary header name because the SD lacks the capability to relay configuration data to relevant endpoints.

```
ORACLE(sip-interface)# options +sbc-initiated-keep-alive-name=rkeep
ORACLE(sip-interface)#
```

3. The **option sbc-initiated-keep-alive-interval** attribute specifies the frequency of SD-initiated Keep-Alives.
4. If SD-initiated Keep-Alive is enabled on the current SIP interface (**options sbc-initiated-keep-alive-name** identifies a proprietary Via header parameter used for Keep-Alive negotiations), use the **options sbc-initiated-keep-alive-interval** attribute to specify the default interval between SD-initiated pings.

   Prefix the command with a plus (+) sign to avoid over-writing existing options configuration.

   Allowable values are integers within the range 1 through 999999999 (seconds).

   In the absence of an explicit assignment, this attribute defaults to a value of 30 seconds.

   This default interval will be used only the proprietary parameter proposed by the endpoint lacks a value.

```
ORACLE(sip-interface)# options +sbc-initiated-keep-alive-interval=45

ORACLE(sip-interface)#
```

5. Use **done**, **exit**, and **verify-config** to complete this configuration.
6. If necessary, repeat Steps 1 through 5 to configure SD-initiated Keep-Alives on additional SIP interfaces.

Statistics

The SD does not gather or maintain endpoint-initiated Keep-Alive statistical counts. However, malformed STUN binding requests and responses are included in the count of mal-formed SIP messages. Additionally, if DoS is enabled, STUN binding exchanges are included in endpoint message counts.

SIP Registration Local Expiration

When you deploy multiple Oracle Communications Session Border Controllers (SBC) in series and they have registration caching and HNT configured, registration cache entries might expire prematurely in instances with several devices provisioned with the same address of record (AoR). Now you can configure a SIP interface option to prevent the premature expiration.

When you use registration caching and HNT, the SBC adjusts the expiration time it sends to user agents (UAs) in REGISTER responses based on the registration interval you configure. It can be the case that a SIP user has multiple registered contact endpoints at the UA to which a response is sent. If the URI in the Contact contains the UA's address and that UA included the Contact in the REGISTER request, then the Contact is seen as exclusively belonging to that UA. In the REGISTER response, this Contact (exclusive to the UA) includes the local expiration time, a time based on the SIP interface configuration's registration or NAT interval value. Additional Contacts (not exclusive to the UA) in the REGISTER response have the expiration time from the REGISTER response the registrar sent to the SBC.

It is this default behavior can cause registration cache entries to expire prematurely in the SBC nearest a registrar when multiple SBCs are deployed in series. Multiple registering UAs for a single SIP user, for example, might trigger the early expiration. The SIP you can configure an option per SIP interface that causes the SBC to send the local registration expiration time in all in the Expires parameter of all Contact headers included in REGISTER responses sent from the SIP interface.

SIP Registration Local Expiration Configuration

You can configure this feature either for the global SIP configuration, or for an individual SIP interface.

To configure SIP registration local expiration for the global SIP configuration:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-config** and press Enter. If you are editing an existing configuration, select the configuration so you can enable this feature.

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

4. **options**—Set the options parameter by typing **options**, a Space, the option name **reg-local-expires** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-config)# options +reg-local-expires
   ```

   If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the**

new option to this configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

5. Save and activate your configuration.

To configure SIP registration local expiration for an individual SIP interface:

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

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

7. Type **session-router** and press Enter.

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

8. Type **sip-interface** and press Enter. If you are editing an existing configuration, select the one on which you want to enable this feature.

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

9. **options**—Set the options parameter by typing **options**, a Space, the option name **reg-local-expires** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-interface)# options +reg-local-expires
```

If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to this** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

10. Save and activate your configuration.

Simultaneous TCP Connection and Registration Cache Deletion

You can configure the Oracle Communications Session Border Controller to automatically start a timer when a user deregisters or changes location by registering with a new contact address.

Not all devices tear down TCP connections associated with these old addresses when a user registers with a new contact address.

Registration Cache Deletion Configuration

You can apply **suppress-reinvite** to the sip-interface facing the User Agents whose re-INVITEs are to be responded to locally.

To enable Registration Cache Deletion:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing configuration, you will need to select and edit it.

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

4. **tcp-conn-dereg**—The delay (in seconds) that is used to determine when to terminate the TCP connection for a user that has been removed from the registration cache or changed location. This feature can be disable by setting the value to zero.

```
ORACLE(session-agent)# tcp-conn-dereg 5300
```

5. Save your work.

SBC Incorrectly Appends Cookie in SIP REGISTER Message

The Oracle Communications Session Border Controller does not recognize a SIP URI containing tel-URI information if it doesn't also contain a "user=phone" parameter. This behavior adversely affects creation of the acme_nat tag and placement of the cookie

You can enable the option **proces-implicit-tel-URI** to recognize an implicit tel-URI and places the cookie in the correct location.

process-implicit-tel-URI Configuration

To enable proccess-implicit-tel-URI

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

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

2. Type **session-router** and press Enter.

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

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

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

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **proccess-implicit-tel-URI** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-config)# options +proccess-implicit-tel-URI
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

SIP HNT Forced Unregistration

If you use HNT and experience the issue explained in this section, consider using the Oracle Communications Session Border Controller (SBC) forced unregistration feature. When this feature is enabled and a registration entry for an endpoint expires, the SBC notifies the soft switch to remove this binding using REGISTER message. In that REGISTER message, the expires header will be set to 0 and the expires parameter in the Contact header will also be set to 0.

The benefits of using forced unregistration include:

- Leveraging existing HNT configuration to provide near real-time information about the UA's status to the registrar/softswitch
- Preserving resource utilization for the SBC and the softswitch by deleting a contact binding that is no longer valid or needed
- Preventing extra bindings from being generated at the softswitch (e.g., in instances when the UA or NAT restart)

This feature applies to:

- HNT endpoints with registration caching enabled by default, and when the **nat-traversal** parameter in the SIP interface configuration is set to always
- non-HNT endpoints with registration caching enabled, when the registration-interval parameter in the SIP interface configuration is used in the expires header sent to the UA in the 200 OK

When to Use Forced Unregistration

For typical HNT use, it is common that the registration interval between the client UA and the Oracle Communications Session Border Controller (SBC) is between 60 and 120 seconds. This differs significantly from the re-registration interval between the SBC and the and the registrar, which varies from approximately 30 to 60 minutes.

If the UA fails to refresh its registration, the contact binding at the SBC is deleted after the registration expires. This expiration is determined by the expires= header in the 200 OK. The binding at the real registrar will remain intact. This creates a discrepancy between the real state of the UA and state of the softswitch. In the best case scenario, the contact binding expires at the softswitch after a few minutes.

For network management, this discrepancy can be problematic because the service provider would be unaware of the UA's status until the binding expires at the softswitch. This can take a considerable amount of time to happen.

In addition, the SBC encodes a cookie in the userinfo of the Contact header in the REGISTER message. This is a function of the source IPv4 address and port from which the request came, i.e., the ephemeral port in the NAT for DSL scenarios. Therefore, additional bindings that remain for long periods of time are created at the registrar if, for example, the:

- UA reboots
- Ethernet link between the UA and the DSL router is lost for over two minutes

- DSL crashes
- DSL/ATM layer between the DSL router

Caution for Using Forced Unregistration

You should use caution when applying SIP HNT forced unregistration for the following reasons:

- It can have an impact on the performance of your Oracle Communications Session Border Controller and the registrar, especially when you have a large number of HNT endpoints in your configuration that become unavailable simultaneously.
- It is possible that the registrar might become vulnerable to overload in the case where the registrar must authenticate a large number of register messages generated when HNT endpoints are de-registered. It is possible that the cached registration credentials might become “stale” over time (e.g., the nonce value usually has a limited lifetime). Without proper credentials, the registrar will reject the de-registrations.

Given these concerns, we recommend that you consult with your Oracle systems engineer before adopting the use of forced unregistration.

SIP HNT Forced Unregistration Configuration

To enable SIP HNT forced unregistration:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. Use the ACLI **select** command so that you can work with the SIP configuration.

   ```
   ORACLE(sip-config)# select
   ```

5. **options**—Set the options parameter by typing **options**, a Space, the option name **force-unregistration**, and then press Enter.

   ```
   ORACLE(sip-config)# options +force-unregistration
   ```

   If you type options force-unregistration, you will overwrite any previously configured options. In order to append the new option to the sip-config’s options list, you must prepend the new option with a plus sign as shown in the previous example.

> **Note:**
>
> In the event of a failover, the Service Route information is required by the SBC to trigger a DE-REGISTER. When the **force-unregistration** option is set, the SBC expects the Registrar to send a Service-Route Header in the 200 OK response of the REGISTER Request.

Adaptive HNT

This section explains how to configure adaptive HNT. The adaptive HNT expires feature allows the Oracle Communications Session Border Controller to automatically determine the maximum SIP REGISTER message expires time interval in order to keep each individual NAT pinhole open when performing SIP HNT. This feature applies only to SIP over UDP.

Overview

Without adaptive HNT, the Oracle Communications Session Border Controller keeps NAT pinholes open and port mapping cached by forcing the UAC to send frequent SIP REGISTER messages. It does so by setting the expires time to a short interval. Some NATs only need a message to be sent by the private client once every twenty minutes, while other NATs delete their cache/pinhole in thirty seconds if no messages appear. Given this large variation in time intervals, the Oracle Communications Session Border Controller's **nat-interval** (expire time), set on the applicable **sip-interface**, has been set to a low value in order to support as many NAT types as possible. However, CPU performance and scalability issues result from such a small refresh time, especially when there is a very large number of potential registered users.

When you use adaptive HNT, the Oracle Communications Session Border Controller waits for a time interval and then sends a SIP OPTIONS message to the UAC to see if it can still be reached. If the UAC can still be reached, the Oracle Communications Session Border Controller increases the timer and tries again. In case the pinhole closes because it has exceeded the NAT's cache time, the Oracle Communications Session Border Controller sets the expires time to be slightly longer than the time it tests using the OPTIONS method. This way, the UAC will send another REGISTER message shortly thereafter and impact on service will be minimal.

Adaptive HNT Example

An example call flow using adaptive HNT involves a basic HNT user and a Oracle Communications Session Border Controller. It begins when the Oracle Communications Session Border Controller receives and forwards the 200 OK for the REGISTER message. Then the Oracle Communications Session Border Controller sends an expires timer for slightly longer than the time for which to test; in this example, it begins the test for the amount of time set for the minimum NAT interval, specified by the **nat-interval** set on the applicable **sip-interface**. It adds ten seconds to this time when it sends the expires timer. This way, there is time for the OPTIONS message to be sent before the REGISTER message is received (which would refresh the NAT's cache). The Oracle Communications Session Border Controller also tries to keep the REGISTER time short enough so that even if the NAT pinhole closes, there is minimal time before the UAC creates a new NAT binding by sending another REGISTER. Because a ten second interval may be too long, you might want to set this value to a better-suited time.

The test succeeds with a minimum test-timer because the UAC responded to the OPTIONS message. So the test-timer value is increased by thirty seconds and tried again. The expires

time in the REGISTER message will be increased to the test-timer value plus ten seconds. This time, the UAC does not respond to the OPTIONS message even though it was sent multiple times. Because the OPTIONS fails, when the Oracle Communications Session Border Controller receives another REGISTER, it responds with the previously successful timer value (in this case, the minimum NAT interval).

However, if the OPTIONS request succeeds, then the Oracle Communications Session Border Controller persists with the test until it fails or until the maximum NAT timer value is reached. In this case, when the OPTIONS message fails, the Oracle Communications Session Border Controller uses the last successful test-timer value as the time for the expires header in the 200 OK for the REGISTER message.

Adaptive HNT over TCP

The SBC supports Adaptive Host NAT Traversal (AHNT) over TCP in addition to UDP. TCP AHNT configuration and behavior is largely the same as for UDP. You use **sip-interface** parameters that are equivalent to, but separate from the UDP parameters to configure Adaptive HNT for TCP.

The SBC detects that an endpoint is behind a NAT using the same process for both UDP and TCP, comparing the layer 3 IP address of a REGISTER with the VIA header before proceeding with a registration. If the first VIA is different from the layer three address, the SBC assumes there is a NAT and sets up flows accordingly. For subsequent HNT timing processes, the SBC refers to TCP-related HNT configuration parameters, which are separate from those used for UDP transport.

When configuring adaptive HNT for TCP, you use the **tcp-nat-interval** parameter on a **sip-interface** to specify the NAT interval for TCP transport. The SBC starts AHNT testing using this value as the initial interval after which the SBC sends an OPTIONS request and waits for an OPTIONS response from the endpoint it is testing until the test expires, calculated as **tcp-nat-interval** plus the **tcp-nat-int-increment**.

Parameters on the **sip-interface** that apply to AHNT testing over TCP connections include:

- **tcp-sip-dynamic-hnt**
- **tcp-nat-interval**
- **tcp-max-nat-interval**
- **tcp-nat-int-increment**
- **tcp-nat-test-increment**

The system ends the AHNT test when:

- The OPTIONS request was not sent because the NAT gateway or UA closed the connection.
- The new expires header value calculated by the SBC for the next RE-REGISTERs 200 OK exceeds your configured value for **max-nat-interval** (or **tcp-max-nat-interval**).
- The early RE-REGISTER occurrence count, monitored by the SBC exceeds 3.
- The UE sends a keep-alive SIP request for a contact to keep the NAT port open for 5 or more consecutive times.
- When you configure the **nat-int-increment** (or **tcp-nat-int-increment**) value higher more than 32s, and the OPTIONS sent by the SBC has not received a response within the Client Transaction expires time (32s).

When the test is complete, the SBC uses the last successful test time for the expires value in the current and subsequent refresh REGISTER's 200 OK for that contact.

Feature Limitations

This Adaptive HNT over TCP feature generates several limitations on SBC operation:

- When you deploy this feature, the configuration change does not apply to existing contacts already using AHNT functionality.
- When you enable both **sip-dynamic-hnt** and **tcp-sip-dynamic-hnt**, the SBC applies this feature based on transport type of each REGISTER request. Either AHNT over TCP or UDP is applicable for a users contact.
- A UA with the same contact does not change its transport type when it sends refresh REGISTER. A UA can only change its transport type by a new registration with the new contact for that transport type.
- The SBC synchronizes the learned expires time to HA only after the AHNT test is complete for both UDP and TCP contacts.
- A SIP contact URI (IP:Port:transport) must remain the same in a user's REGISTER and RE-REGISTER. The transport type between the UA or NAT Gateway and the SBC cannot change until that contact expires.

Adaptive HNT Call Flows

This section presents call flow diagrams to help explain AHNT functionality on the SBC.

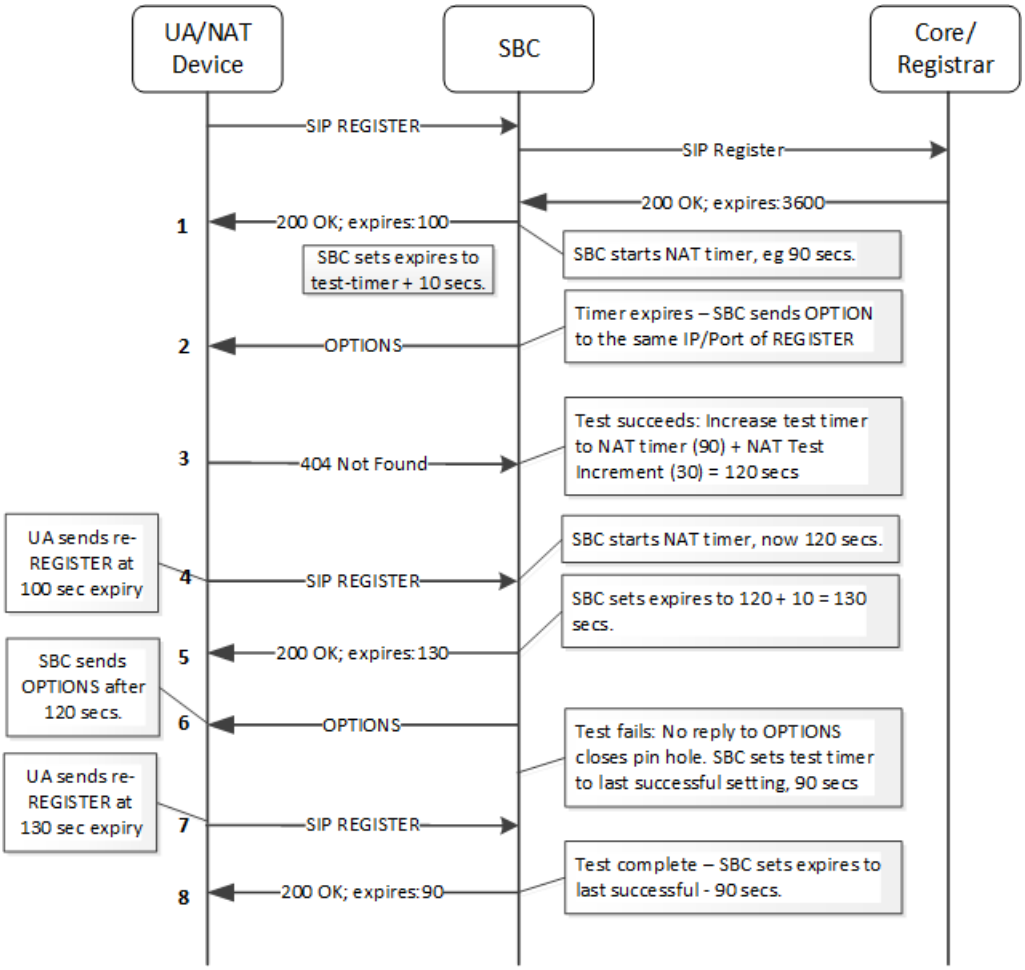
The call flows below depict the SBC performing AHNT testing to establish an expires value that keeps a NAT pinhole open while additional signaling and media traverses the NAT device for a specific call. The NAT device uses this pinhole for either UDP or TCP transport.

Call flows are the same for both TCP and UDP, using the applicable parameters on the **sip-interface**:

- **sip-dynamic-hnt** (or **tcp-sip-dynamic-hnt**)—Enables or disables AHNT for UDP or TCP connections respectively.
- **nat-interval** (or **tcp-nat-interval**)—The number of secs used to calculate the test-timer for sending OPTIONS request or to perform the AHNT test.
- **nat-int-increment** (or **tcp-nat-int-increment**)—The number of seconds added to the expires header value in REGISTERs 200 OK. The actual AHNT test would be done only during this incremented interval time we added to expires header in REGISTERs 200OK.
- **nat-test-increment** (or **tcp-nat-test-increment**)—The number of seconds used to increment the next test timer value when a RE-REGISTER arrives.
- **max-nat-interval** (or **tcp-max-nat-interval**)—The maximumn number of seconds used for the expires value. When reached, the SBC stops AHNT testing.

Normal ANHT Process

This call flow depicts an example of the SBC performing AHNT testing. The steps provided below expand upon the behavior.



As shown in the call flow above, the SBC implements this feature within the context of registration. The first three lines in the flow present a successful registration using the registrar, and ending with a 200 OK from the registrar to the SBC. Assume TCP transport, but note the behavior is the same for UDP when the equivalent configuration is in place.

When it receives the 200 OK reply to the REGISTER request from the registrar, the SBC refers to the expires value, normally 3600 seconds, and checks whether you have enabled the **tcp-sip-dynamic-hnt** parameter for this connection's transport protocol on the **sip-interface** where the REGISTER request arrived. If so, the SBC:

1. Replaces the expires header parameter value in its 200 OK to the endpoint with the sum of the values configured in the **tcp-nat-interval** and **tcp-nat-int-increment** parameters. In the flow, the system has used the defaults of both parameters respectively and arrived at an expires time of 90 + 10 = 100 seconds.
   In addition, the SBC starts a timer using the value configured in **tcp-nat-interval**. Notice that the expires time is slightly longer than the timer time. This allows the SBC to decide whether this test iteration succeeded or failed and change the value of the next expires time accordingly.
2. The SBC sends an OPTIONs message at the 90th second and waits for a reply. Note that the test timer is active.
3. The end station replies, in this case, with a 404 - Not found message within the test window. This concludes the first iteration of the AHNT test successfully, indicating that the NAT device does not close the pinhole sooner than the expires time of 90 seconds. Because this test was successful, the SBC recalculates the test time using the sum of the values configured in the **tcp-nat-interval** and **tcp-nat-testincrement** parameters. Again,

the flow shows the system using the defaults of both parameters respectively and arriving at an expires time of 90 + 30 = 120 seconds.

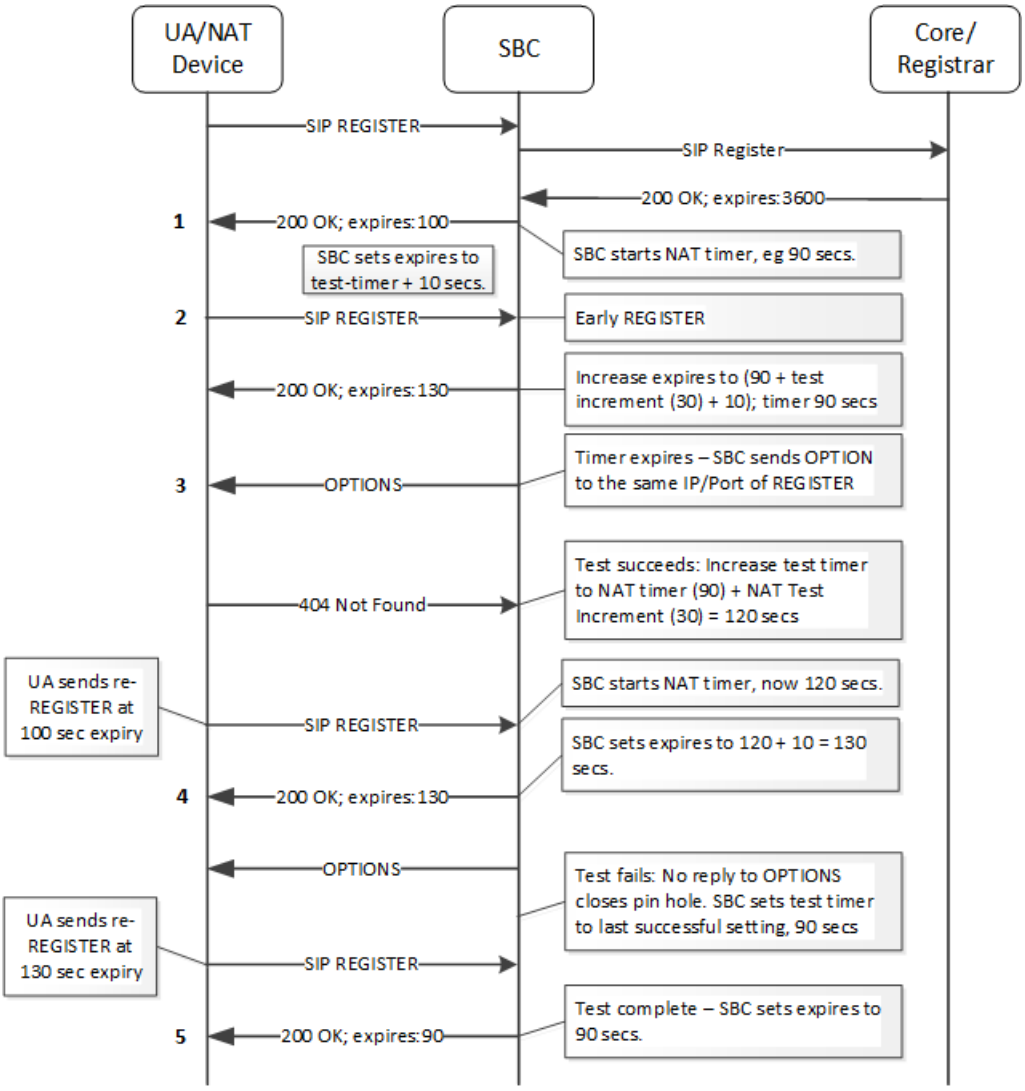
4. The UA responds with a re-REGISTER at the original expires time.
5. The SBC sends the 200 OK to the re-REGISTER, including a new expires time and starts the test timer.
   The SBC sets the expires using the sum of the test times and the **tcp-nat-int-increment** parameters (120 +10 = 130 seconds).
6. The SBC sends an OPTIONS request at or after 120th second and waits for the response. The actual testing happens between 120th second to 130th second.
   For this test iteration, the UA does not respond before the test window expires (130 seconds). The SBC reverts is test timer to that last successful value of 90 seconds.
7. The UA sends a re-REGISTER at the expires time of 130 seconds.
8. The SBC sends a 200 OK for the re-REGISTER and sets the expires time to the same value as the last successful test timer value, which in this case is 90 seconds.

> **Note:**
>
> The system does not perform any calculation using the **tcp-nat-int-increment** value for the final setting.

Early Refresh REGISTER Scenario

In some cases, a UA may send a re-REGISTER before the expires time. The SBC does not end its AHNT testing when this happens, instead adjusting its test timing to try and avoid any further early REGISTER refreshes. Specifically, the SBC increases the expires time it sends to the UA in the 200 OK by the NAT test's increment value, and restarts the OPTIONS test timer without increasing its value.

The SBC, however, also tracks the number of early REGISTER refreshes that it receives during a session, and ends AHNT testing if that count reaches three in a row. Each early refresh REGISTER resets the NAT device pinhole. If the NAT device is keeping the pinhole open, there is no need for the SBC to expend processing cycles for the same purpose.



As shown in the call flow above, the SBC implements this feature within the context of registration. The first three lines in the flow present a successful registration using the registrar, ending with a 200 OK from the registrar to the SBC.

1. Like the first example, this flow includes the system using the defaults of the interval and increment parameters respectively to set an expires time of 90 + 10 = 100 seconds. In addition, the system starts the OPTIONS timer using 90 seconds.
2. In this flow, however, the UA sends a re-REGISTER before the OPTIONS timer expires. As described above, the SBC increases the expires time using the **tcp-nat-test-increment** value (30 seconds) and sends the 200 OK with an expires time of 130 seconds. The SBC also restarts the OPTIONS test timer using its original value of 90 seconds.
3. After 90 seconds, the SBC sends the OPTIONS request and gets a 200 OK (or other) response.
4. For this iteration, there is no early REGISTER refresh. The SBC starts the test timer using its normal calculation, in this case 90 + 30 = 120, and sends the 200 OK with an expires calculated as the first successful test, 130 seconds.
5. This call flow completes by receiving no reply to the OPTIONS request in time, so the SBC reverts to the last successful expires time of 90 seconds.

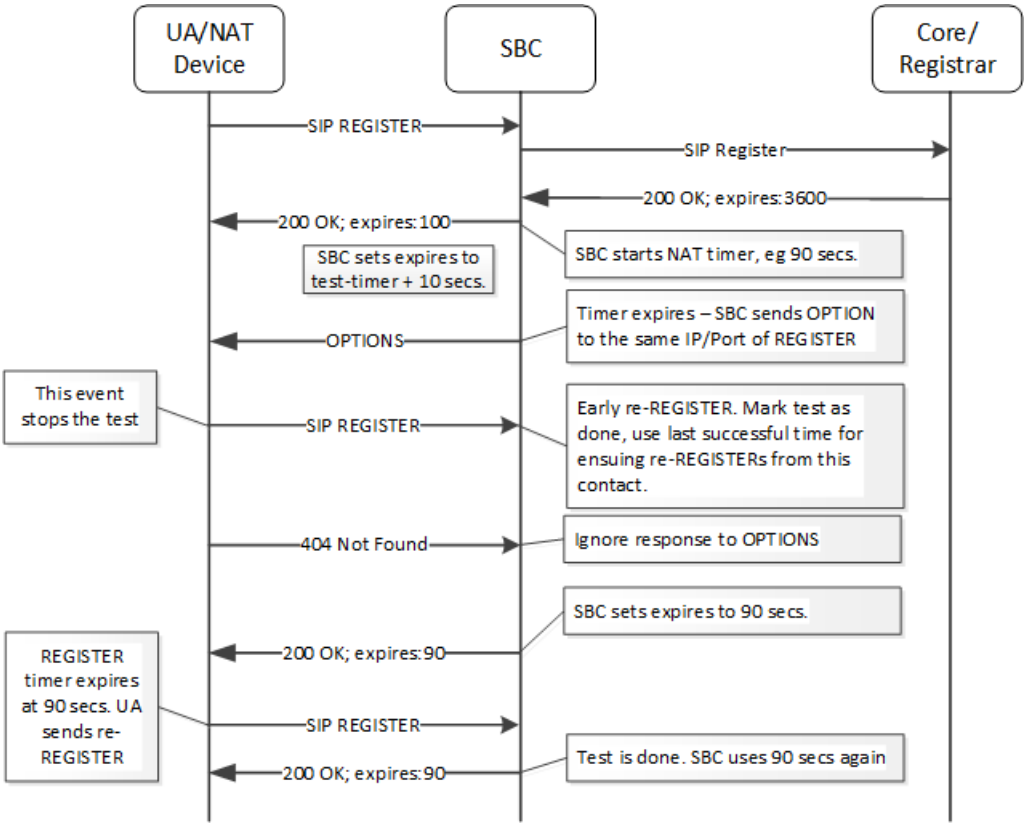
> **Note:**
>
> The system does not perform any calculation using the **tcp-nat-int-increment** value for the final setting.

If the UA/NAT sends another early refresh REGISTER at any time during the session, the SBC increases the expires time in the 200 OK by the **tcp-nat-test-increment** and restarts the test timer to its original value.

Early Refresh REGISTER while OPTIONS In-Progress

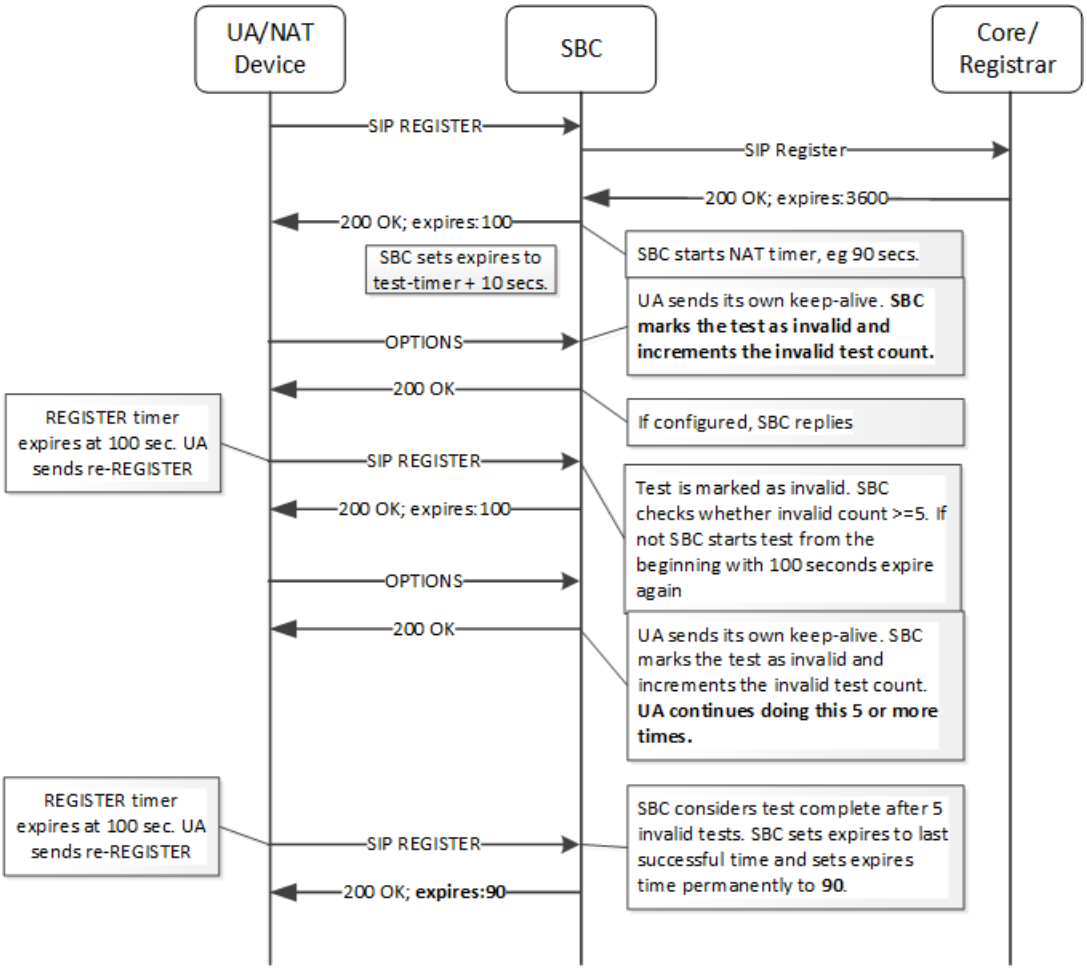
Within this example, the SBC receives a refresh REGISTER before getting a 200 OK for the OPTIONS request it sent to the UA. When it gets this refresh REGISTER, the SBC marks the test as done and sets the expires time to the **tcp-nat-interval** if no test completed successfully, or to the last successful test's expires time.

In this same scenario, if the OPTIONS client transaction expires due to no response from UA, the SBC marks the test as done and inserts the same expires header values as above. By default, OPTIONS expiry happens at 32 seconds.



Keep-Alive from UA Flow

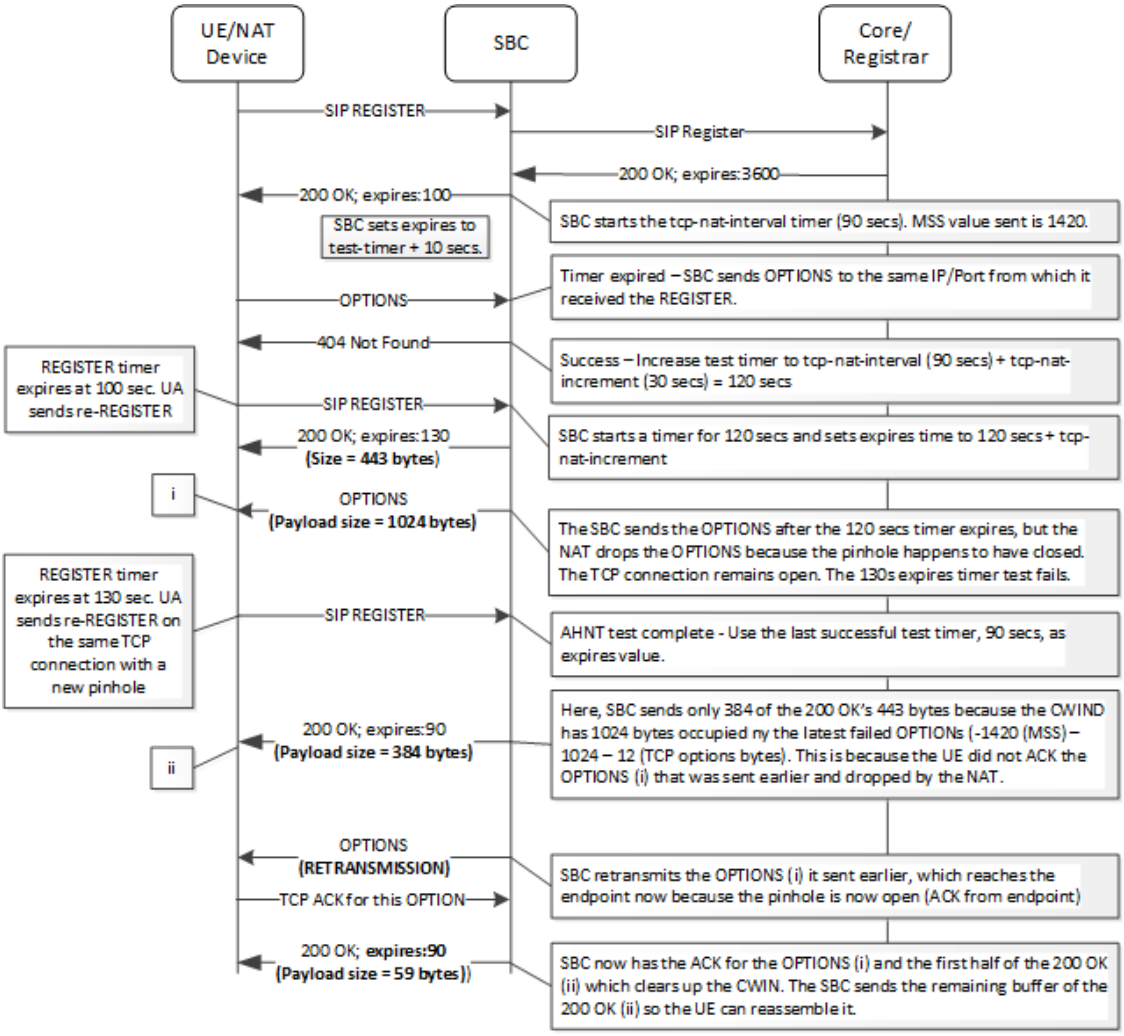
Within other use cases, the UA itself tries to keep the NAT pinhole open by sending OPTIONS or any other out of dialog SIP request for that contact from the same IP address and port number that it used for its last REGISTER. When it detects this, the SBC disables AHNT testing for that contact for 5 or more consecutive instances, and uses the last successful test time for future RE-REGISTERs expires times.



AHNT Operation in Conjunction with TCP Behavior

Some call flows applicable to this feature are subject to normal packet delivery delays that allow the NAT pinhole to close. Examples include TCP connections that experience segmented packet transfer wherein a segment may not have received an ACK, lengthening the amount of time consumed by a transmission. In these cases, the SBC persists with its AHNT behaviors, ultimately waiting out the closed pinhole and proceeding with packet reassembly after the TCP connection resumes transmission.

Consider the situation wherein the OPTIONS pings are retransmitted after a pinhole is closed and re-opened. The SBC would not be aware of whether this situation was caused by the absence of an ACK to an OPTIONS or because there was a network issue. In this case, the SBC follows standard TCP behavior and waits for a segment's ACK from an endpoint within its congestion window (CWIND) before sending the additional segments to the endpoint.



The steps below provide additional explanation on the call flow above:

1. The SBC establishes the TCP connection with TCP maximum segment value (MSS) of 1420. The SBC starts AHNT testing and establishes an expires value of 100.
2. The SBC follows its AHNT testing procedure until it finds an expires value that keeps the pinhole open.
3. In the meantime, the end station sends another REGISTER to which the SBC provides a 200 OK.
4. Notice the first 200 OK sent by the SBC is 443 bytes. The expires timer is lengthened as the AHNT testing is continuing to identify the longest window it can used without the pinhole closing.
5. After the re-REGISTER and 200 OK, the NAT device happens to close the pinhole.
6. The SBC sends an OPTIONS, with a size of 1024 bytes to continue with the AHNT test. But OPTIONS does not reach the UE because the pinhole is closed and it was dropped by the NAT. TCP keeps the connection open waiting for the ACK to the OPTIONS.
   In addition, the TCP CWIND still has 1024 bytes occupied by the most recent OPTIONS.
7. The UE sends another refresh REGISTER, resulting in the SBC completing the AHNT test and reverting to the last successful expires timer of 90 seconds. This re-REGISTER also re-opens the NAT pinhole.

8. Next, the SBC prepares to send 200 OK with expires of 90 seconds. Because the 1024 bytes of the unanswered OPTIONS still occupies the a large amount of the CWIN, the SBC can only send a part of the 200 OK, in this case 384 bytes.
9. Later, the SBC re-transmits an OPTIONS ping and receives an ACK because the pinhole is open. This ACK applies to the OPTIONS and the 384 bytes of the outstanding 200 OK.
10. The CWIN is now cleared, so the SBC is able to send the remaining 59 bytes of the 200 OK.
11. At this point, every re-REGISTER's 200 OK for this UE has an expires value set to 90 seconds.

Synchronize A-HNT Successful Timer to Standby

Adaptive HNT enables the Oracle Communications Session Border Controller to determine, through testing, an optimum SIP REGISTER expires time interval that keeps the NAT pinhole open. For an HA node, this successful time value is determined through testing by the active system and then replicated to the standby. If there is a switchover during the active system's testing process, then it will restart for that endpoint.

Adaptive HNT Configuration

You configure the SIP interface to set the state of this feature and to define the increments of time the Oracle Communications Session Border Controller uses to perform adaptive HNT. Remember that the Oracle Communications Session Border Controller uses the time you specify as the NAT interval, the supported time interval, as the basis on which to begin testing.

If you are configuring AHNT for TCP connections, the following **sip-interface** parameters perform the same functions as the equivalent UDP parameters in the procedure below:

- **tcp-sip-dynamic-hnt**
- **tcp-max-nat-interval**
- **tcp-nat-int-increment**
- **tcp-nat-test-increment**

To configure adaptive HNT:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. **sip-dynamic-hnt**—Enable this parameter if you want to use adaptive HNT. The default value is **disabled**. The valid values are:
   - enabled | disabled

> **Note:**
>
> Updates to **sip-dynamic-hnt** applies to new registration and when the registration cache is cleared.

5. **max-nat-interval**—Set the amount of time in seconds that testing should not exceed. The Oracle Communications Session Border Controller will keep the expires interval at this value. The default value is **3600**. The valid range is:
   - Minimum—0
   - Maximum—4294967295
6. **nat-int-increment**—Set the amount of time in seconds to use as the increment in value in the SIP expires header. The default value is **10**. The valid range is:
   - Minimum—0
   - Maximum—4294967295
7. **nat-test-increment**—Set the amount of time in seconds that will be added to the test timer. The default value is **30**. The valid range is:
   - Minimum—0
   - Maximum—4294967295

SIP IP Address Hiding and NATing in XML

Adding to its topology hiding and NAT capabilities, the Oracle Communications Session Border Controller now performs those functions for pertinent IP addresses that are not part of the standard SIP message header format. Previously, such addresses were visible to the next hop in the SIP session path.

Note that this feature adds to the Oracle Communications Session Border Controller's pre-existing ability to perform this function for XML messages; this new support is specifically for the keyset-info message type.

For incoming SIP NOTIFY messages, the Oracle Communications Session Border Controller searches for the application/keyset-info+xml content type in the message. When it finds this content type, it searches further to detect the presence of <di:remote-uri> or <di:local-uri> XML tags and then NATs the IP addresses in the tags it finds. Specifically, the Oracle Communications Session Border Controller changes:

- The <di:remote-uri> IP address to be the egress SIP interface's IP address
- The <di:local-uri> IP address to be the Ip address of the next hop to which the message is being sent

Sample SIP NOTIFY with NATed XML

The following is a sample SIP NOTIFY message as it might arrive at the Oracle Communications Session Border Controller.

> **Note:**
>
> that it contains the <di:remote-uri> or <di:local-uri> XML tags on which the system will perform NAT; these lines appear in bold text.

```
NOTIFY sip:15615281021@10.152.128.253:5137;transport=udp SIP/2.0
To: 15615281021  <sip:15615281021@10.152.128.102:5080>;tag=5c93d019904036a
From: <sip:15615281021@10.152.128.102:5080>;tag=test_tag_0008347766
Call-ID: 3215a76a979d0c6
CSeq: 18 NOTIFY
Contact: <sip:15615281021@10.152.128.102:5080;maddr=10.152.128.102>
Via: SIP/2.0/UDP 10.152.128.102:5060;branch=z9hG4bK_brancha_0023415201
Event: keyset-info
Subscription-state: active;expires=2778
Accept: application/keyset-info+xml
Content-Type: application/keyset-info+xml
Content-Length: 599
Max-Forwards: 70
<?xml version="1.0"?>
<keyset-info xmlns="urn:ietf:params:xml:ns:keyset-info"
     version="16"
     entity="15615281021">
  <ki-data>
    <ki-state>"active"</ki-state>
    <ki-event>"unknown"</ki-event>
  </ki-data>
   <di:dialog id="dialog_id_201" call-
id="1395216611-1987932283256611-11-0884970552" local-
tag="test_tag_0008347790" direction="recipient">
    <di:state>trying</di:state>
    <di:duration>2778</di:duration>
    <di:local-uri>sip:15615281021@10.152.128.253:5137</di:local-uri>
    <di:remote-uri>sip:1004@10.152.128.102</di:remote-uri>
   </di:dialog>
</keyset-info>
```

Once the Oracle Communications Session Border Controller has completed the NAT process, the <di:remote-uri> and <di:local-uri> XML tags look like this

```
<di:local-uri>sip:15615281021@192.168.200.99:5137</di:local-uri>
<di:remote-uri>sip:1004@192.168.200.49</di:remote-uri>
```

because egress the SIP interface's IP address is 192.168.200.49 and the next hop's IP address is 192.168.200.99.

This feature does not require any configuration.

SIP Server Redundancy

This section explains how to configure SIP server redundancy. SIP server redundancy involves detecting that an upstream/downstream SIP signaling entity has failed, and adapting route policies dynamically to remove it as a potential destination.

Overview

You establish SIP server redundancy by creating session agents, which are virtual representations of the SIP signaling entities. These agents are then collected into a session agent group, which is a logical collection of two or more session agents that behaves as a single aggregate entity.

Rather than direct signaling messages to a single session agent (IP), the signaling message is directed to a session agent group (SAG). The group will have a set distribution pattern: hunt, round robin, proportionally distributed, and so on. Signaling is spread amongst the agents using this chosen pattern.

You direct the signaling message by configuring a route policy, known as a local policy, which determines where SIP REQUESTS should be routed and/or forwarded. The values in the To and From headers in the SIP REQUEST are matched with the content of the local policy within the constraints set by the session agent's previous hop value and SIP interface values such as the list of carriers.

To summarize, you need:

- two or more session agents
- a session group containg those session agents
- a local policy which directs traffic to the session agent group

Configuration Overview

You make a session agent group a target by using a local policy to select the next hop from the members of a session agent group. You need to set the replace URI field of the configured local policy to enabled; which causes NAT rules such as realm prefixing to be overridden. The replace URI field allows you to indicate whether the local policy's value is used to replace the Request-URI in outgoing requests. This boolean field can be set to either enabled or disabled.

When the SIP NAT's route home proxy field is set to forced, it forces the Request to be forwarded to the home proxy without using a local policy. When this option is set to either disabled or enabled and the Request-URI matches the external address of the SIP NAT, the local policy is used.

However, the local policy only replaces the Request-URI when the original Request-URI matches the SBC's IP address or hostname. This behavior is in accordance with that described in RFC 3261. The original Request-URI will be the home proxy address value (the home address of the SIP NAT into the backbone) and not the Oracle Communications Session Border Controller (SBC) address.

Using strict routing, the Request-URI would be the next hop, but the message would also include a Route header with the original Request-URI. With loose routing, the Request-URI remains unchanged and the next hop value is added as the top Route header.

Sometimes the next hop field value must replace the Request-URI in the outgoing request, even if the original Request-URI is not the SBCC. To accomplish this, an option has been added to the local policy that causes the next hop value to be used as the Request-URI and prevents the addition of Route headers. This option is the replace uri value in the local policy.

The following table lists the policy attributes for the local policy:

Parameter	Description
next hop	IP address of your internal SIP proxy. This value corresponds to the IP address of the network interface associated with the SIP proxy.
realm	Number of the port associated with the SIP port.
replace uri	Stores the transport protocol used for sending an receiving signaling messages associated with the SIP port.
allow anonymous	Indicates whether this SIP port allows anonymous connections from session agents.

> **Note:**
>
> You should also define the ping method intervals for the session agents so that the SBC can detect when the agents are back in service after failure.

SIP Server Redundancy Configuration

To enable replace URI:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements. The system prompt changes.

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

3. Type **local-policy** and press Enter. The system prompt changes.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **policy-attributes** and press Enter. The system prompt changes.

   ```
   ORACLE(local-policy)# policy-attributes
   ORACLE(local-policy-attributes)#
   ```

   From this point, you can configure policy attributes for the local policy. To see all local policy attribute options, enter a **?** at the system prompt.

5. **action**—Set this parameter to **replace-uri**, which causes NAT rules such as realm prefixing to be overridden. The default value is **none**. Valid values are:

   - none | replace-uri | redirect

     The replace URI field allows you to indicate whether the local policy's value is used to replace the Request-URI in outgoing requests. This boolean field can be set to either enabled or disabled.

Administratively Disabling a SIP Registrar

The Oracle Communications Session Border Controller's registration cache feature is commonly used to support authorization. It also allows the Oracle Communications Session Border Controller to respond directly to SIP REGISTER requests from endpoints rather than forwarding every REGISTER message to the Registrar(s). In the Oracle Communications Session Border Controller, Registrars are frequently configured as session agents, and an association between each endpoint and its Registrar is stored with the registration cache information.

In Release 4.0.1 and later, the i**nvalidate-registrations** parameter in the session agent configuration enables the Oracle Communications Session Border Controller to detect failed Registrar session agents and automatically forward subsequent REGISTER requests from endpoints to a new Registrar. You can now perform the same behavior manually through a new ACLI command. When you use this command, the Oracle Communications Session Border Controller acts as though the registrations have expired.

For each SIP session agent, you can enable the manual trigger command, and then use the command from the main Superuser ACLI prompt. The **reset session-agent** command provides a way for you to send a session agent offline. Session agents can come back online once they send 200 OK messages the Oracle Communications Session Border Controller receives successfully.

Without using the manual trigger, session agents can go offline because of they do not respond to pings or because of excessive transaction timeouts. However, you might not want to use these more dynamic methods of taking session agents out of service (and subsequently invalidating any associated registrations). You can disable both of these mechanisms by setting the following parameters to 0:

- **ping-interval**—Frequency (amount of time in seconds) with which the Oracle Communications Session Border Controller pings the entity the session agent represents)
- **ttr-no-response**—Dictates when the SA (Session Agent) should be put back in service after the SA is taken OOS (Out Of Service) because it did not respond to the Oracle Communications Session Border Controller

However, you can still use the new SIP manual trigger even with these dynamic methods enabled; the trigger simply overrides the configuration to send the session agent offline.

Considerations for Implicit Service Route Use

When implicit service route support is enabled for a SIP interface (in IMS applications), the Oracle Communications Session Border Controller stores the Service Route URIs from the Service-Route headers that are included in 200 OK responses to REGISTER messages. Subsequently, and even when a session agent is rendered invalid, re-REGISTER messages follow the route stored in the cache instead of using the one defined in the Oracle Communications Session Border Controller.

However, you might not want to use this behavior when you send session agents offline. If you instead want use the route defined in the Oracle Communications Session Border Controller, then you need to configure the SIP interface option called **route-register-no-service-route**.

Manual Trigger Configuration

This section shows you how to enable the manual trigger for sending session agents out of service, and how to then use the trigger from the command line. This section also shows you how to verify that you have successfully put a session agent out of service.

To enable a SIP session agent to manually trigger it to go out of service:

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

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

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

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

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

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

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

4. **invalidate-registrations**—Set this parameter to enabled if you want to use the manual trigger to send this session agent offline (and therefore invalidate the registrations associated with it). The default is disabled.
5. Save and activate your configuration.

   To use the manual trigger that sends session agents offline:

6. Note the hostname value (typically the IP address of the endpoint) for the session agent you want to put out of service. You use this name as an argument in the ACLI command to use the manual trigger.
7. At the Superuser prompt, type **reset session-agent**, a Space, and the hostname value for the session agent. The press Enter.

   ```
   ORACLE# reset session-agent 192.168.20.45
   ```

   If you enter a session agent that does not exist, the system notifies you that it cannot carry out the reset.

Manual Trigger Confirmation

To confirm that a session agent has been sent offline:

- Use the **show sipd endpoint-ip** command to confirm the session agent state.

  ```
  ACMEPACKET# show sipd endpoint-ip 1016
  User <sip:1016@172.18.1.80>
          Contact exp=3582
  ```

```
            UA-Contact: <sip:1016@192.168.1.132:5060> UDP
                realm=access local=172.18.1.132:5060
UA=192.168.1.132:5060
    SD-Contact: <sip:1016-o3badgbbnjcq5@172.18.2.80:5060> realm=core
    Call-ID: 1-7944@192.168.1.132'
    SR=172.18.2.92
    SA=172.18.2.93
    Service-Route='<sip:test@s-cscf::5060;orig;lr>'
ACMEPACKET# reset session-agent 172.18.2.92
Accepted
Reset SA failover timer
ACMEPACKET# show sipd endpoint-ip 1016
User <sip:1016@172.18.1.80>
        Contact <invalidated> exp=3572
            UA-Contact: <sip:1016@192.168.1.132:5060> UDP
                realm=access local=172.18.1.80:5060
UA=192.168.1.132:5060
    SD-Contact: <sip:1016-o3badgbbnjcq5@172.18.2.80:5060> realm=core
    Call-ID: 1-7944@192.168.1.132'
    SR=172.18.2.92 (failed 2 seconds ago)
    SA=172.18.2.93
    Service-Route='<sip:test@s-cscf::5060;orig;lr>'
ACMEPACKET#
```

In the above ACLI example the first iteration of the **show sip endpoint-ip** command provides information for the in-service 172.18.2.92 session agent; the second command iteration displays information for the now out-of-service session agent.

> **Note:**
>
> There can be multiple users registered with the same phone number. In those cases, the **show sipd endpoint-ip <phone-number>** output is a single entry which first matches the registration cache.

Surrogate Agent Refresh on Invalidate

Surrogate agent registrations normally only re-register when nearing their expiration time. When a registrar fails, the surrogate agent will wait until the expiration time to refresh the registration with an in-service registrar.

You can configure your Oracle Communications Session Border Controller to immediately refresh the surrogate agent registrar with an in-service registrar by enabling the existing parameter **invalidate-registrations**.

Invalidate Registrations

An existing feature called **invalidate-registrations** located in the session agent keeps track of when surrogate agents go out of service. When REGISTER messages are received, registration entries that had out-of-service session agents since the last REGISTER will always allow the message through to the registrar (as opposed to responding directly from the cache).

The **invalidate-registrations** parameter in session agent configuration enables the Oracle Communications Session Border Controller to detect failed Registrar session agents.

If invalidate-registrations is enabled for the session agent, a response from a surrogate REGISTER that contains a service-route header that corresponds to a session-agent is installed to the registration cache entry.

The surrogate-agents are scanned. Surrogate agents with registration entries matching the out-of-service registrar have their timer reset to initiate a refresh. For an immediate refresh, the registration entry will only be considered when the service-route session agent goes out-of-service. The service-route session agent takes precedence and any previous registrar session agent w ill not be considered for an immediate refresh of the surrogate-agent registration.

Performance Impact

In cases with a large number of surrogate-agent registrations, there may be an impact to CPU usage when a session-agent goes out-of-service. All of the surrogate-agent registrations are scanned at that time. Refresh registrations are then sent out on timers.

Media Inactivity Timer Configuration

To disable the media inactivity timer for calls placed on hold:

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

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

2. Type **session-router** and press Enter.

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

3. Type **session-agent** and press Enter.

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

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

4. **invalidate-registration**—Set this parameter to **enabled** if you want to use the manual trigger to send this session agent offline (and therefore invalidate the registrations associated with it). The default is **disabled**.
5. Save and activate your configuration.

Support for Encoded Multipart Message Bodies

SIP messages and responses may arrive at the Oracle Communications Session Border Controller with encoded multipart message bodies, such that the content of the body is unreadable. This information may be encoded for the purpose of compressing the data. Normally, the Oracle Communications Session Border Controller would consider the body invalid and reject the entire message, replying to the sender with a `400 Invalid Body` error response. The user, however, can configure the **sip-config** option, **proxy-content-type-encodings**, allowing the Oracle Communications Session Border Controller to accept, process and forward messages containing these encoded parts. This configuration causes the Oracle Communications Session Border Controller to ignore the encoding, identify the end of the

message via content length, and pass the message towards its intended recipient with the multipart body fully encoded.

The condition that triggers this functionality is the Oracle Communications Session Border Controller recognizing the presence of a multipart message body and the definition of the encoding type within the message.

```
NOTIFY sip:user@example.com SIP/2.0
Via: SIP/2.0/TCP
...
Content-Type: multipart/mixed;boundary="imdn-boundary"
Content-Encoding: gzip

... Encoded multipart content ...
```

When configured, the **proxy-content-type-encodings** is simply a list of strings. The Oracle Communications Session Border Controller looks to match the string defining the content encoding with a string in the list to proceed with the functionality.

The Oracle Communications Session Border Controller only performs this procedure on messages encoded with types for which it is configured and that are properly formed. Examples of when the Oracle Communications Session Border Controller does not perform this procedure include:

- The Oracle Communications Session Border Controller receives a message with an encoded multipart message block, but **proxy-content-type-encodings** list is empty. In this case, the Oracle Communications Session Border Controller responds with a `415 Unsupported Media Type`.
- The message arrives with a multipart message body with encoding for which the **proxy-content-type-encodings** is configured, but there is no terminating boundary. In this case, the Oracle Communications Session Border Controller replies with a `400 Invalid Body error`.
- The Oracle Communications Session Border Controller receives a response with an encoded multipart message block, but **proxy-content-type-encodings** list is empty. In this case, the Oracle Communications Session Border Controller simple drops the response.

Multipart Message Encoding Support Configuration

The procedure below provides the steps needed to configure the Oracle Communications Session Border Controller for multipart message encoding support.

To have your Oracle Communications Session Border Controller proxy messages despite the presence of the specified multipart message encoding:

1. In Superuser mode, use the following command sequence to access the **sip-config** element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   From this point, you can configure encoding support.

2. Configure the desired encoding types using the option followed by comma-separated coding types.

   The example below configures support for gzip and compressed encoding.

   ```
   ORACLE(sip-config)# options +proxy-content-type-encodings=gzip,compress
   ```

3. Type **done** and exit configuration mode. Save and activate your configuration.

SIP Distributed Media Release

This section explains how to configure distributed media release (DMR). SIP DMR lets you choose whether to include multi-system (multiple Oracle Communications Session Border Controllers) media release information in SIP signaling requests sent into a specific realm.

Overview

The SIP DMR feature lets RTP/RTCP media be sent directly between SIP endpoints (for example, SIP phones or user agents) without going through a Oracle Communications Session Border Controller; even if the SIP signaling messages traverse multiple Oracle Communications Session Border Controllers. It encodes IPv4 address and port information for the media streams described by the media, for example SDP.

With SIP DMR, the media realm and IPv4 address and port information from the UA's SDP is encoded into SIP messages (either in the SIP header or in the SDP) as they enter the backbone network. The information is decoded by a Oracle Communications Session Border Controller from SIP messages that come from the backbone network. The decoded address and port information is put into the SDP sent the UAs in the access (private/customer) network.

This functionality lets the RTP/RTCP flow directly between the UAs in the access network without traversing the Oracle Communications Session Border Controllers and without passing into the backbone network. The media can then flow directly between the two SIP endpoints in the same network, if it is serviced by multiple Oracle Communications Session Border Controllers.

You can enable this feature on a per-realm basis and multiple realms can be supported.

Endpoint Locations

You can configure the Oracle Communications Session Border Controller to release media when the source and destination of the call are in the same network, customer VPN, or customer LAN. In architectures that use DMR, the Oracle Communications Session Border Controller is only part of the media path for traffic that originates and terminates in different networks.

If configured to do so, the Oracle Communications Session Border Controller can release media:

- Between endpoints supported by a single Oracle Communications Session Border Controller
  In the same network/VPN

  In the same network behind the same NAT/firewall
- Between endpoints supported by multiple distributed Oracle Communications Session Border Controllers
  In the same network/VPN

Location of the Encoded Information

Encoded media release information can appear in three different places:

- SDP attribute
  Media release data can be encoded into an SDP attribute in the SIP message body (for example, media-release=sdp;acme-media). The encoded data is placed into an acme-media attribute in the SDP:

  ```
  a=acme-media:<encoded-media-interface-info>
  ```

- SIP header parameter
  Media release data can be placed in a header parameter of a SIP header (for example, media-release=Contact;acme-media). The encoded data is placed into an acme-media parameter in the Contact header:

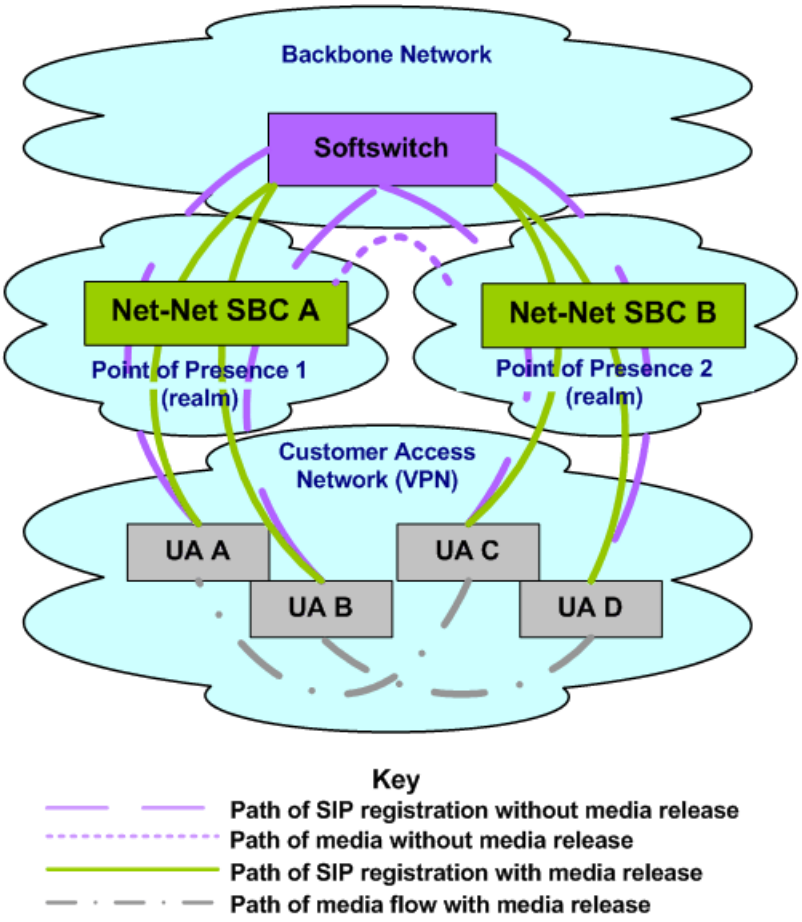
  ```
  Contact: <sip:1234@abc.com>;acme-media=<encoded-media-interface-info>
  ```

- SIP header
  Media release data can appear in a SIP header (for example, media-release=P-Media-Release). The encoded data is placed into a P-Media-Release header:

  ```
  P-Media-Release: <encoded-media-interface-info>
  ```

Example Distributed Media Release

The following example shows the network diagram for DMR in a multiple-site VPN environment supported by multiple, distributed Oracle Communications Session Border Controllers.



As shown in the network diagram, UA A and UA B register with the softswitch through Oracle Communications Session Border Controller A while UA C and UA D register with the softswitch through Oracle Communications Session Border Controller B. Without DMR, the media for calls between UA A/UA B and UA C/UA D is steered through both Oracle Communications Session Border Controller A and Oracle Communications Session Border Controller B.

With SIP DMR, the media realm and IPv4 address and port information from the UA's Session Description Protocol (SDP) is encoded into SIP messages (either in the SIP header or in the SDP) as they enter the backbone (public/service provider) network. The information is decoded from SIP messages that come from the backbone network. The decoded address and port information is put into the SDP sent to the UAs in the access (private/customer) network. This functionality allows for the RTP/RTCP to flow directly between the UAs in the access network without traversing the Oracle Communications Session Border Controllers and without passing into the backbone network.

Overview of SIP DMR Configuration

To configure SIP DMR:

1. Edit the SIP config element's option field.

   The media-release="<header-name>[;<header-param>]" option defines how the SIP distributed media release feature encodes IPv4 address and port information. If the media-release parameter is configured in the options field but no header is specified, the parameter value of P=Media-Release will be used. This parameter is optional and is not configured by default.

2. Enable SIP DMR for the entire realm by setting the realm config element's msm release field to enabled.

   The media IPv4 address and port information is encoded into outgoing SIP messages and decoded from incoming SIP messages for all of the realms (in each realm-config element) with which the SIP distributed media release will be used.

> **Note:**
>
> You can also use the realm config element's mm in network field to release the media back to a connected network that has multiple realms. This field is not specific SIP distributed media release and it is not required for the SIP DMR to work. However, if this field is set to enabled and the ingress and egress realms are part of the same network interface, it lets the Oracle Communications Session Border Controller release the media.

SIP DMR Configuration

To configure media release:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. Type **options** followed by a Space.
5. After the Space, type the media release information in the following format:

   ```
   media-release="<header-name>[;<header-param>]"
   ```

   - header-name either refers to the SIP header in which to put the information or to the special header-name value of sdp to indicate the information should be put into the SDP.
   - parameter-name refers to the header parameter name in which to put the information or, in the case of the special header-name value of sdp, to the SDP attribute name in which to put the information.

     For example:

     ```
     ORACLE(sip-config)# options media-release=P-Media-Release
     ```

6. Press Enter.

> **Note:**
>
> If the media-release parameter is configured in the options field, but no header is specified, then the parameter value of P-Media-Release will be used. P-Media-Release is a proprietary header and means that the media will be encoded in the SIP header with this name.

The following example shows where the encoded information (for example, SDP data) is passed.

```
media-release="P-Media-Release"
media-release="Contact;acme-media"
media-release="sdp;acme-media"
```

Configuring the Realm

You need to set the each realm config element's msm release field to enabled for all the realms for which you want to use SIP DMR.

Although the mm in network field is not specific to the SIP distributed media release feature, it can be used to release the media back to a connected network that has multiple realms. This field does not need to be configured in order for the SIP distributed media release feature to work. However, if this field is set to enabled and the ingress and egress realms are part of the same network interface, it lets the Oracle Communications Session Border Controller release the media.

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

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

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

   From this point, you can configure realm parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

4. **msm-release**—Enable DMR within this realm on this Oracle Communications Session Border Controller. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Repeat for each realm on which you want to enable DMR.

Add-On Conferencing

This section explains how to configure the add-on conferencing functionality. It also includes a description of the SIP B2BUA functionality related to the SIP add-on conferencing. This description includes information about Contact header mapping and processing and Refer-to header processing.

Overview

SIP add-on conferencing lets you:

- Use the Oracle Communications Session Border Controller's add-on conferencing feature for network architectures in which the conference initiator is located on a different network than that of the media server.
- Configure the Oracle Communications Session Border Controller to enable Contact header mapping for the Refer-To header.

Caveats

The following caveats are associated with add-on conferencing:

- Contact header mapping is not replicated on the standby Oracle Communications Session Border Controller in an HA Oracle Communications Session Border Controller pair architecture.
- Upon switchover, any conferences in progress remain in progress, but no new parties can be invited to or join the conference.
- By default, the Oracle Communications Session Border Controller does not map SIP Contact headers for reasons of performance.

Add-On Conferencing Scenario

The add-on conferencing scenario described in the following example applies to a network architecture involving the Oracle Communications Session Border Controller and a media server that is located on a different network from the other conference participants. In this scenario, the Oracle Communications Session Border Controller resides on a standalone network that connects two additional, separate networks.

Some network architectures have a media server on a different network from the one on which the phones reside. In this scenario, all requests and/or responses going from the phones (Phone A, Phone B, or Phone C) to Media Server D and vice versa are translated according to their corresponding SIP-NAT. All headers subjected to NAT are encoded and decoded properly as they traverse the Oracle Communications Session Border Controller, except for the Contact header. This exception occurs because the SIP process on the Oracle Communications Session Border Controller runs as a SIP B2BUA and not as a SIP proxy.

The SIP B2BUA re-originates the Contact headers of the User Agents (UAs) participating in SIP sessions with local Contact headers to make sure that they receive all future in-dialog requests. For an in-dialog request, the B2BUA can identify the dialog and find the Contact URI of the other leg of the call.

The Oracle Communications Session Border Controller add-on conferencing feature applies to situations when the Contact URI is used in another dialog. In such a case, the SIP B2BUA will not be able to find the correct dialog that retrieves the correct Contact URI of the other leg if it needs to replace the Contact URI.

Using the SIP add-on conferencing, the SIP B2BUA on the Oracle Communications Session Border Controller can map the Contact headers it receives to the Contact headers it creates. It can also convert the Refer-To URI to the correct value required for forwarding the REFER request.

SIP B2BUA Functionality

This section describes the role of the Oracle Communications Session Border Controller's SIP B2BUA in the add-on conferencing scenario that requires Contact header mapping for the Refer-To header.

When the Oracle Communications Session Border Controller starts up, the SIP B2BUA reads and parses the list of options in the SIP configuration. If the refer to uri prefix is an appropriate value (it is not an empty string), the Oracle Communications Session Border Controller will have a text prefix value the media server can use to denote a conference ID in its Contact header. With this information, the SIP B2BUA sets up a Contact header mapping.

You configure the Oracle Communications Session Border Controller to enable Contact header mapping for the Refer-To header by editing the SIP config options parameter. The SIP B2BUA on the Oracle Communications Session Border Controller can then map the Contact headers it receives to the Contact headers it creates.

Contact Header Processing

The Contact header mapping matches a Contact header that contains the refer to URI prefix to the corresponding Contact header that the Oracle Communications Session Border Controller's SIP B2BUA re-originates. Contact headers that do not contain the refer to URI prefix are not mapped (so that performance of the Oracle Communications Session Border Controller is minimally affected).

Only the Contact header in an INVITE request and its 200 OK response are checked for the refer to URI prefix and added to the Contact header mapping. Contact headers appearing in other SIP requests/responses are not checked.

Target Mapping and Conferences

If the Oracle Communications Session Border Controller is configured to enable Contact header mapping for the Refer-To header, then Contact header target maps are established for each individual call. The Oracle Communications Session Border Controller's SIP B2BUA uses these maps to allow the media server to connect the conference initiator with the conferenced-in parties.

Prior to terminating the call (hanging up), the conference initiator can contact other parties and invite those additional parties to join the conference. These other parties can join the existing conference because the target mapping for the conference is still in effect on the Oracle Communications Session Border Controller.

Once the conference initiator hangs up, the Oracle Communications Session Border Controller discards the mapping from the conference.

Refer-To Header Processing

When a Refer-To header is present in a REFER request that arrives at the SIP B2BUA after the incoming request is properly translated according to its SIP-NAT, the SIP B2BUA follows these steps:

1. The SIP B2BUA parses the Refer-To URI.

2. If the user part of the Refer-To URI contains the refer to URI prefix, the SIP B2BUA searches the Contact header mapping for a match of the user part of the URI.

   If the user part of the Refer-To URI does not contain the refer to URI prefix, the SIP B2BUA leaves the existing Refer-To URI unchanged.

3. If the user part of the Refer-To URI contains the refer to URI prefix and a match of the Refer-To URI is found, the SIP B2BUA replaces the existing Refer-To URI with the URI of the corresponding Contact URI stored in the matched record. This replacement enables the NAT function to properly decode the replacement URI and change it back to the form originally received by the Oracle Communications Session Border Controller. As a result, the correct conference ID is restored in the Refer-To header prior to the request being sent to its next hop.

   If the user part of the Refer-To URI contains the refer to URI prefix but a matched URI cannot be found, the SIP B2BUA will leave the existing Refer-To URI unchanged and will write a WARNING level log message to record the failure.

Add-on Conferencing Configuration

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. Type **options** followed by a Space.

5. After the Space, type the add-on conferencing information in the following format:

   ```
   options refer-to-uri-prefix=”conf=”
   ```

   For example:

   ```
   ORACLE(sip-config)# options refer-to-uri-prefix=”conf”
   ```

6. Press Enter.

SIP REFER Method Call Transfer

The Oracle Communications Session Border Controller automatically converts a received REFER method into an INVITE method, thus allowing the Oracle Communications Session Border Controller to transfer a call without having to proxy the REFER back to the other UA.

This function can be configured for a specified realm or session agent. When both elements have the SIP REFER method call transfer functionality configured, the session-agent configuration takes precedence over realm-config. Furthermore, this function only works from a realm-config when the REFER request is from an end point that is not configured as a session-agent.

The Oracle Communications Session Border Controller has a configuration parameter giving it the ability to provision the handling of REFER methods as call transfers. The parameter is called refer-call-transfer. When this feature is enabled, the Oracle Communications Session Border Controller creates an INVITE message whenever it receives a REFER. The Oracle Communications Session Border Controller sends this INVITE message to the address in the Refer-To header. Included in the INVITE message is all the unmodified information contained in the REFER message. The previously negotiated codec is also still used in the new INVITE message. NOTIFY and BYE messages are sent to the UA upon call transfer completion.

If a REFER method is received containing no Referred-By header, the Oracle Communications Session Border Controller adds one, allowing the Oracle Communications Session Border Controller to support all call agent screen applications.

In addition, the SIP REFER method call transfer feature supports the following:

- Both unattended and attended call transfers
- Both successful and unsuccessful call transfers
- Early media from the Referred-To party to the transferee
- REFER method transfer from different sources within the destination realm
- The REFER event package as defined in RFC 3515. This applies for situations where multiple REFER methods are used within a single dialog.
- Third party initiated REFER method signalling the transfer of a call by associating the REFER method to the dialogue via the REFER TargetDialog.
- The Referred-To party can be both in a different realm (and thus a different steering pool) from the referrer, and in the same realm
- The associated latching should not prohibit the Referred-To party from being latched to while the referee is still sending media.

Unsuccessful Transfer Scenarios

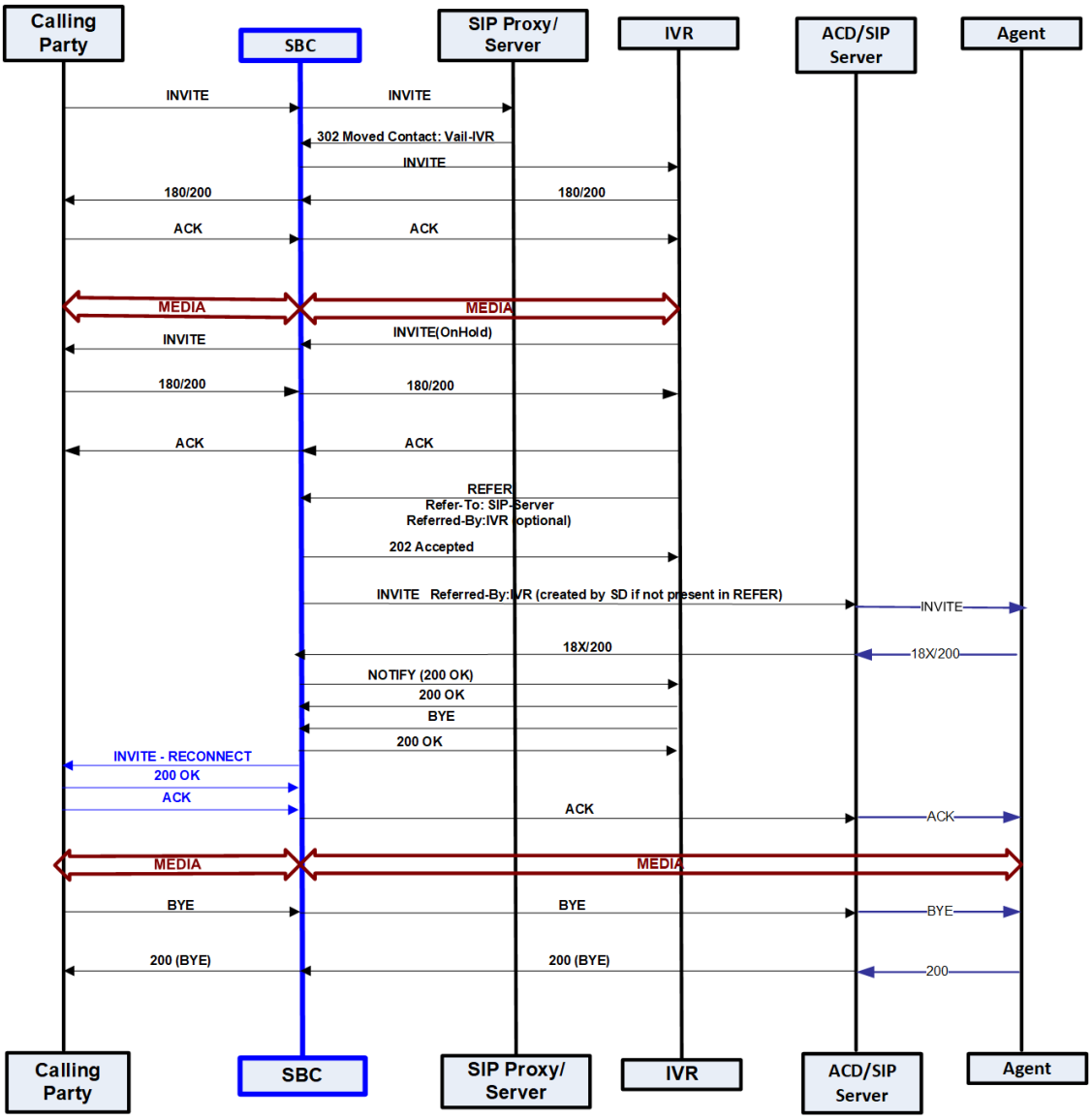
The Oracle Communications Session Border Controller does not successfully handle the following failed, unusual, and unexpected transfer scenarios:

- The new INVITE to the Referred-To party gets challenged, the Oracle Communications Session Border Controller does not answer the challenge. It is treated with the 401/407 response just as any other unsuccessful final response.
- The header of the REFER message contains a method other than INVITE or contains URI-parameters or embedded headers not supported by the Oracle Communications Session Border Controller.
- The Oracle Communications Session Border Controller shall allow the Referred-To URI that happens to resolve to the same next-hop as the original INVITE went to, to do so.
- The Oracle Communications Session Border Controller ignores any MIME attachment(s) within a REFER method.
- The Oracle Communications Session Border Controller recurses (when configured to do so) when the new INVITE sent to the Referred-To party receives a 3xx response.

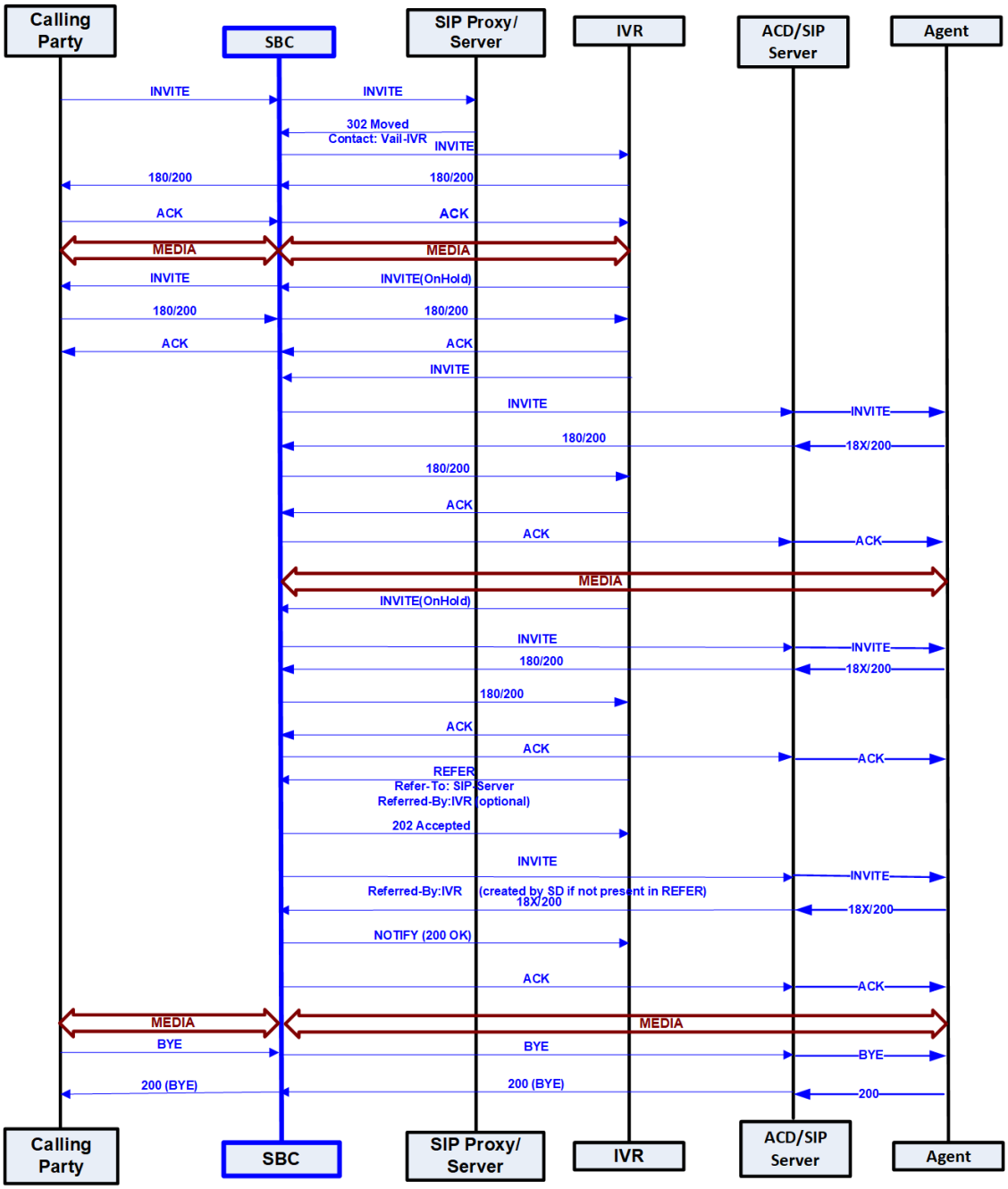
- The transferee indicated support for 100rel, and the original two parties agreed on using it, yet the Referred-To party does not support it.
- The original parties negotiated SRTP keys.
- The original parties agreed on a codec using a dynamic payload type, and the Referred-To party happens to use a different dynamic payload number for that codec.

Call Flows

The following is an example call flow for an unattended call transfer:



The following is an example call flow of an attended call transfer:



SIP REFER Method Configuration

To enable SIP REFER method call transfer in the realm-config:

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

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

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

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

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

4. **refer-call-transfer**—Set to **enabled** to enable the refer call transfer feature. The default for this parameter is **disabled**.
5. Save and activate your configuration.

   To enable SIP REFER method call transfer in the session-agent:

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

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

7. Type **session-router** and press Enter.

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

8. Type **session-agent** and press Enter.

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

9. **refer-call-transfer**—Set to **enabled** to enable the refer call transfer feature. The default for this parameter is **disabled**.
10. Save and activate your configuration.

REFER-Initiated Call Transfer

The Oracle Communications Session Border Controller supportes REFER-initiated call transfer either by proxying the REFER to the other User Agent in the dialog, or by terminating the received REFER and issuing a new INVITE to the referred party. These static alternate operational modes can be configured for specific SIP interfaces, realms, or session agents.

An additional operational mode can determine on a call-by-call basis whether to proxy the REFER to the next hop, or terminate the REFER and issue an INVITE in its stead.

The configuration parameter **dyn-refer-term**, and **refer-call-transfer** parameter specify call transfer modes.

With the **refer-call-transfer** parameter set to **disabled** (the default), all received REFERs are simply proxied to the peer User Agent.

With the **refer-call-transfer** parameter set to **enabled**, the Oracle Communications Session Border Controller terminates all REFERs, generates a new INVITE, and sends the INVITE to the address in the Refer-To header.

With the **refer-call-transfer** parameter set to **dynamic**, the Oracle Communications Session Border Controller determines REFER handling on a call-by-call basis as follows:

1. Check the **refer-call-transfer** value for the session agent from which the REFER was received, or for ingress realm (the realm that received the REFER).

   If the value is disabled, proxy the REFER to the peer User Agent, to complete REFER processing.

   If the value is enabled, terminate the REFER and issue an new INVITE to the referred party, to complete REFER processing.

   If the value is dynamic, identify the next hop egress realm.

2. Check the **dyn-refer-term** value for the next hop egress realm.

   If the **dyn-refer-term** value is disabled (the default), proxy the REFER to the next hop to complete REFER processing.

   If the **dyn-refer-term** value is enabled, terminate the REFER and issue an new INVITE to the referred party to complete REFER processing.

Supported Scenarios

In the basic scenario for REFER initiated call transfer, a call is established between two User Agents (Alice and Bob). User Agent Bob then sends a REFER request to transfer the call to a third User Agent Eva. With dynamic call-transfer enabled, the Oracle Communications Session Border Controller (SBC) prevents the REFER from being sent to Alice and generates the INVITE to Eva.

If the INVITE to Eva succeeds, the SBC sends a re-INVITE to Alice modifying the SIP session as described in Section 14 of RFC 3261, *SIP: Session Initiation Protocol*. At this point the SBC cancels the original dialog between the SBC and Bob.

If the INVITE to Eva fails, call disposition depends on whether or not Bob issued a BYE after the REFER call transfer. If the Oracle Communications Session Border Controller did receive a BYE from Bob (for instance, a blind transfer), it proxies the BYE to A. Otherwise, the SBC retains the original SIP session and media session, thus allowing Bob to re-establish the call with Alice by sending a re-INVITE. In this case, the SBC sets a timer (32 seconds), after which a BYE will be sent.

If a REFER method is received containing no Referred-By header, the SBC adds one, allowing the SBC to support all call agent screen applications.

In addition, the SIP REFER method call transfer feature supports the following:

- Both unattended and attended call transfers
- Both successful and unsuccessful call transfers
- Early media from the Referred-To party to the transferee
- REFER method transfer from different sources within the destination realm
- The REFER event package as defined in RFC 3515. This applies for situations where multiple REFER methods are used within a single dialog.
- Third party initiated REFER method signalling the transfer of a call by associating the REFER method to the dialogue via the REFER TargetDialog.
- The Referred-To party can be both in a different realm (and thus a different steering pool) from the referrer, and in the same realm

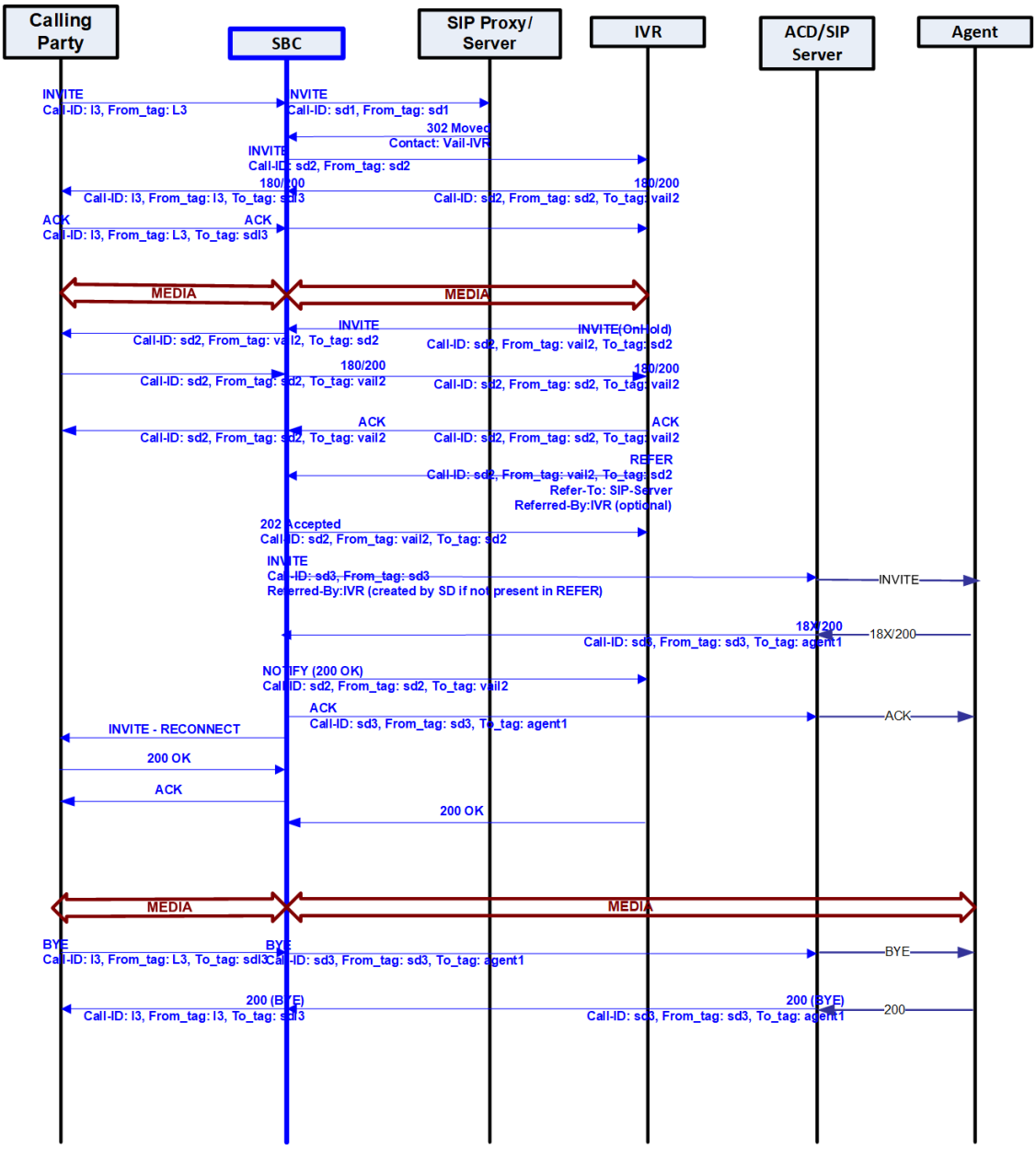
- The associated latching should not prohibit the Referred-To party from being latched to while the referee is still sending media.

The SBC does not successfully handle the following anomalous transfer scenarios:

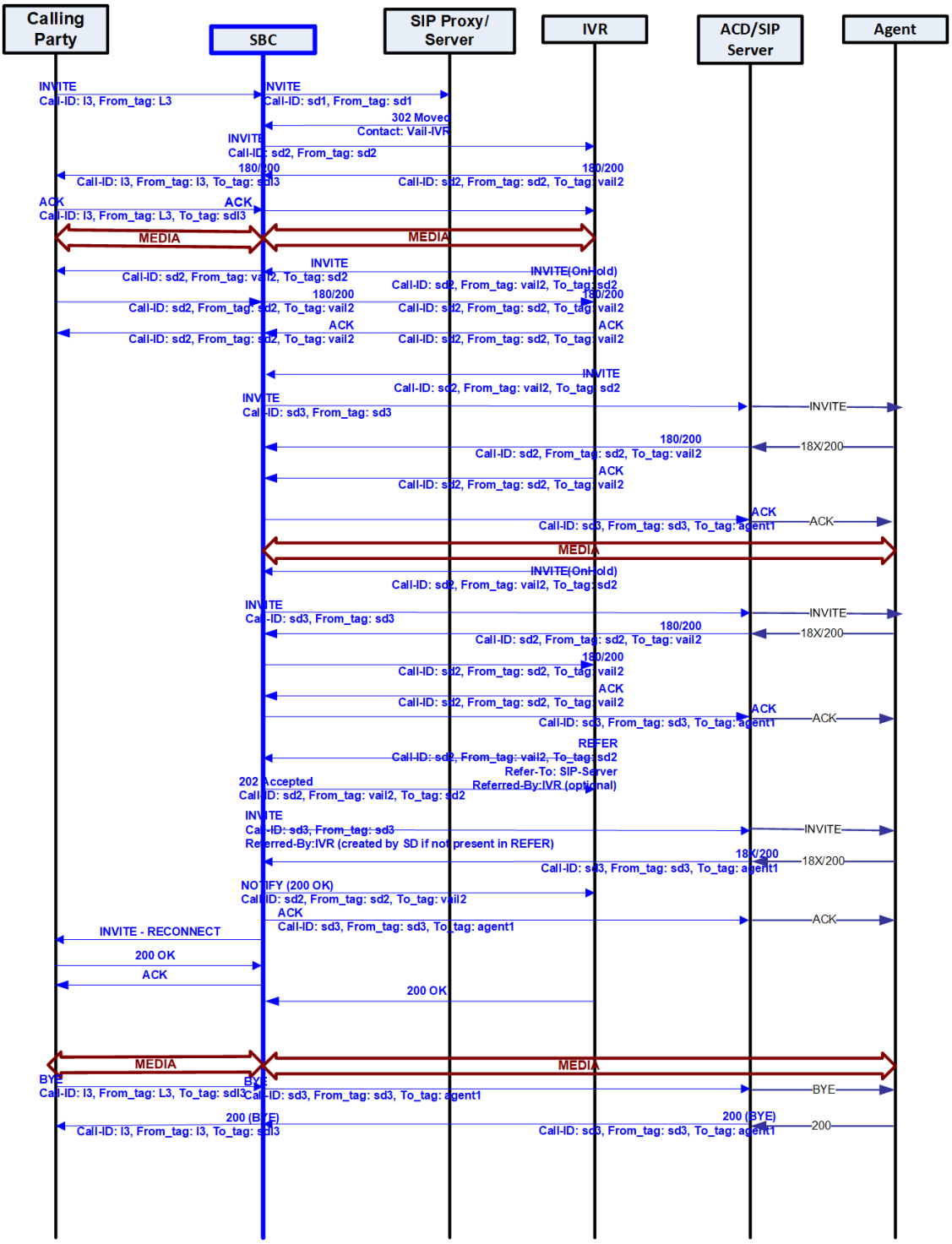
- The new INVITE to the Referred-To party gets challenged — the SBC does not answer the challenge. It is treated with the 401/407 response just as any other unsuccessful final response.
- The header of the REFER message contains a method other than INVITE or contains URI-parameters or embedded headers not supported by the SBC.
- The SBC shall allow the Referred-To URI that happens to resolve to the same next-hop as the original INVITE went to, to do so.
- The SBC ignores any MIME attachment(s) within a REFER method.
- The SBC recurses (when configured to do so) when the new INVITE sent to the Referred-To party receives a 3xx response.
- The transferee indicated support for 100rel, and the original two parties agreed on using it, yet the Referred-To party does not support it.
- The original parties negotiated SRTP keys.

Call Flows

The following is an example call flow for an unattended call transfer:



The following is an example call flow of an attended call transfer:



Maintaining RBT During Transfers

The SBC may stop sending its configured ring back tone (RBT) to the caller when operating within some transfer scenarios. Applicable scenarios include the presence of network infrastructure that issues a BYE from the callee to the SBC while the transfer is underway. You can configure the SBC to persist with RBT for the duration of the transfer process so the caller does not unexpectedly lose RBT. By default, this configuration is not set, which provides for proper transfer behavior within most infrastructures.

Default refer handling processing includes referencing your **refer-call-transfer** configuration, which, when enabled, gives the system the ability to provision the handling of REFER methods as call transfers. When this parameter is enabled, the SBC creates an INVITE message whenever it receives a REFER. It then sends this INVITE to the address in the Refer-To header. The SBC includes all the unmodified information contained in the REFER message in this INVITE, and uses the previously negotiated codec. The SBC sends NOTIFY and BYE messages to the UA upon call transfer completion. This behavior is interrupted if the tranferor sends a BYE because the SBC then tears down the flows between itself and both the caller, which carries the RBT, and the transferor.

You can configure the SBC to avoid interruptions to this local RBT being played towards the ingress side (A party) for call transfer scenarios wherein the SBC receives a BYE from the transferor before a 200 OK from the transferee during the call transfer process. To enable this behavior, you set the **refer-early-bye-enhancement** option in the **sip-config**, as shown below.

```
ORACLE(sip-config)# options +refer-early-bye-enhancement
```

When you include the "+" sign in this syntax, the system allows previously configured options field values to remain operational. If you do not use the "+" sign, the system removes any other configured options leaving only this option operational.

> **Note:**
>
> If you have set this option and then downgrade to a version that does not support it, the system ignores the option setting.

The SBC supports this behavior for HA deployments:

- If the callee triggers the transfer after the switchover, the SBC transfers media using new flows.
- If the HA switchover happens while RBT is ongoing, the SBC does not continue sending RBT on the new active. If the new active receives a refresh REFER/INVITE, it restarts RBT per its normal behavior.

This feature interacts with the following related features:

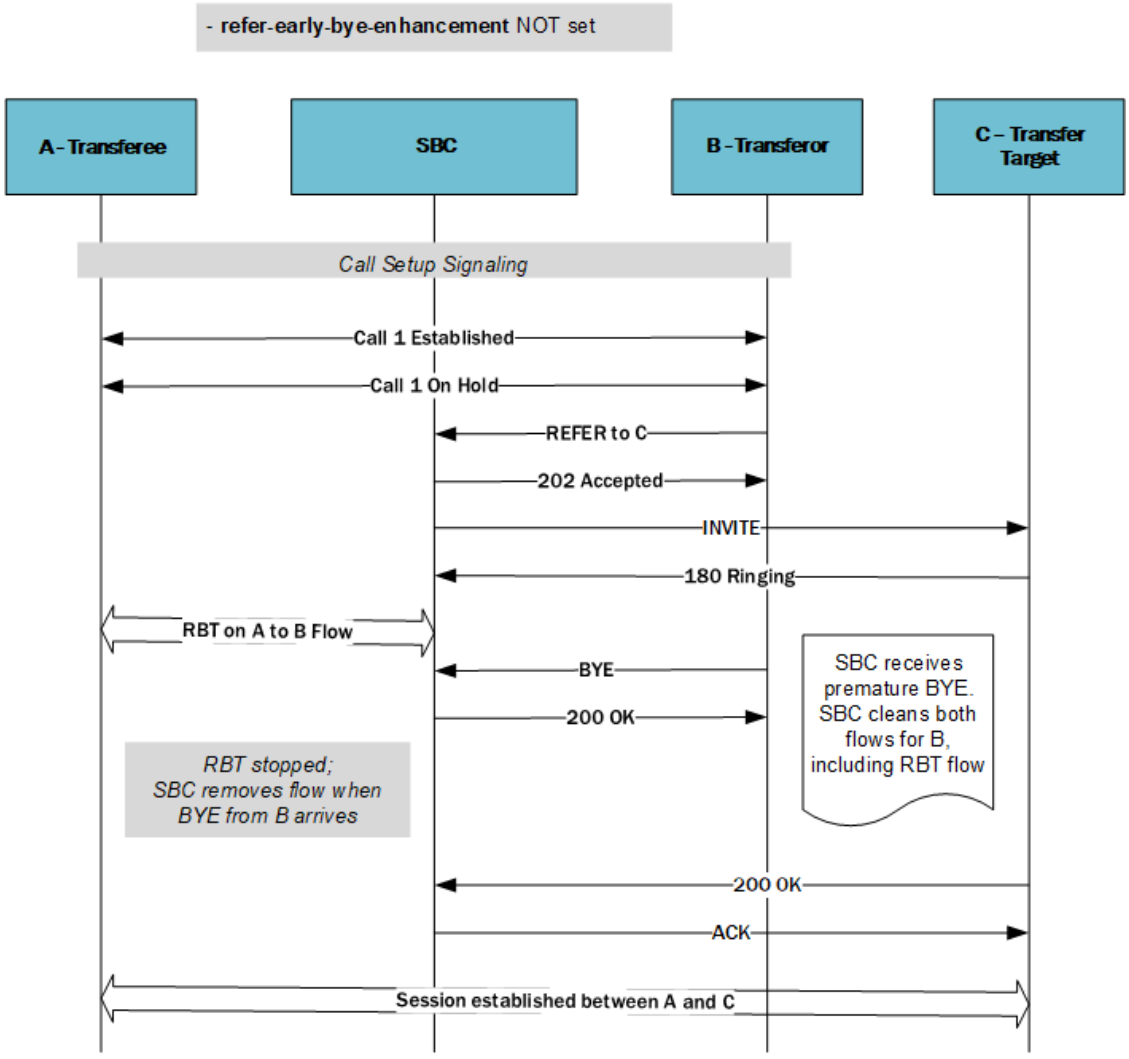
- **refer-call-transfer** Required—This feature is applicable only when the REFER is handled locally by the SBC. You must enable **refer-call-transfer** on the realm or session agent that receives the REFER.
- **refer-reinvite**
  - Enabled—When enabled, this option enables the Refer with Replaces feature on the SBC. This feature is not applicable to scenarios wherein the SBC uses a reINVITE in the existing dialog to complete an attended call transfer. Therefore, this feature requires that you do not enable **refer-reinvite**. These configurations are actually intended to disable RBT on the applicable flows.
  - Disabled—During an attended call transfer scenario with the **refer-reinvite** option disabled, the SBC handles and preserves the media flows towards the transferee side, which allows uninterrupted RBT even if the transferor sends a premature BYE before the Transfer Agent accepts the (200 OK from C party).
    Ultimately, the SBC terminates the flows preserved for RBT whether the attended transfer completed successfully or unsuccessfully.

- **refer-reinvite-no-sdp**—If you enable **refer-reinvite-no-sdp** within the **sip-config**, then the system plays RBT because the SipMedia is retained in SipDialog. In this case, the system does not handle the Premature BYE, instead passing the BYE on to the targets and terminating the calls.

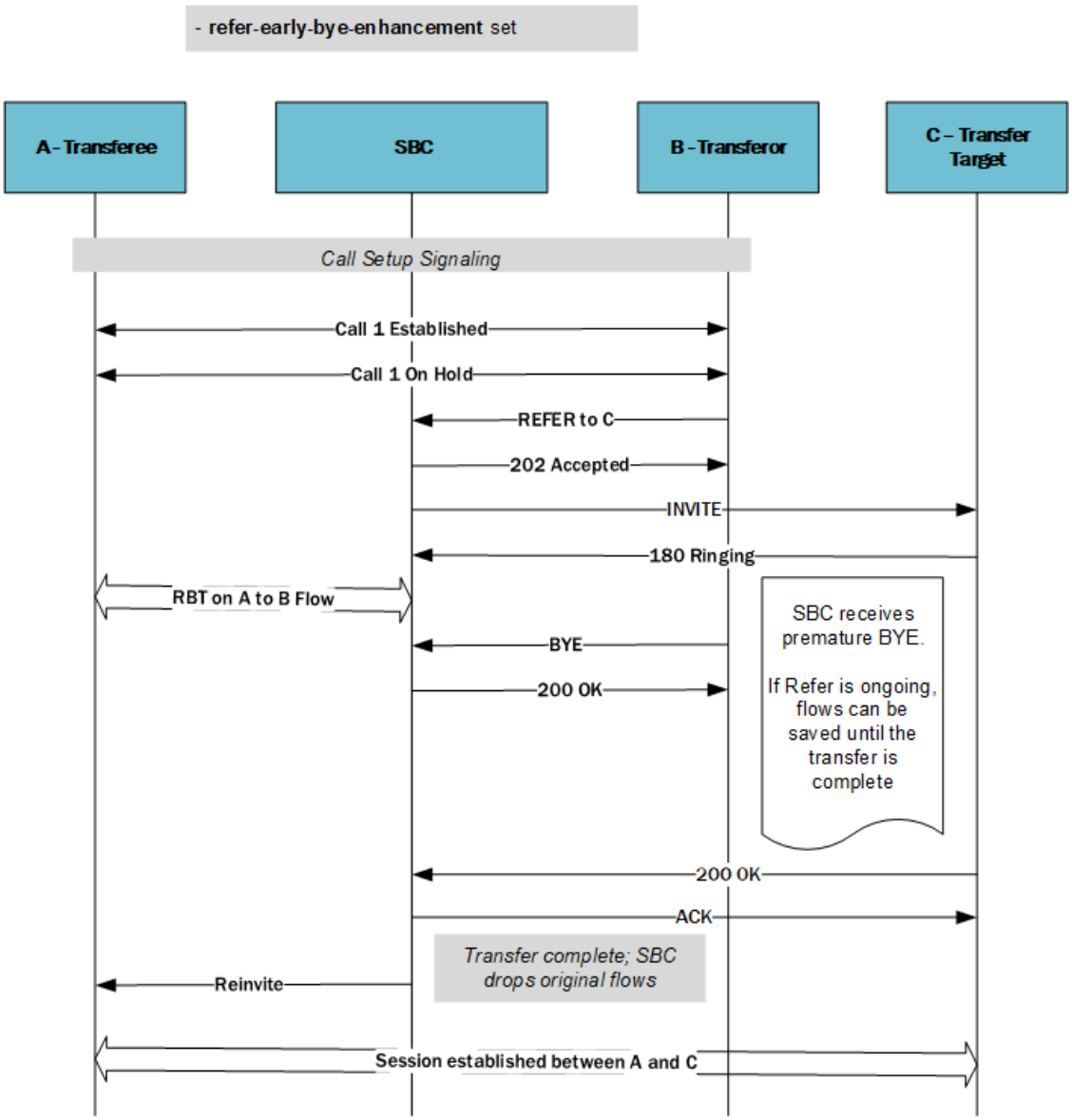
Call Flows

The following call flows depict the SBC attempting to handle RBT during refer scenarios. These images include two scenarios wherein the system does not support, and then does support RBT through the refer processing.

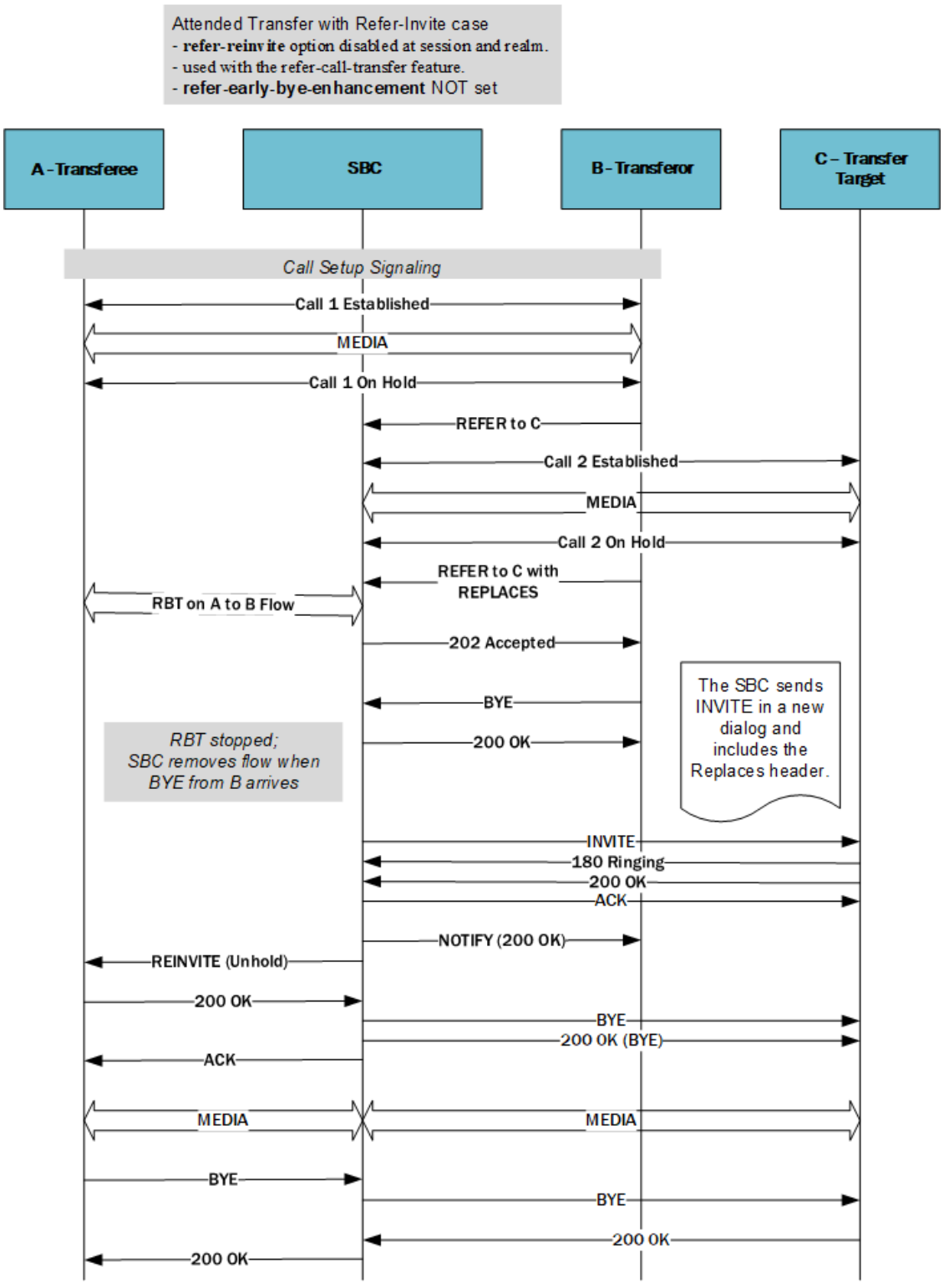
This first scenario is a simple transfer scenarios. Upon receiving a BYE from user agent, the SBC performs its default behavior, which includes the system stopping the RBT and clearing the flow on both sides. This drops the flow from the SBC to A even though it is busy supporting an RBT media flow.



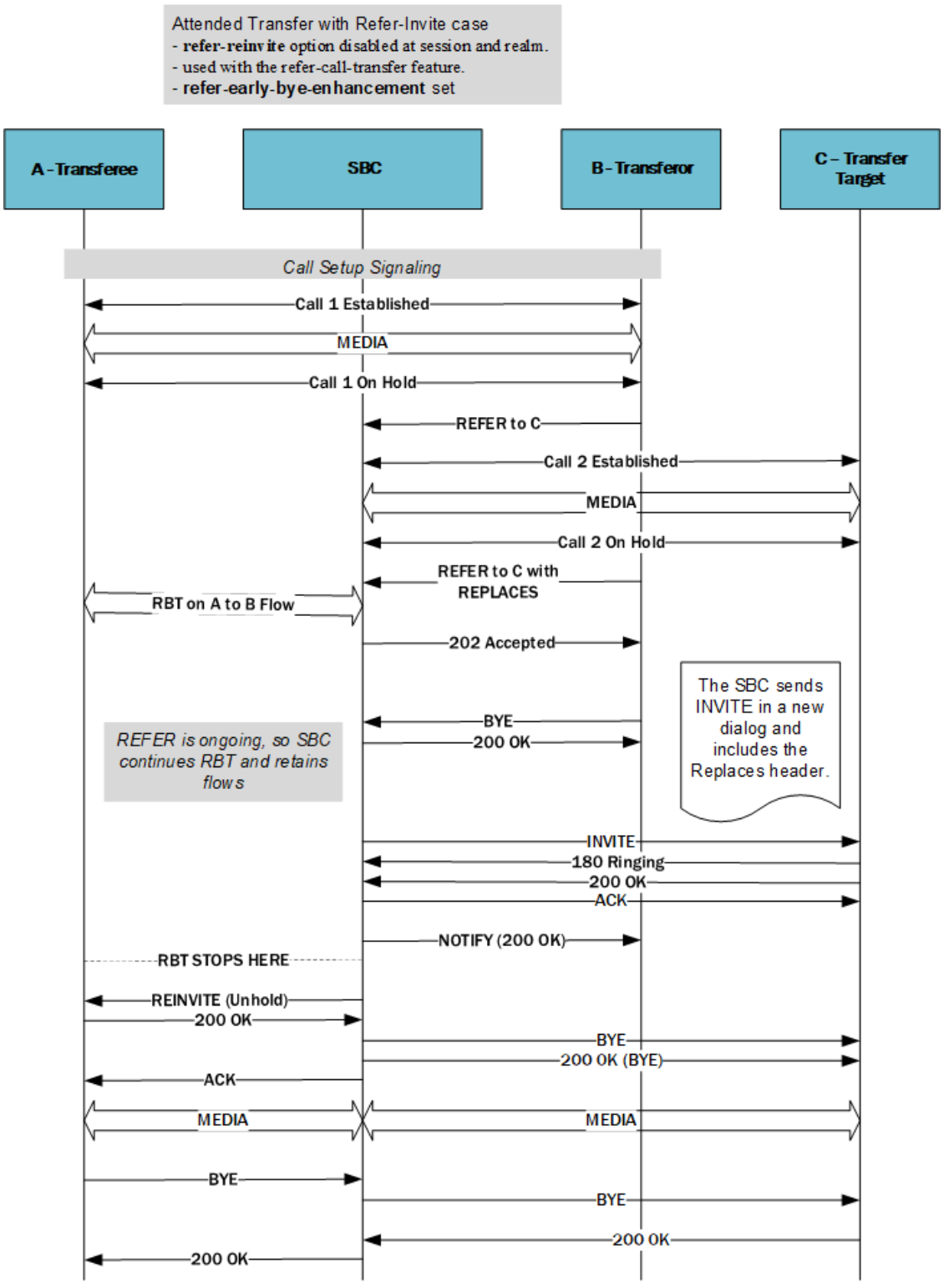
When you set the **refer-early-bye-enhancement** option, the SBC does not tear down the flow during this blind/attended (Invite) transfer because it detects a pending REFER. Instead, the SBC maintains the flow until the transfer is complete or it receives a 200OK from B.



This scenario depicts the SBC supporting an attended Transfer with Refer-Invite case. You have left the **refer-reinvite** option disabled at the session and realm. Here, the SBC sends an INVITE in a new dialog and includes the Replaces header. This is used in conjunction with the refer-call-transfer feature



Again, because you have set the **refer-early-bye-enhancement** option, the SBC does not tear down the flow during this attended Transfer with Refer-Invite case because it detects a pending REFER. Instead, the SBC maintains the flow until after it sends a NOTIFY (200 OK) to B confirming the transfer is complete.



Applicable Logging

The SBC provides debug-level log messages for troubleshooting this feature. DEBUG logs are available to monitor proper processing of the feature. NOTICE logs are also available to track unrecoverable errors. The system captures logs in the log.sipd file, assuming you have enabled DEBUG mode.

Examples of applicable log messages include the following, showing the system recognizing a trigger that results in invoking the feature:

```
Feb 9 11:30:22.121 [CONFIG] (0) earlyByeEnhancement: true
Feb 9 11:30:25.349 [SESSION] (2) Sipd_is_refer_early_bye_enhancement is ON
is_pending true
Feb 9 11:30:25.349 [SESSION] (2) is_pending true
```

Other examples of applicable log messages include the following, showing the system processing the feature:

```
Feb 9 11:30:25.349 [SESSION] (2) <sipRefer.cpp:1837>
ReferCall::complete_transfer setting the pdialog->dropFlowFlag to true.
Feb 9 11:30:25.349 [SESSION] (2) <sipTrans.cpp:17268>
SipClientTrans::drop_media_for_bye dropFlowFlag is true.
Feb 9 11:30:30.365 [TRANS] (2) <sipRefer.cpp:1887>
ReferCall::complete_transfer dropFlowFlag is true, dropping the pending flows.
```

REFER Source Routing

If, after the conclusion of static or dynamic REFER handling, the REFER is terminated and a new INVITE issued, users now can specify a policy lookup behavior based upon either the source realm of the calling party (the INVITE originator), or the source realm of the referring party (the REFER originator).

Behavior is controlled by a new **refer-src-routing** parameter in the **sip-config** configuration element.

disabled, the default value, specifies that the Oracle Communications Session Border Controller performs a policy lookup based on the source realm of the calling party.

enabled specifies that the Oracle Communications Session Border Controller performs a policy lookup based on the source realm of the referring party. Note: Enable refer-src-routing when refer-call-transfer is set to dynamic.

REFER Source Routing Configuration

To enable realm-based REFER method call transfer:

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

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

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

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

4. **refer-call-transfer** — Retain the default (**disabled**) to proxy all REFERs to the next hop. Use **enabled** to terminate all REFERs and issue a new INVITE. Use **dynamic** to specify REFER handling on a call-by-call basis, as determined by the value of the **dyn-refer-term** parameter.
5. **dyn-refer-term** (meaningful only when **refer-call-transfer** is set to dynamic) — Retain the default (**disabled**) to terminate the REFER and issue a new INVITE. Use **enabled** to proxy the REFER to the next hop.
6. Save and activate your configuration.

   To enable session-agent-based REFER method call transfer:
7. In Superuser mode, type **configure terminal** and press Enter.

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

8. Type **session-router** and press Enter.

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

9. Type **session-agent** and press Enter.

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

10. **refer-call-transfer** — Retain the default (**disabled**) to proxy all REFERs to the next hop. Use **enabled** to terminate all REFERs and issue a new INVITE. Use **dynamic** to specify REFER handling on a call-by-call basis, as determined by the value of the **dyn-refer-term** parameter.
11. Save and activate your configuration.

    To specify policy lookup for a newly generated INVITE:
12. In Superuser mode, type **configure terminal** and press Enter.

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

13. Type **session-router** and press Enter.

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

14. Type **sip-config** and press Enter.

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

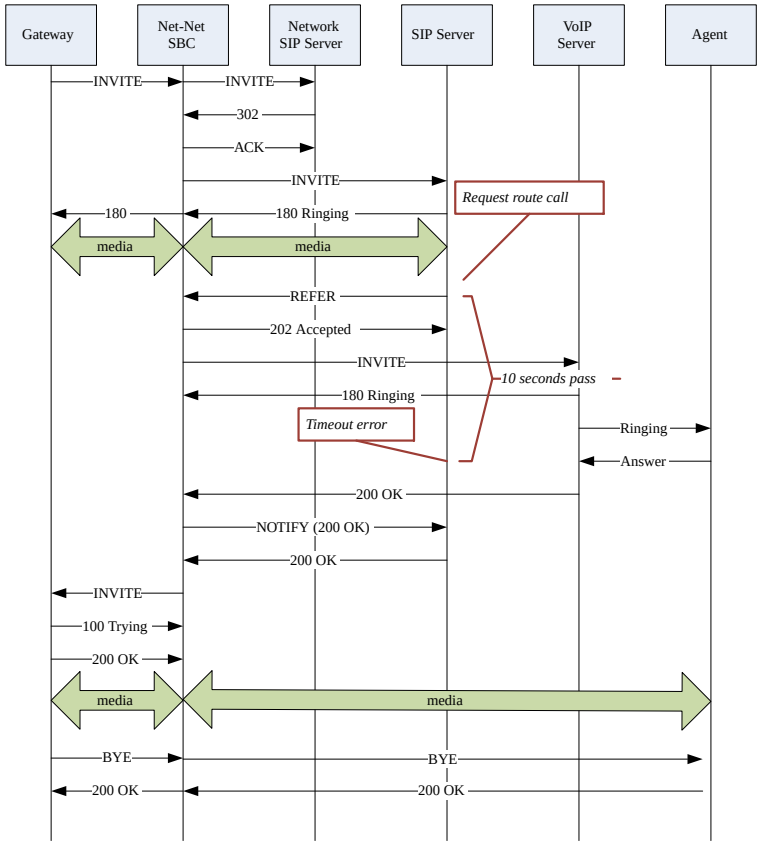
15. **refer-src-routing** — Retain the default (**disabled**) to perform a policy lookup based upon the source realm of the calling party (the issuer of the original INVITE). Use enabled to perform a policy lookup based upon the source realm of the referring party (the issuer of the REFER).
16. Save and activate your configuration.

180 & 100 NOTIFY in REFER Call Transfers

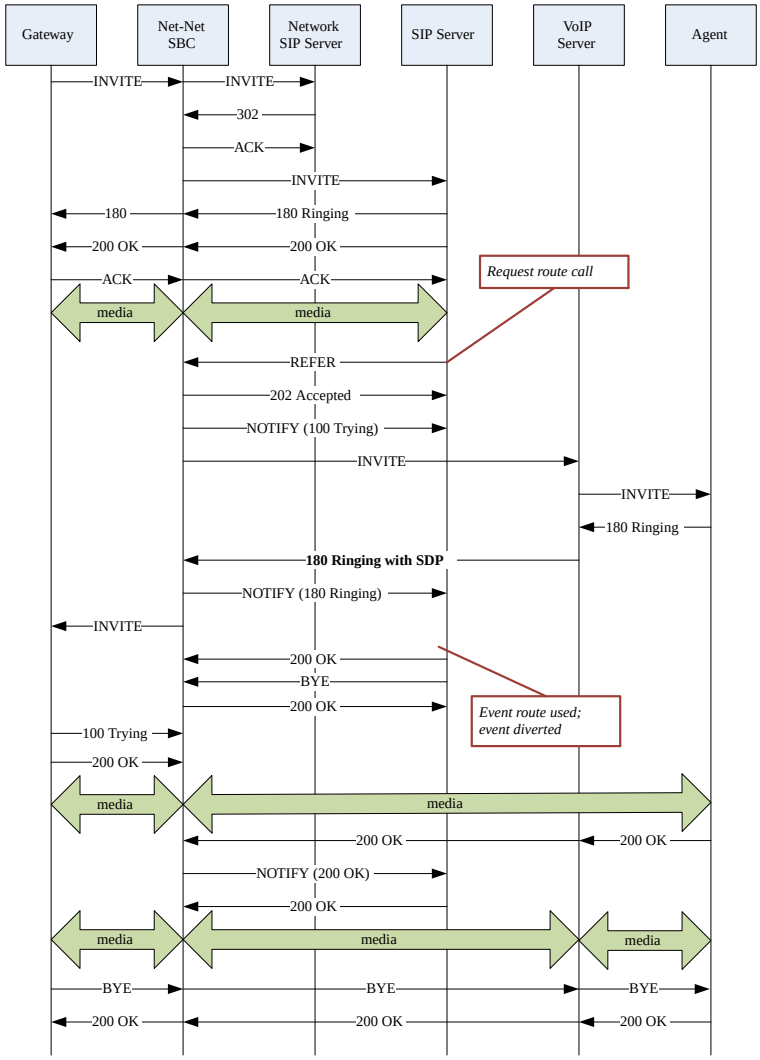
When you configure your Oracle Communications Session Border Controller to support REFER call transfers, you can enable it to send a NOTIFY message after it has sent either a 202 Accepted or sent a 180 Ringing message. If your network contains elements that comply with RFC 5589, and so expect the NOTIFY message after the 202 Accepted and each provisional 180 Ringing, you want to set the **refer-notify-provisional** to either **initial** or **all**, according to your needs.

Without this parameter changed from its default (**none**), the Oracle Communications Session Border Controller does not return send the NOTIFY until it receives the 200 OK response from the agent being called. If the time between the REFER and the NOTIFY exceeds time limits, this sequencing can cause the Oracle Communications Session Border Controller's NOTIFY to go undetected by devices compliant with RFC 5589. Failures during the routing process can result.

You can see how a sample call flow works without setting the **refer-notify-provisional** parameter.



When you compare the call flow above to the one depicting the scenario when the Oracle Communications Session Border Controller has the **refer-notify-provisional** changed from its default, you can see that the Oracle Communications Session Border Controller now response with a NOTIFY in response to the 202 Accepted and it sends another after the 180 Ringing.This causes the event to be diverted successfully.



Sample Messages

In compliance with RFC 5589, the NOTIFY message with 100 Trying as the message body looks like the sample below. Note that the expires value in the subscription state header is

populated with a value that equals 2* TIMER C, where the default value of TIMER C is 180000 milliseconds.

```
NOTIFY sips:4889445d8kjtk3@atlanta.example.com;gr=723jd2d SIP/2.0
Via: SIP/2.0/TLS 192.0.2.4;branch=z9hG4bKnas432
Max-Forwards: 70
To: <sips:transferor@atlanta.example.com>;tag=1928301774
From: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>;tag=a6c85cf
Call-ID: a84b4c76e66710
CSeq: 73 NOTIFY
Contact: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, tdialog
Event: refer
Subscription-State: active;expires=360
Content-Type: message/sipfrag
Content-Length: ...
SIP/2.0 100 Trying
```

Also in compliance with RFC 5589, the NOTIFY message with 180 Ringing as the message body looks like the sample below. Again, the expires value in the subscription state header is populated with a value that equals 2* TIMER C, where the default value of TIMER C is 180000 milliseconds.

```
NOTIFY sips:4889445d8kjtk3@atlanta.example.com;gr=723jd2d SIP/2.0
Via: SIP/2.0/TLS 192.0.2.4;branch=z9hG4bKnas432
Max-Forwards: 70
To: <sips:transferor@atlanta.example.com>;tag=1928301774
From: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>;tag=a6c85cf
Call-ID: a84b4c76e66710
CSeq: 73 NOTIFY
Contact: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, tdialog
Event: refer
Subscription-State: active;expires=360
Content-Type: message/sipfrag
Content-Length: ...
SIP/2.0 180 Ringing
```

Also in compliance with RFC 5589, the NOTIFY message with 200 OK as the message body looks like the sample below.

```
NOTIFY sips:4889445d8kjtk3@atlanta.example.com;gr=723jd2d SIP/2.0
Via: SIP/2.0/TLS 192.0.2.4;branch=z9hG4bKnas432
Max-Forwards: 70
To: <sips:transferor@atlanta.example.com>;tag=1928301774
From: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>;tag=a6c85cf
Call-ID: a84b4c76e66710
CSeq: 74 NOTIFY
Contact: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, tdialog
Event: refer
Subscription-State: terminated;reason=noresource
```

```
Content-Type: message/sipfrag
Content-Length: ...
SIP/2.0 200 OK
```

180 and 100 NOTIFY Configuration

You can apply the **refer-notify-provisional** setting to realms or to session agents. This section shows you how to apply the setting for a realm; the same steps and definitions apply to session agents.

If you do not want to insert NOTIFY messages into the exchanges that support REFER call transfers, you can leave the **refer-notify-provisional** set to **none**. This means that the Oracle Communications Session Border Controller will send only the final result NOTIFY message. Otherwise, you want to choose one of the two settings described in the instructions below.

To enable 100 and 180 NOTIFY messages in REFER call transfers:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

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

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

4. **refer-notify-provisional**—Choose from one of the following settings, where the Oracle Communications Session Border Controller:
   - **initial**—Sends an immediate 100 Trying NOTIFY, and the final result NOTIFY
   - **all**—Sends an immediate 100 Trying NOTIFY, plus a notify for each non-100 provisional messages the Oracle Communications Session Border Controller receives; and the final result NOTIFY

     ```
     ORACLE(realm-config)# refer-notify-provisional all
     ```

5. Save your work.

SIP REFER Re-Invite for Call Leg SDP Renegotiation

Enhancing the original implementation of SIP REFER termination introduced in Release S-C6.0.0, this change to Oracle Communications Session Border Controller behavior allows for SDP renegotiation between both parties of a transferred call.

Scenario

In a call transfer initiated by SIP REFER, a call is established between two user agents, UA-A and UA-B. UA-B then sends a REFER request to transfer the call to UA-C. The challenge is

that UA-A and UA-B had already been communicating using mutually agreed-on codec, while UA-C might not be using an entirely different codec.

To solve this problem, the Oracle Communications Session Border Controller causes a new SIP session and new media session to be created between UA-A and UA-C. The Oracle Communications Session Border Controller removes any resources allocated for use between UA-A and UA-B, and then severs its connection with UA-B. The session between UA-A and UA-C continues.

Alterations to SIP REFER

The Oracle Communications Session Border Controller sends re-INVITE with the negotiated SDP to the user agent for which the Oracle Communications Session Border Controller performs the call transfer.

More about SIP REFER

The Oracle Communications Session Border Controller makes the new call between Party A and Party C appear as though A were participating to allow the Oracle Communications Session Border Controller's natural media setup occur in the same way as if the REFER had actually been sent to A and A had sent a new INVITE.

When the Oracle Communications Session Border Controller receives a REFER request and determines it needs to handle it locally, it creates a new INVITE made to look like one from Party A. And the Oracle Communications Session Border Controller actually processes this INVITE as though it were from Party A. As a result, new SIP and new media sessions are created with new media ports for Parties A and C. When the INVITE to Party C receives a final response, the Oracle Communications Session Border Controller sends the result to Party B using a SIP NOTIFY request.

If the new INVITE succeeds, the old context and flows disappear and the new context and flows for the A-to-C connection remain in place. Because of the new media ports, the Oracle Communications Session Border Controller sends a re-INVITE to Party A, directing media to the new port and forwarded to Party C. Next, the original dialog with Party B needs to be terminated; if the Oracle Communications Session Border Controller has not received Party B's BYE, it will wait five second and then send Party B a BYE.

If the INVITE to Party C fails, the new SIP and media sessions are deleted as are the new context and flows. The Oracle Communications Session Border Controller treats Party A differently depending on whether or not a BYE was received from Party B. If a BYE was received from Party B. then the Oracle Communications Session Border Controller sends a BYE to Party A. If not, the original SIP and media sessions as well as the context and media flows remain in tact. This way, Party B can re-establish the call with Party A using a re-INVITE. In the case, the Oracle Communications Session Border Controller waits 32 second before sending a BYE.

If the Oracle Communications Session Border Controller receives a BYE while processing the INVITE to Party C, it sends a CANCEL message to Party C in an attempt to cancel the call. The BYE passes to Party B, and associated sessions, contexts, and flows terminate normally. Still, the Oracle Communications Session Border Controller waits for the final response to the INVITE to Party C. If the Oracle Communications Session Border Controller receives a successful response, it sends an ACK and then a BYE to terminate the abandoned call. If the Oracle Communications Session Border Controller receives an unsuccessful final response, it uses its normal response error handling processes. In either of these last two cases, all sessions, context, and flow are deleted.

Please note that the Oracle Communications Session Border Controller does not remove the a=sendonly attribute from the SDP it sends to Party A during the A-to-B call, and extra media ports are not allocated for the original media session.

SIP REFER with Replaces

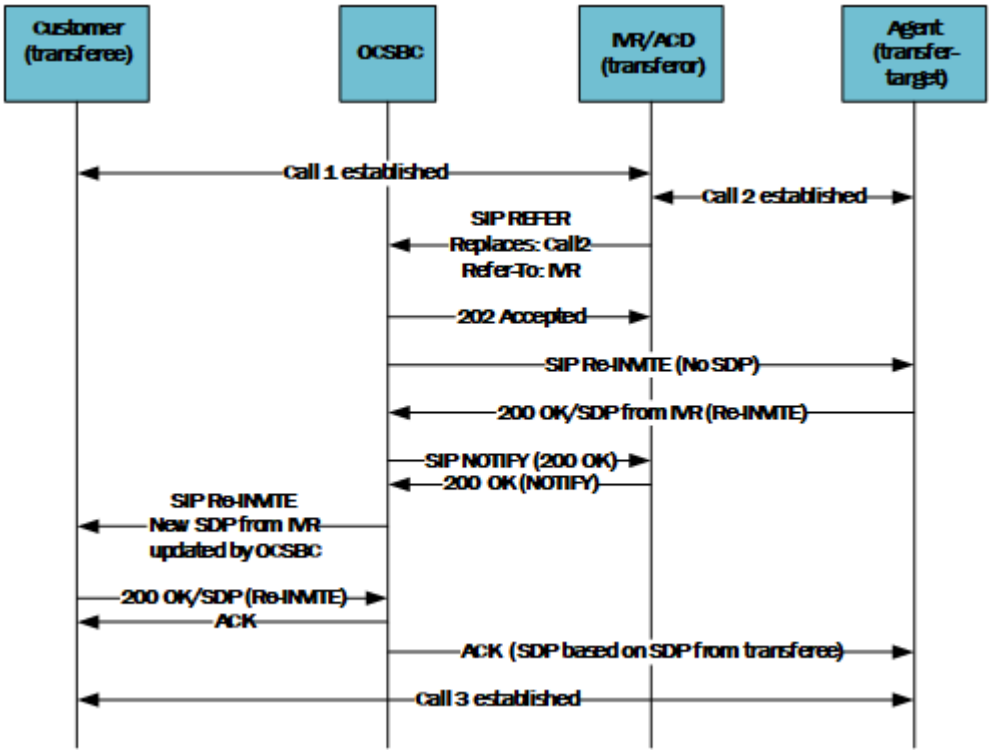
To support enterprise and call center applications, the Oracle Communications Session Border Controller (SBC) provides the ability for one party participating in a three-way call to request direct connectivity between the other two parties and to leave the call silently when that connectivity is established. SIP supports this function using the Replaces header in a REFER message, also known as REFER with Replaces.

A typical example of a refer with replaces flow includes three parties:

- Transferee—A customer who calls into a company's service department.
- Transferor—The service department's IVR/ACD, which manages the transfer.
- Transfer-Target—The agent who ultimately talks with the customer.

To support the SIP REFER, the SBC first establishes calls from the transferee to the transferor (Call 1) and from the transferor to the transfer-target (Call 2). After establishing these calls, the SBC receives a SIP REFER from the transferor and manages the SIP and SDP signaling to replace Call 2 with a new call, Call 3, which is between the transferor and the transfer-target.

As shown below, call replacement consists of SIP ReINVITEs and associated messaging to establish flows for Call 3. This setup is subject to timing, compatibility and other hurdles that could cause the REFER to fail. Assuming successful setup, the SBC subsequently issues BYEs for Call 1 and Call 2 and supports any further signaling for Call 3.



The common high-level sequence of a REFER with REPLACES application includes:

1. The transferee calls a customer service line and reaches, via the SBC, an Interactive Voice Response system/Automatic Call Distribution (IVR/ACD) system. In some architectures, these are two separate elements, but can always be understood as the transferor.

2. Based on the customer's selection from the menu of options, the IVR/ACD contacts an agent, the transfer-target, via the SBC.
3. Since the ultimate goal is for the IVR/ACD to drop out of the path, it sends a REFER with Replaces to the SBC. This message indicates the SBC should replace the IVR/ACD endpoint in the call leg with the transfer-target's endpoint.
4. The SBC processes the REFER with Replaces, issuing ReINVITEs to the transferee with the agent's parameters.
5. There are multiple significant behaviors that the SBC performs after the Re-INVITE, including:
   - Retains the original sessions and dialogs intact, supporting any subsequent REFER attempt by the transferor.
   - Manages SDP such that, in the case of attended transfer, it:
     – Sends a Re-INVITE without SDP to the transfer-target.
     – Sends Re-INVITE with updated SDP to transferee, based on 200OK SDP received from the transfer-target.
     – Sends ACK with new SDP to transfer-target, based on 200OK SDP received from the transferee. (This behavior can be changed for backward compatibility.)
   - Waits for a final success response to the Re-INVITE retaining both the transferee-to-transferor and transferor-to-transfer target call legs until it receives indications of success or failure. If there is a failure response, the SBC sends the failure response to the transferor using a SIP NOTIFY, relying on it to either issue another REFER or release the calls.

     > **Note:**
     >
     > the SBC only handles 4xx, 5xx and 6xx error responses for the REFER ReINVITE towards the Transfer Target.

     An example of such a failure is the receipt of a Re-INVITE from the transfer-target at the same time the SBC sends a Re-INVITE to the transfer-target. To manage this sequence, the SBC accepts the ensuing 491 Request Pending message from the transfer-target and:
     – Sends a SIP NOTIFY to the transferor indicating the transfer failed due to a 491 Request Pending,
     – Retains both calls, then
     – Waits for the transferor to issue a new SIP REFER in an attempt to complete the transfer.
   - If it receives a BYE from the transferee after it has sent a Re-INVITE (with SDP) to the transfer-target, the SBC sends 2 BYEs to the transferor, one for the transferee to transferor leg and one for the transferor to transfer-target leg, and also sends 1 BYE to the transfer-target for the transferor to transfer-target leg.
6. The IVR/ACD drops out of the media path once the bridged call between the transferee and the transfer-target is established.

Direct media connectivity between endpoints must be possible in order for the REFER with Replaces to be carried out properly.

For example, if both endpoints (such as the customer and agent from the example above) are behind the same firewall, direct media connectivity should be possible. However, if one endpoint is behind a firewall and the other is not, then direct media connectivity may not be possible.

Note also that the SBC complies with RFC 3264 by retaining identical o= line in the SDP when issuing an offer that modifies the session with the exception that it increments the sess-version value by one. This helps maintain SDP consistency across the dialog.

For licensing capacity purposes, note that a bridged session counts as a single call. Note also that the SBC supports attended call transfers during HA switchover.

Sending Offerless Responses to Re-Invites

By default, the SBC sends SDP in responses to Re-INVITEs to the transfer-target. This supports changes to the SDP during attended transfers. If desired, you can configure the SBC to send offer-less responses to the Re-INVITE to the transfer target. You configure this by enabling the **refer-reinvite-no-sdp** parameter on the **sip-config**. This is a global configuration

SIP REFER with Replaces Configuration

You enable SIP REFER with Replaces handling either in the realm configuration or in the session agent configuration. This section provides the steps to apply this option to a session agent. The syntax for applying this option to a realm is the same.

To enable sending ReINVITES to a referred agent on an existing session/dialog:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

3. Type **session-agent** and press Enter.

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

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name **refer-reinvite**. Then press Enter.

   ```
   ORACLE(session-agent)# options +refer-reinvite
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

SDP Handling for Compatibility Configuration

You enable **refer-reinvite-no-sdp** parameter in the **sip-config** to prevent the system from including SDP in the response to the Re-INVITE it receives from the transfer-target.

To enable this parameter:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

   You must select the **sip-config** using the ACLI **select** command to edit it.

4. **refer-reinvite-no-sdp**—Enable this parameter to cause the system to globally exclude SDP from re-INVITE responses sent to the transfer target.

   ```
   ORACLE(sip-config)#refer-reinvite-no-sdp enabled
   ```

SIP REFER-to-BYE

The Oracle Communications Session Border Controller's SIP REFER-to-BYE capability addresses situations when other network elements do not support the REFER method but do offer blind transfer in a SIP BYE request. The target number is encoded in a Reason header of the BYE request. In such cases, the Oracle Communications Session Border Controller terminates the REFER and passes the Refer-To number in a Reason header of the BYE.

You configure both SIP interfaces and SIP session agents with the refer-to-bye option to use this function:

- SIP interface—You add this ability to SIP interfaces facing the SIP elements that need to receive a BYE instead of a REFER. This setting only applies when the next hop is not a session agent.
- SIP session agent—The SIP session agent takes precedence over the SIP interface. You add this ability to SIP session agents that need to receive a BYE instead of a REFER. If the next hop SIP element—the remote target in the dialog—is a session agent, in other words, you need to configure the option for it. Note that when you use this option for SIP session agents, the SIP interface or realm on which the REFER is received takes precedence over the REFER-to-BYE capability.

When a REFER request arrives and the REFER-to-BYE capability applies, the Oracle Communications Session Border Controller responds to it with a 202 Accepted and sends a NOTIFY to terminate the implicit refer subscription. This NOTIFY contains a message/sipfrag body with SIP/2.0 200 OK. Upon receiving the response to this NOTIFY, the Oracle

Communications Session Border Controller sends a BYE with an added Reason header (encoded with the Refer-To number) to the other end.

The network element that does not accept REFERs takes the BYE with the Reason header and issues a new initial INVITE that initiates transfer, which the Oracle Communications Session Border Controller sees as starting a new and independent session.

SIP hold-refer-reinvite

When SIP hold-refer-reinvite is enabled for REFER with Replaces, the system queues the outgoing Invite populated from the received REFER based on the dialog state.

In a deployment where a call goes through the Oracle Communications Session Border Controller (SBC) before going to an Interactive Voice Response (IVR) server, the SBC proxies the intermediate reinvite that the IVR sends to the transfer target. If the intermediate reinvite is in either the pending state or the established state when the IVR initiates the transfer to the transfer target, the SBC terminates the call prematurely. The hold-refer-reinvite option allows the SBC to queue the Out Going INVITE from the received REFER request when the previously proxied reinvite request is in either the pending state or the established state. The result is a successful call.

Enable the SIP hold-refer-reinvite option from the ACLI command line or the Web GUI in Advanced mode.

Enable hold-refer-reinvite - ACLI

The SIP hold-refer-reinvite parameter for REFER with Replaces is a parameter that you enable to prevent premature call termination in a deployment where calls are proxied by the Oracle Communications Session Border Controller.

- Confirm that refer-reinvite is added to realm/SA/SipInterface options.
- Confirm that refer-call-transfer is enabled on realm/SA/SipInterface
- Confirm that the session agent on which you want to enable hold-refer-reinvite is configured.

To enable hold-refer-reinvite, select a configured session agent and enable the parameter on the selected agent.

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. Type **select**, and press ENTER.

   The system displays a numbered list of session-agents.

3. Type the number of the agent on which you want to enable hold-refer-invite, and press ENTER.
4. Type **hold-refer-reinvite enabled**, and press ENTER.
5. Type **done** to save the configuration.

- Enable the refer-hold-reinvite parameter in the realm configuration.
- Enable the refer-hold-reinvite parameter in the session agent configuration.

SIP Roaming

This section explains how to configure SIP roaming. SIP roaming lets subscribers move from one active SIP device to another (at the same site or multiple sites) and retain service at the last registering device.

Overview

The Oracle Communications Session Border Controller supports multiple active registrations for the same user. The softswitch makes decisions regarding the current location of the user and the handling of requests from devices that are not currently identified as the user location. When there are multiple NATs, the Oracle Communications Session Border Controller is still required to let the softswitch be able to differentiate it.

The Oracle Communications Session Border Controller's SIP roaming ability supports the following features:

- Multiple active registrations from the same user can be cached, allowing subscribers to move from one active SIP device to another (at the same site or multiple sites) and still retain service at the last registering device. With the SIP roaming feature, one person, using multiple devices, can be contacted at all of the devices. These multiple devices (with their unique contact information) register to indicate that they are available for anyone that wants to contact that one person.
- The Oracle Communications Session Border Controller can also inform network devices (such as softswitches) of private SIP device IPv4 addresses (endpoints) and the public firewall address of the user location.

Process Overview

Caller 1 wants to contact Person A. Caller 1 sends a message to persona@acmepacket.com, but Person A has configured more than one SIP-enabled device to accept messages sent to that address. These devices have unique addresses of desk@10.0.0.4 and phone2@10.0.0.5. Person A has desk@10.0.0.4 and phone2@10.0.0.5 registered with the Oracle Communications Session Border Controller for anything addressed to persona@acmepacket.com.

With the SIP roaming feature, the Oracle Communications Session Border Controller accepts and stores both registrations for persona@acmepacket.com. That way, when someone wants to get in touch with Person A, the messages are sent to both devices (desk@10.0.0.4 and phone2@10.0.0.5) until Person A answers one of them. You do not need to configure your Oracle Communications Session Border Controller for this functionality; your Oracle Communications Session Border Controller automatically provides it.

Using Private IPv4 Addresses

In addition to supporting multiple registries, the Oracle Communications Session Border Controller (SBC) can also distinguish user locations by their private IPv4 address and the IPv4 address of the public firewall. Using this information, the SBC adds private endpoint and public firewall information to Contact headers.

For example, entering this information causes a Contact header that formerly appeared as the following:

```
Contact:<sip:027411620263.67.143.217>
```

to subsequently appear as the following:

```
Contact:<sip:027411620263.67.143.217;ep=192.168.1.10;fw=10.1.10.21>
```

The SBC SIP proxy reads this information and populates the contact-endpoint and contact-firewall fields with the appropriate values.

Example 1 With a NAT Firewall

The Oracle Communications Session Border Controller (SBC) SIP proxy is configured with the following changeable parameters:

- endpoint= IP address of the SIP UA
- useradd= IP address of the Firewall Public IP address or the source layer 3 IP address of Register message
- userport= IP address port number of the Firewall Public IP address or the source layer 3 IP address port of Register message
- SBC address=63.67.143.217
- firewall public address=10.1.10.21
- firewall public address port=10000
- SIP endpoint behind firewall=192.168.1.10

SIP message Contact header:

```
Contact:<sip:027411620263.67.143.217; endpoint=192.168.1.10;
useradd=10.1.10.21; userport=10000; transport=udp>
```

Example 2 Without a NAT Firewall

The Oracle Communications Session Border Controller SIP proxy is configured with the following changeable parameters:

- useradd= IP address of the SIP UA or the source layer 3 IP address of Register message
- userport= IP address port number of the SIP UA or the source layer 3 IP address port of Register message
- Oracle Communications Session Border Controller address=63.67.143.217
- SIP endpoint=192.168.1.10
- SIP endpoint IP address port=5060

SIP message Contact header:

```
Contact:<sip:027411620263.67.143.217; useradd=192.168.1.10; userport=5060;
transport=udp>
```

For SIP, the softswitch responsibility is that the URI SD put in the Contact of the REGISTER message should be reflected in the 200-OK response to the REGISTER request. The Contact header of the response should have an expires header parameter indicating the lifetime of the registration.

The following example shows a Oracle Communications Session Border Controller Send:

```
Contact: <sep: 027411620263.67.143.217 endpoint=192.168.1.10;
useradd=10.1.10.21; userport=10000>;
```

The following examples shows the softswitch Respond:

```
Contact: <sep: 027411620263.67.143.217 endpoint=192.168.1.10;
useradd=10.1.10.21; userport=10000>;  expires=360
```

The contact field for endpoint and firewall parameters only appear in the following:

- Contact header of a REGISTER request sent from the Oracle Communications Session Border Controller to the softswitch server
- Contact header of a REGISTER response sent from the softwitch server to the Oracle Communications Session Border Controller
- Request-URI of an initial INVITE sent from the UT CSA server to the Oracle Communications Session Border Controller

An active endpoint is deleted when it does not register within the registration-interval setting or receives a 401 Unauthorized.

SIP Roaming Configuration

You can configure the SIP configuration's options parameter to indicate that you want to use the private IP address of the SIP device that the user is using and/or the public firewall address that identifies the location of the device. If defined, these options will be added as parameters to all Contact headers.

You can identify the endpoint and/or firewall information using the following options:

- contact-endpoint=<value> where <value> is the endpoint address or label
- contact-firewall=<value> where <value> is the firewall address or label

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. Type **options** followed by a Space.
5. After the Space, type the information for an endpoint or a firewall, or both:

```
contact-endpoint=”<label>
contact-firewall=”<label>
"contact-endpoint="<label>",contact-firewall="<label>""
```

6. Press Enter.

   For example, if you want your Oracle Communications Session Border Controller to add private endpoint and public firewall information to Contact headers, and you want to label this information as ep and fw, you would enter the following information in the ACLI.

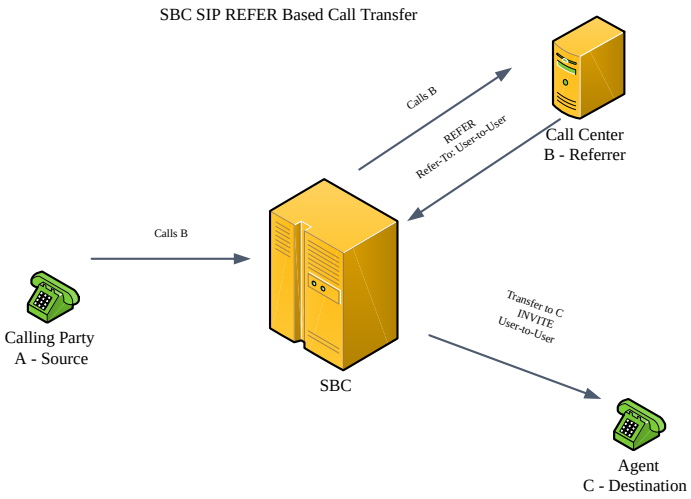
```
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)# options "contact-endpoint="ep",contact-firewall="fw""
```

SIP REFER Call Transfer UUI Relay

The SIP REFER Call Transfer **User to User Information (UUI)** Relay option assists in the transfer of caller details through using the information in the "Refer-To" header in a new "User to User" header in the INVITE to the Referred-to party. This feature only works when the **refer-call-transfer** option is enabled on the realm or session agent where the REFER is received. This behavior change is enabled by default. This option can be used by a Call Center application to transfer a call with user information to an agent.

A new INVITE is sent with the information about the calling user to the agent by way of the "Refer-To" header using the **User to User Information.** The product variable needs to relay this UUI as a separate header in the INVITE message while transferring the call to the destination. In this scenario, the product terminates the REFER message having captured the UUI and then sends an INVITE with the UUI to the agent. This feature works only when the **refer-call-transfer** option is enabled on the Realm or Session Agent where REFER is received. By default this option is enabled and thus needs to be specifically disabled in the configuration if not wanted.

The following illustration shows the path of the call from the source to the destination by way of the product utilizing the **User to User Information** relay



SIP REFER UUI Relay

Refer-To Header

The Oracle Communications Session Border Controller supports the format of **User-to-User** in the "Refer-To" header in the REFER message. The following example shows the new text after the IP address and port in the second to last line.

```
REFER sip:7325550000@192.168.28.10:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.29.1:5060;branch=z9hG4bK-10659-1-4
From: 7325550000 <sip:7325550000@192.168.29.1:5060>;tag=1
To: 7321110000 <sip:732111000@192.168.28.19:5060>;tag=1
Call-ID: 1-10679@192.168.28.1
...
Refer-To: <sip:9785550000@192.168.29.1:6062?User-to-
User=56a390f3d2b7310023a%3Bencoding%3Dhex%3Bpurpose%3Disdn-
interwork%3Bcontent%3Disdn-uui
Content-Length:0
```

User to User Parameter in INVITE

If the UUI is received in the "Refer-To" header in the REFER message, the Oracle Communications Session Border Controller adds the new UUI header in the INVITE message to the destination party. The system encodes the INVITE message as shown in the following example. All the escape characters received in the UUI parameter converts to plain text and all

the UUI header parameters are relayed. In the example below, note the new UUI content below **Content-Length** and in the **Supported** fields.

```
INVITE sip:9785550000@192.168.29.1:6062 SIP/2.0
Via: SIP/2.0/UDP 192.168.29.20:5060;branch=z9hG4bKgkm8110090jmlvcfk80.1
From: 7321110000 <sip:7321110000@192.168.28.1:5060>;tag=1
To: <sip:9785550000@192.168.29.1:6062>
Call-ID: 537e3302f02bbb0ddd8b8d7538f8b33030@192.168.28.1
CSeq: 2 INVITE
Contact:<sip:sipp@192.168.29.20:5060;transport=UDP>
Session-Expires: 3600;refresher=uac
Max-Forwards: 69
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 135
User-to-User:56a390f3d2b7310023a;encoding=hex;purpose=isdn-
interwork;content=isdn-uui
Supported: timer,uui
Referred-By:<sip.7325550000@192.168.29.1:5060>
Route:<sip:9785550000@core2:6020,lr>

v=0
o=user153655765 2353687637 IN IP$ 192.168.28.1
s=-
c=IN IP4 192.168.28.1
t= 0 0
m=audio 8000 RTP/AVP 8
a=rtpmap 8 PCMA/8000
```

User to User Header in REFER

The product supports the REFER header that replaces the default header with the UUI header in an attended call transfer.

```
Aug 4 10:09:10.233 On [256:0]192.168.12.1:5060 received from
192.168.12.2:5060
REFER sip:1000@192.168.12.1:5060 SIP/2.0
Via: SIP/2.0/udp 192.168.12.2:5060;branch=z9hG4bK-bob2alice-9
Max-Forwards: 10
From: bob <sip:2000@acme.com>;tag=bob-0
To: alice <sip:1000@acme.com>;tag=alice-1
Call-ID: 1-192.168.11.2
CSeq: 3 REFER
Refer-To: <sip:3000@acme.com?Replaces=1-192.168.12.2%3Bto-
tag%3Dcarol2bob-0%3Bfrom-tag%3Dbob2carol-2&
User-to-User=56a390f3d2b7310023a%3Bencoding%3Dhex%3Bpurpose%3Disdn-
interwork%3Bcontent%3Disdn-uui>
```

INVITE Message Details

The Oracle Communications Session Border Controller adds the UUI parameter to the **Supported** header in the INVITE message to the **Referred-to** party. The Oracle Communications Session Border Controller encodes only one instance of the UUI header in the INVITE message to the **Referred-to** party. The Oracle Communications Session Border Controller will only process the first UUI parameter received in the "Refer-To" header. The

Oracle Communications Session Border Controller supports a maximum of 128 octets hex content in the UUI header field, excluding parameters. The overall length of the header complies with RFC 3261. If the header length exceeds the maximum, the Oracle Communications Session Border Controller discards the UUI header and cannot relay the call.

SIP REFER Call Transfer UUI Relay Configuration

The SIP REFER Call Transfer **User to User Information** Relay is enabled by default. To disable UUI relaying on a configured realm:

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

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

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **options**—Set the options parameter by typing options, a space, **disable-refer-to-uui=yes**. You may prepend the option name with a plus to add and not replace this option to the existing **realm-config** option list.

```
ORACLE#options +disable-refer-to-uui=yes
```

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

Embedded Header Support

This section explains how to configure embedded header support. The Oracle Communications Session Border Controller supports methods of extracting an embedded P-Asserted-Identity header from a contact header to support E911 when integrated with certain vendor's systems. See RFC 3455 *Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)* for more information.

The embedded header support feature watches for a specified embedded header contained in a Contact header received in a 3XX message. When the specified embedded header is found, the full <header=value> pair is inserted as a unique header in a redirected INVITE message that exits the Oracle Communications Session Border Controller. If the outgoing INVITE message were to contain the specified header, regardless of the use of this feature, the value extracted from the 3XX message replaces the INVITE message's specified header value.

If an incoming Contact header in a 3XX message looks like:

```
Contact: <ESRN@IPv4_Intrado_GW;user=phone?P-Asserted-Identity=%3Csip:+1-
ESQK@IPv4_My_EAG;user=phone%3E>
```

Then, if you configure your Oracle Communications Session Border Controller to parse for the embedded P-Asserted-Identity header to write as a unique header in the outgoing invite message, the outgoing INVITE and P-Asserted-Identity headers will look like:

```
INVITE SIP: ESRN@IPv4_Intrado_GW;user=phone
P-Asserted-Identity: +1-ESQK@IPv4_My_EAG;user=phone
```

Embedded Header Support Configuration

Embedded header support is enabled in the session agent configuration.

To configure embedded header support:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. Select the session agent where you want this feature.

   ```
   ORACLE(session-agent)# select
   <hostname>:
   1: asd          realm=      ip=1.0.0.0
   2: SIPSA        realm=      ip=10.10.102.1
   selection:2
   ORACLE(session-agent)#
   ```

5. **request-uri-headers—Enter a list of embedded headers extracted from the Contact header that will be inserted in the re INVITE message.** To configure this parameter for multiple headers, enclose the headers in double quotes and separate them with spaces. This completes the configuration of embedded header support.

   ```
   ORACLE(session-agent)# request-uri-headers P-Asserted-Identity
   ```

Dialog Transparency

This section explains how to configure dialog transparency, which prevents the Oracle Communications Session Border Controller from generating a unique Call-ID and modifying dialog tags.

Overview

With dialog transparency enabled, the Oracle Communications Session Border Controller is prevented from generating a unique Call-ID and from modifying the dialog tags; the Oracle

Communications Session Border Controller passes what it receives. Therefore, when a call made on one Oracle Communications Session Border Controller is transferred to another UA and crosses a second Oracle Communications Session Border Controller, the second Oracle Communications Session Border Controller does not note the context of the original dialog, and the original call identifiers are preserved end to end. The signalling presented to each endpoint remains in the appropriate context regardless of how many times a call crosses through a Oracle Communications Session Border Controller or how many Oracle Communications Session Border Controllers a call crosses.

Without dialog transparency enabled, the Oracle Communications Session Border Controller's SIP B2BUA rewrites the Call-ID header and inserted dialog cookies into the From and To tags of all messages it processes. These dialog cookies are in the following format: SDxxxxxNN-. Using these cookies, the Oracle Communications Session Border Controller can recognize the direction of a dialog. However, this behavior makes call transfers problematic because one Oracle Communications Session Border Controllers' Call-ID might not be properly decoded by another Oracle Communications Session Border Controller. The result is asymmetric header manipulation and failed call transfers.

Dialog Transparency Configuration

You set one parameter in your SIP configuration to enable dialog transparency.

- For new configurations, this feature defaults to enabled
  To enable SIP dialog transparency:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. Use the ACLI **select** command so that you can work with the SIP configuration.

   ```
   ORACLE(sip-config)# select
   ```

5. **dialog-transparency**—Enter the state of SIP dialog transparency you require for your Oracle Communications Session Border Controller. The default value is **enabled**. The valid values are:
   - enabled | disabled

Route Header Removal

This section explains how to enable the Oracle Communications Session Border Controller to disregard and strip all SIP Route headers. You set an option in a SIP interface configuration to strip all Route headers for SIP requests coming from this interface.

When the Oracle Communications Session Border Controller with this option configured receives an INVITE from an interface, it removes the route headers. However, although it

removes the headers, the Oracle Communications Session Border Controller maintains backward compatibility with RFC 2543 nodes. To do so, it normalizes the request to an RFC 3261 loose routing form before it removes the headers.

Route Header Removal Configuration

The following information explains how to remove SIP route headers.

To configure SIP route header removal:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

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

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

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

4. Type **options strip-route-headers** and press Enter. This completes the configuration of SIP route header removal.

   ```
   ORACLE(sip-interface)# options strip-route-headers
   ```

SIP Via Transparency

This section explains the inbound Via header transparency feature, which enables the Oracle Communications Session Border Controller to insert its Via header on top of the top-most Via header received from user equipment (UE). It then forwards it on to the IP Multimedia Subsystem (IMS) core with the original Via header now located as the bottom-most Via header.

The Oracle Communications Session Border Controller still replaces the Contact and other header addresses with its own, and does not pass on the core's Via headers in outbound requests.

This feature is targeted for the Telecoms & Internet converged Services & Protocols for Advanced Networks (TISPAN) with SIP hosted NAT traversal support. It works with SIP NAT bridged, local-policy routed, and non-SIP NAT configurations, regardless of registration handling.

Some equipment acts as Proxy-CSCF (P-CSCF) and Serving-CSCF (S-CSCF) nodes, with the Oracle Communications Session Border Controller is located between the equipment and user endpoints. The equipment needs to see the each user endpoint's original Via header in order to perform some implicit authentication, admission, and control functions in a TISPAN-compliant model.

You enable Via header transparency on the access SIP interface. Received Via headers are saved for inclusion in requests going out another interface or session agent that does not have the parameter set, in other words, the core side. For any received SIP message where the inbound previous hop interface was enabled for Via header transparency, theOracle

Communications Session Border Controller adds its own Via header as it forwards it, and it also copies the received top-most Via as the new bottom-most Via, if the outbound next hop interface/session agent is not enabled for Via header transparency. The Oracle Communications Session Border Controller also adds a received= parameter to the copied Via header, per the SIP RFC 3261.

Any message received from an interface without Via header transparency enabled, does not have the received Via header copied over to any other direction.

For HNT, where the original top-most (and only) Via header from a UE is a private/false address, the SD should still copy that false address into the core-side, and the received= parameter will contain the real Layer-3 addressing.

SIP Via Transparency Configuration

You can configure SIP Via header transparency for the access SIP interface using the ACLI.

To configure SIP Via header transparency for an access interface:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-level configuration elements.

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

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

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

4. You can either add support to a new SIP interface configuration or to an existing SIP interface configuration:

   For a new SIP interface configuration, you can add the option by typing options, a Space, and then via-header-transparency.

   ```
   ORACLE(sip-interface)# options via-header-transparency
   ```

   For an existing SIP interface configuration without options configured, select the SIP interface, type options followed by a Space, and then via-header-transparency.

   ```
   ORACLE(sip-interface)# select
   ORACLE(sip-interface)# options via-header-transparency
   ```

   For an existing SIP interface configuration with options configured, select the SIP interface, type options followed by a Space, the plus sign (+), and the via-header-transparency option.

   ```
   ORACLE(sip-interface)# select
   ORACLE(sip-interface)# options +via-header-transparency
   ```

5. Save your work using the ACLI **save** or **done** command.

Symmetric Latching

Symmetric latching, or forced HNT, ensures that symmetric RTP/RTCP is used for a SIP endpoint. Symmetric RTP/RTCP means that the IP address and port pair used by an outbound RTP/RTCP flow is reused for the inbound flow. The IP address and port are learned when the initial RTP/RTCP flow is received by the Oracle Communications Session Border Controller. The flow's source address and port are latched onto and used as the destination for the RTP/RTCP sourced by the other side of the call. The IP address and port in the c line and m line respectively in the SDP message are ignored.

If your network is configured with nested realms in order to separate signalling from media, make sure that the symmetric latching feature is enabled on the signaling realm.

> **Note:**
>
> This description is applicable to RTCP only when you also enable the HNT RTCP option in the media-manager configuration. Do not enable symmetric latching on core-facing interfaces.

Symmetric Latching Configuration

To configure symmetric latching:

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

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

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **symmetric-latching** — identifies whether and how to enable symmetric latching on the SBC. This completes the configuration of forced Hosted NAT Traversal (HNT). The default value for this parameter is **disabled**. The valid values are:
   - **disabled** —
   - **enabled** —
   - **pre-emptive** —

```
ORACLE(realm-config)# symmetric-latching pre-emptive
```

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

Enabling RTCP Latching

To enable RTCP symmetric latching:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. Select the media manager configuration so that you can enable HNT RTCP.

   ```
   ORACLE(media-manager-config)# select
   ```

5. **hnt-rtcp —** Enable support of RTCP when the SBC performs HNT. The default value is **disabled**. The valid values are:

   - enabled | disabled

   ```
   ORACLE(media-manager-config)# hnt-rtcp enabled
   ```

6. Save your work using either the ACLI **save** or **done** command.

SIP Pre-emptive Symmetric Media Latching

The Oracle Communications Session Border Controller (SBC) supports symmetric media latching within a realm. However, when two SBCs are in different realms and both realms are configured for symmetric latching, then both will wait for received media packets from the other before transmitting, which results in dropped calls. This feature lets the user configure the SBC to transmit its RTP packets pre-emptively to the peer SDP connection address and then to re-latch the peer RTP source address after receiving the first RTP packet from that peer.

In a call forwarding scenario where a media server (MS) behind Network Address Translation (NAT) is between two SBCs, both SBCs will detect NAT, so both will wait to receive RTP packets before latching to the RTP source address. This feature prevents this by adding the new value **pre-emptive** to the configuration option parameter **symmetric-latching** in the configuration element **realm-config**. When the value is set to **pre-emptive**, the SBC sends the RTP packets to the received SDP connection address without waiting on the latch. Once the RTP packets are received from the peer endpoint, the SBC detects the NAT address mapping; if there is a change in the RTP source address, the SBC re-latches to the new RTP source address. Subsequent RTP packets are then sent to the peer RTP source address. This is also the behavior when there is an UPDATE or reINVITE with the SDP message.

Pre-emptive Symmetric Media Latching Configuration

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

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

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **symmetric-latching** — identifies whether and how to enable symmetric latching in the realm. The default value is **disabled**.
   - **disabled**
   - **enabled**
   - **pre-emptive** — symmetric latching is enabled but the SBC sends RTP packets to the received SDP connection address without waiting on the latch
4. Type **done** to save your configuration.

SIP Number Normalization

This section explains the SIP number normalization feature that applies to the SIP To URI. (Currently the Oracle Communications Session Border Controller supports number normalization on From and To addresses for both inbound and outbound call legs.) Number normalization includes add, delete, and replace string functions that result in consistent number formats.

Number normalization is supported for the following call types:

- SIP to SIP
- H.323 to SIP

Number normalization applies to the SIP To URI. It occurs on ingress traffic, prior to the generation of accounting records or local policy lookups. RADIUS CDR attributes are populated with the normalized numbers. Local policy matching is based on the normalized numbers.

Terminology

The following lists explains the terminology used later.

- X is any digit having the value 0 through 9
- N is any digit having the value 2 through 9
- 0/1 is a digit having the value of either 0 or 1

- NXX is a form of Numbering Plan Area (NPA).
- CC is a 1, 2, or 3 digit country code used in international dialing
- NN is a national number that can be a four to fourteen digit national number used in international dialing, where the combination of CC+NN is a 7 to 15 digit number.
- + symbol in E.164 indicates that an international prefix is required
- E.164 numbers are globally unique, language independent identifiers for resources on Public Telecommunication Networks that can support many different services and protocols.
- N11 number is any of the three-digit dialing codes in the form N11 used to connect users to special services, where N is a digit between 2 and 9

Calls from IP Endpoints

The Oracle Communications Session Border Controller uses the following number normalization rules:

- North American Numbering Plan (NANP) calls: where a number with the format 1NPANXXXXXX is received, the Oracle Communications Session Border Controller adds a plus sign (+) as a prefix to the NANP number. The Oracle Communications Session Border Controller also adds the string ;user=phone after the host IP address in the SIP URI. For example:

  ```
  sip:+1NPANXXXXXX@ipaddr;user=phone
  ```

- International NWZ1 calls: Oracle Communications Session Border Controller receives an international call with the format 011CCNN. The Oracle Communications Session Border Controller deletes the 011 prefix and adds a plus sign (+) as a prefix to CC+NN; and also adds the string ;user=phone after the host IP address in the SIP URI. For example:

  ```
  sip:+CCNN@ipaddr;user=phone
  ```

- Private number calls: when a private number with the format nxxxx (where n=2 through 9) is received, no number normalization is applied by the Oracle Communications Session Border Controller.
- Calls to numbers such as N11, 0-, 0+, 00-, and 01+: the Oracle Communications Session Border Controller adds ;phone-context=+1 after the number and also adds the string ;user=phone after the host IP address in the SIP URI. For example:

  ```
  sip:N11;phone-context=+1@ipaddr;user=phone
  sip:01CCNN;phone-context=+1@ipaddr;user=phone
  ```

- Calls with numbers that are already normalized are not modified by the Oracle Communications Session Border Controller.

Calls from IP Peer Network

For calls received from external peer networks, the Oracle Communications Session Border Controller uses the following number normalization rules:

- Global numbers such as NANP and international E.164 numbers should have already been normalized. If not, the Oracle Communications Session Border Controller applies the same number normalization rules listed in the prior section.

- Calls to numbers such as N11, 0-, 0+, 00-, and 01+: the Oracle Communications Session Border Controller adds ;phone-context=+1 after the number and also adds the string ;user=phone (if absent) after the host IP address in the SIP URI.

SIP Number Normalization Configuration

You can configure SIP number normalization for the realm and session agent using the ACLI.

Realm

To configure SIP number normalization for a realm:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

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

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

4. You can either add SIP number normalization support to a new session agent configuration or to an existing session agent configuration:

   - For a new realm configuration, add the option by typing **options**, a Space, and then **number-normalization**.

     ```
     ORACLE(realm-config)# options number-normalization
     ```

   - For an existing realm configuration without any options already configured, select the realm, type **options** followed by a Space, and then **number-normalization**.

     ```
     ORACLE(realm-config)# select
     ORACLE(realm-config)# options number-normalization
     ```

   - For an existing realm configuration with other options, select the realm, type **options** followed by a Space, the plus sign (**+**), and the **number-normalization** option.

     ```
     ORACLE(realm-config)# select
     ORACLE(realm-config)# options +number-normalization
     ```

5. Save your work using the ACLI **save** or **done** command.

Session Agent

To configure SIP number normalization for a session agent:

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

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the media-level configuration elements.

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

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

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

4. You can either add SIP number normalization support to a new session agent configuration or to an existing session agent configuration:
   - For a new a session agent configuration, add the option by typing **options**, a Space, and then **number-normalization**.

     ```
     ORACLE(session-agent)# options number-normalization
     ```

   - For an existing session agent configuration without any options already configured, select the session agent, type **options** followed by a Space, and then **number-normalization**.

     ```
     ORACLE(session-agent)# select
     ORACLE(session-agent)# options number-normalization
     ```

   - For an existing session agent configuration with other options, select the session agent, type **options** followed by a Space, the plus sign (**+**), and the **number-normalization** option.

     ```
     ORACLE(session-agent)# select
     ORACLE(session-agent)# options +number-normalization
     ```

5. Save your work using the ACLI **save** or **done** command.

SIP Port Mapping

This section contains information about the SIP port mapping feature. SIP port mapping lets you allocate a unique SIP signaling transport address (IP address and UDP port) on the Oracle Communications Session Border Controller in the provider network for each registered endpoint (user agent).

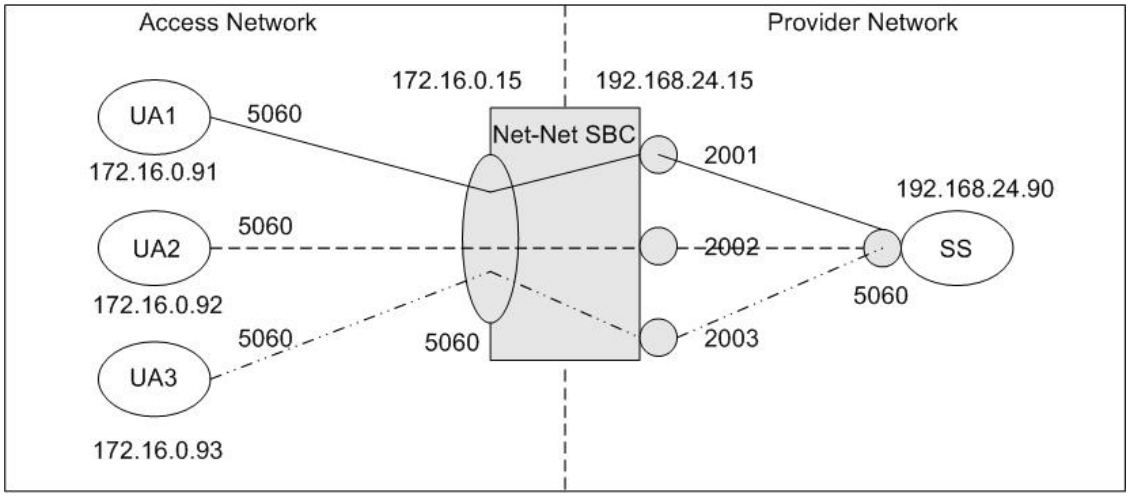
About SIP Port Mapping

You might need to provide a unique signaling transport address for each registered endpoint for admission control, if required by your softswitch vendor. If you have questions about your softswitch, contact the vendor for assistance.

When a Oracle Communications Session Border Controller resides between the endpoints and the softswitch, the softswitch sees the same transport address (that of the Oracle Communications Session Border Controller) for all endpoints. By allocating a unique UDP port

for each endpoint, the Oracle Communications Session Border Controller provides each of them a unique transport address.

The following example illustrates the SIP port mapping feature.



The diagram shows UA1, UA2, and UA3 are endpoints within the access network and that the SIP interface for the access network is 172.16.0.15:5060. On the provider network, the SIP interface is at 192.168.24.15, with the SIP port mapping feature enabled. The softswitch/ registrar is also located on the provider network at 192.168.24.90:5060.

The diagram shows that port 2001 on the provider network is allocated to UA1 on the access network, port 2002 is allocated to UA2, and port 2003 is allocated to UA3. Because of this allocation, all SIP signaling messages sent from the endpoints in the access network to the softswitch on the provider network travel through an allocated signaling port. For example, all signaling messages between UA1 and the softswitch use 192.168.24.15:2001 as the transport address.

How SIP Port Mapping Works

The Oracle Communications Session Border Controller (SBC) allocates SIP port mapping (signaling) ports during a REGISTER request that has registration caching applied. When you define a range of signaling ports for the SIP interface, you create a pool of signaling ports that can be allocated during the REGISTER request.

The SBC allocates a signaling port from the pool when it creates the registration cache entry for a Contact in a REGISTER request. It allocates a separate signaling port for each unique Contact URI from the access side. The registration cache Contact entry contains the mapping between the Contact URI in the access/endpoint realm (the UA-Contact) and the Contact URI in the registrar/softswitch realm (the SD-Contact).

The SD-Contact is the allocated signaling port. The signaling port gets returned to the pool when the Contact is removed from the registration cache. The removal can occur when the cache entry expires; or when the endpoint sends a REGISTER request to explicitly remove the Contact from the registrar. When a signaling port returns to the pool it gets placed at the end of pool list; in a least-recently-used allocation method for signaling ports

When the SBC forwards the REGISTER request to the softswitch, it replaces the UA-Contact with SD-Contact. For example, if UA1 sends a REGISTER request with a Contact URI of sip:ua1@172.16.0.91:5060, it is replaced with sip:192.168.24.15:2001 when the REGISTER request is forwarded to the registrar.

The same translation occurs when UA1 sends that same URI in the Contact header of other SIP messages. SIP requests addressed to the allocated signaling transport address (SD-Contact) are translated and forwarded to the registered endpoint contact address (UA-Contact).

> **Note:**
>
> The maximum number of registered endpoints cannot exceed the number of signaling ports available. If no signaling ports are available for a new registration, the REGISTER request receives a 503 response.

The SBC still processes requests received on the configured SIP port address. Requests sent into the registrar/softswitch realm that are not associated with a registered user will use the configured SIP port address.

Using SIP port mapping with SIPconnect—where unique ports are used for each registered PBX—hinders the SBC from routing incoming calls to the corresponding PBX because the SBC uses DN for the PBX's parent during registration, but the incoming INVITE from the softswitch contains the child DN in its Request URI. Thus the SBC cannot find a matching SBC-Contact because the username of the Request URI contains the child DN, but the username of the SBC-Contact contains the parent DN.

You can enable SIPconnect support in either the realm configuration or session agent for the SIP access network by setting the **sip-connect-pbx-reg** option. With this option set and the destination realm configured for port mapping, the SBC inserts a special search key in the registration table. Rather than adding the SD-Contact as the key as with regular (non-SIPconnect) registrations, the SBC strips user information and instead uses the host and port information as the registration key. The SBC still forwards the registration message with an intact contact username.

SIP Port Mapping Based on IP Address

Some registrars need to know that multiple contacts represent the same endpoint. The extension to this feature answers the expectation from registrars that an endpoint registering multiple AoRs will use a single core-side mapped port to show that the AoRs really represent a single endpoint.

When you enable SIP port mapping based on IP Address, the Oracle Communications Session Border Controller supports core-side UDP port mapping based on the endpoint's IP address. It ignores the username portion of the AoR or Contact.

The Oracle Communications Session Border Controller performs the port mapping allocation and lookup based on all requests using the via-key from the SIP Request. The via-key is a combination of Layer 3 and Layer 5 IP information in the message. The Oracle Communications Session Border Controller performs an additional lookup in the registration table to determine if a via-key already exists. If it does, then the Oracle Communications Session Border Controller uses the port already allocated and does not allocate a new one.

About NAT Table ACL Entries

To enable SIP signaling messages to reach the host processor, the Oracle Communications Session Border Controller adds NAT table ACL entries for each SIP interface. With UDP

without SIP port mapping applied, it adds a single ACL entry for each SIP port in the SIP interface configuration. For example:

```
untrusted entries:
intf:vlan source-ip/mask:port/mask dest-ip/mask:port/mask   prot type    index
0/0:0     0.0.0.0                  172.16.1.15:5060         UDP  static  10
0/3:0     0.0.0.0                  192.168.24.15:5060       UDP  static  16
0/1:0     0.0.0.0                  192.168.50.25:5060       UDP  static  17
```

Using SIP Port Mapping

When you use SIP port mapping, one or more ACL entries are added to the NAT table to enable the range of ports defined. The NAT table does not support the specification of port ranges. However, it does support masking the port to enable ranges that fall on bit boundaries. For example, an entry for 192.168.24.15:4096/4 defines the port range of 4096 through 8191.

The algorithm for determining the set of ACLs for the port map range balances the need to represent the range as closely as possible, with the need to minimize the number of ACL entries. For example, a range of 30000 through 39999 would result in the following set of ACLs.

```
untrusted entries:
intf:vlan source-ip/mask:port/mask dest-ip/mask:port/mask   prot type    index
0/3:0     0.0.0.0                  192.168.24.15:30000/4    UDP  static  13
0/3:0     0.0.0.0                  192.168.24.15:32768/4    UDP  static  14
0/3:0     0.0.0.0                  192.168.24.15:36864/4    UDP  static  15
```

However, the first entry actually enables ports 28672 though 32767 and the last entry allows port 36864 through 40959. If SIP messages are received on ports outside the configured range (28672 through 29999 or 40000 through 40959 in this case), they are ignored.

Oracle recommends you use port map ranges that fall on bit boundaries to ensure the fewest possible ACL entries are created and only the configured ports are allowed by the ACLs. For example, a range of 32768 to 49151 provides for 16,384 signaling ports in a single ACL entry (192.168.24.15:32768/2).

> **Note:**
>
> If the ACLs added for the port map range do not include the SIP port configured in the SIP interface; the normal SIP ACL entry for the SIP port is also added.

Dynamic Configuration

Dynamic configuration of SIP port mapping can cause disruption in service for existing registration cache entries; depending on the changes made to the defined port map range. If the range of mapping ports is reduced, it is possible that SIP signaling messages from the registrar/softswitch realm will no longer be sent to the host processor because of the changes in the NAT Table ACL entries.

When the range of mapping ports is changed, any signaling ports in the free signaling port pool not allocated to a registration cache entry are removed from the pool. When an allocated signaling port that is no longer part of the defined mapping port range is released, it is not returned to the pool of free steering ports.

The administrator is warned when the changed configuration is activated after the port map range of a SIP interface has been changed.

Registration Statistics

The SIP registration cache statistics include counters for free and allocated signaling ports. You can issue a show registration command to display the statistics:

```
17:36:55-190
SIP Registrations          -- Period -- -------- Lifetime --------
                  Active     High   Total       Total   PerMax     High
User Entries           4        4       0           7        4        4
Local Contacts         4        4       0           7        4        4
Free Map Ports     12284    12284       0       12291    12288    12288
Used Map Ports         4        4       0           7        4        4
Forwards               -        -       1          22        4
Refreshes              -        -       3          43        3
Rejects                -        -       0           0        0
Timeouts               -        -       0           1        1
Fwd Postponed          -        -       0           0        0
Fwd Rejected           -        -       0           0        0
Refr Extension         0        0       0           0        0        0
Refresh Extended       -        -       0           0        0
```

The labels for the first two items reflect the restructured registration cache:

- User Entries: counts the number of unique SIP addresses of record in the cache. Each unique address of record represents a SIP user (or subscriber). The address of record is taken from the To header in the REGISTER request. There might be one or more registered contacts for each SIP user. The contacts come from the Contact header of the REGISTER request.
- Local Contacts: counts the number of contact entries in the cache. Because the same user can register from multiple endpoints (user agents); the number of Local Contacts might be higher than the number of User Entries.
- Free Map Ports: counts the number of ports available in the free signaling port pool.
- Used Map Ports: counts the number of signaling ports allocated for registration cache entries. The value of Used Map Ports will equal the number of Local Contacts when the port mapping feature is used for all registrar/softswitch realms in the Oracle Communications Session Border Controller.

SIP Port Mapping Configuration

You configure the SIP port mapping feature on a per-realm basis using the SIP interface configuration. Configure the port map range on the SIP interface for the realm where the registrar/softswitch resides. Port mapping is only applied when the access/ingress realm has registration caching and/or HNT enabled.

The range of SIP mapping ports must not overlap the following:

- Configured SIP port, which might be used for signaling messages not associated with a registered endpoint.
- Port range defined for steering pool configuration using the same IP address as the SIP interface. If overlap occurs, the NAT table entry for the steering port used in a call prevents SIP messages from reaching the host processor.

To configure SIP port mapping:

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

```
ORACLE# configure terminal
```

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

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

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

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

4. **port-map-start**—Set the starting port for the range of SIP ports available for SIP port mapping. The valid range is 1025 through 65535. The default values is **0** and when this value is set, SIP port mapping is disabled. The valid range is:
   - Minimum: 0, 1025
   - Maximum: 65535

```
ORACLE(sip-interface)# port-map-start 32768
```

5. **port-map-end**—Set the ending port for the range of SIP ports available for SIP port mapping. The valid range is 1025 through 65535. If you set the value to the default **0**, SIP port mapping is disabled. The valid range is:
   - Minimum—0, 1025
   - Maximum—65535

> **Note:**
>
> If not set to zero (0), the ending port must be greater than the starting port.

```
ORACLE(sip-interface)# port-map-end 40959
```

6. **options**—If you want to use SIP port mapping based on IP address, set the options parameter by typing **options**, a Space, the option name **reg-via-key** with a plus sign in front of it, type the equal sign and the word **all**. Then press Enter.

```
ORACLE(sip-interface)# options +reg-via-key=all
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

7. Save your work using the ACLI **done** command.

The following example shows SIP port mapping configured for a SIP interface:

```
sip-interface
        state                           enabled
        realm-id                        backbone
        sip-port
                address                         192.168.24.15
                port                            5060
                transport-protocol              UDP
                allow-anonymous                 all
        sip-port
                address                         192.168.24.15
                port                            5060
                transport-protocol              TCP
                allow-anonymous                 all
        carriers
        proxy-mode
        redirect-action
        contact-mode                    none
        nat-traversal                   none
        nat-interval                    30
        registration-caching            enabled
        min-reg-expire                  120
        registration-interval           3600
        route-to-registrar              enabled
        teluri-scheme                   disabled
        uri-fqdn-domain
        trust-mode                      agents-only
        max-nat-interval                3600
        nat-int-increment               10
        nat-test-increment              30
        sip-dynamic-hnt                 disabled
        stop-recurse                    401,407
        port-map-start                  32768
        port-map-end                    40959
        last-modified-date              2005-09-23 14:32:15
```

SIP Port Mapping for TCP and TLS

In releases prior to S-C6.2.0, the Oracle Communications Session Border Controller (SBC) supports SIP port mapping for UDP and now you can enable this feature for SIP sessions using TCP and TLS. Port mapping enables the SBC to allocate a unique port number for each endpoint registering through it by giving it a transport address (or hostport) in the registered Contact.

When you enable this feature for TCP and TLS, the SBC designates a port from a configured range for each endpoint that registers with SIP servers in the SIP interface's realm. You establish that range of ports using the **port-map-start** and **port-map-end** parameters. Unlike its behavior with UDP port mapping—where the SBC sends requests on the SIP interface from the allocated port mapping, the SBC sends all requests over an existing connection to the target next hop for TCP/TLS port mapping. If a connection does not exist, the system creates one. So for TCP/TLS port mapping, only the Contact header contains the transport address of the mapping port (i.e., the transport address of the configured SIP port). And the system refuses TCP and TLS connections on the allocated mapping port.

With TCP/TLS port mapping enabled, the SBC sends the Path header with the transport address in Register requests, unless you specify that it should not do so. Standards-conformant SIP servers (that support RFC 3327) might attempt to send requests to the allocated mapping port if the Path header is absent.

> **Note:**
>
> ACL entries in the NAT table that permit TCP/TLS signaling for a SIP port configuration with TCP/TLS port mapping are the same as they would be for a TCP/TLS SIP port without port mapping enabled. Additional ACL entries that need to be set up for UDP port mapping are not required for TCP/TLS port mapping.

RTN 1684

SIP Port Mapping Configuration for TCP TLS

You enable TCP/TLS port mapping in a per-realm basis using the SIP interface configuration; setting the **tcp-port-mapping** value in the **options** parameter enables the feature. Enabling this parameter turns on the port mapping feature for UDP as well.

By default, the Oracle Communications Session Border Controller includes the Path header in Register requests it sends from that SIP interface. If you do not this header to be included, however, you can set the value as **tcp-port-mapping=nopath**.

To enable TCP/TLS port mapping for a SIP interface:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing configuration, you will need to select and edit it.

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

4. **options**—Set the options parameter by typing options, a Space, the option name **tcp-port-mapping** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +tcp-port-mapping
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save your work.

SIP Configurable Route Recursion

When the Oracle Communications Session Border Controller routes SIP requests from a UAC to a UAS, it might determine that there are multiple routes to try based on a matching local policy. The Oracle Communications Session Border Controller recurses through the list of routes in a specific order according to your configuration and the quality of the match. There are other scenarios when a UAS replies with a 3xx Redirect response to the Oracle Communications Session Border Controller, the 3xx response can include multiple Contacts to which the request should be forwarded in a specific order. In both cases, the Oracle Communications Session Border Controller needs to recurse through a list of targets.

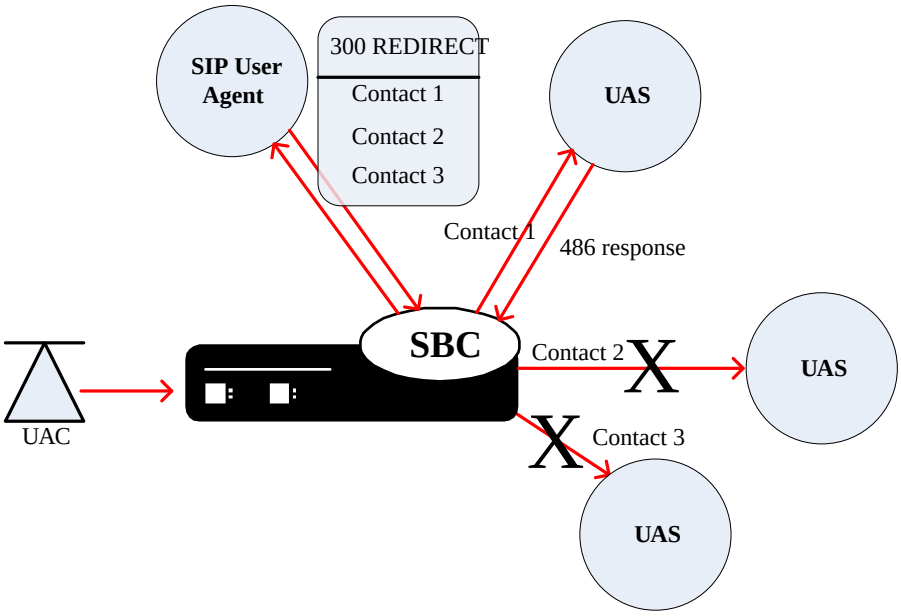
When the Oracle Communications Session Border Controller receives a non-successful (or non-6xx response) final response from the UAS, and there are multiple targets for the original request, the Oracle Communications Session Border Controller will forward the request to the next target and wait for a response. While the process of forwarding the request to multiple targets as explained in the previous paragraph is called serial forking, and the process of forwarding the request to contacts received in redirect responses is called recursion, the term recursion is used for both processes in this notice.

Use the SIP Route Recursion feature when you want the Oracle Communications Session Border Controller to forward a response to the UAC and stop recursing through the target list immediately after receiving the 3xx, 4xx, or 5xx response code that you configure. When this feature is disabled, the Oracle Communications Session Border Controller only stops recursing when it receives a message with a 401 or 407 response code. Using this feature, you can configure a specific message or range of messages to stop recursing on when received. The Oracle Communications Session Border Controller retains its default behavior to stop recursing on a 401 or 407 response code when SIP Route Recursion is configured on a SIP interface. The Oracle Communications Session Border Controller will always stop recursing when it receives a global failure (6xx); this behavior is not configurable.

You can disable response recursion for either a SIP interface or for a SIP session agent, providing you with flexibility for various network architectures. For instance, a PSTN gateway might be the only hop to reach a given endpoint, whereas several session agents might need to be contacted if multiple devices map to a contacted address of record.

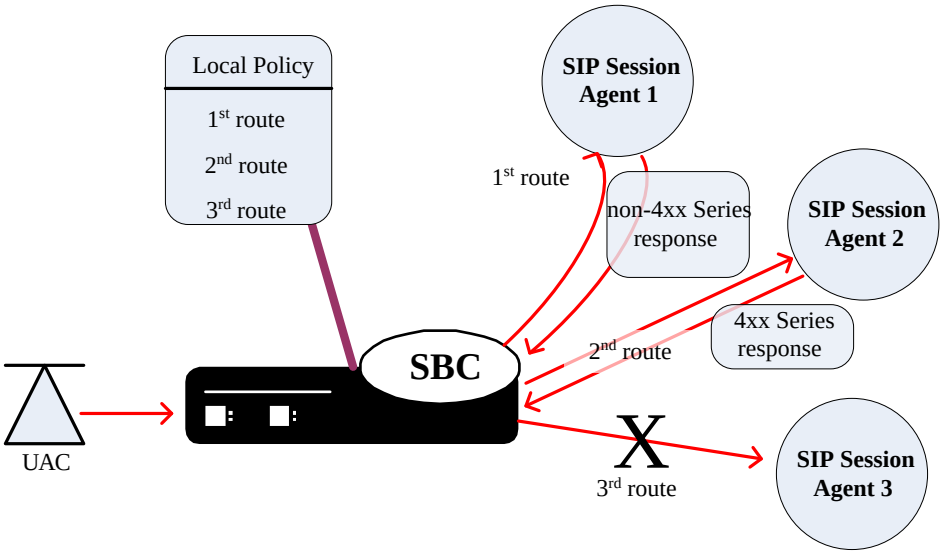
Example 1

A more detailed example is when a softswitch might return a list of contacts for multiple PSTN gateways in a Redirect message. If the PSTN target number contacted on redirection is busy, a 486 response will be sent to the Oracle Communications Session Border Controller. Since the single target is located in the PSTN, a subsequent request through a different gateway will yield another 486 response. The Oracle Communications Session Border Controller should be configured to return the 486 response to the UAC immediately. No other SIP requests should be sent to applicable targets/contacts that were enumerated in the redirect list. See the following example:



Example 2

The Oracle Communications Session Border Controller might determine from a local policy lookup that several routes are applicable for forwarding a SIP message. The Oracle Communications Session Border Controller will try each route in turn, but the SIP response recursion disable feature can be implemented to stop the route recursion when a configured responses message is received by the Oracle Communications Session Border Controller. See the following example:



There are a few conditions on the parameter used to configure response recursion:

- SIP Route Recursion is configurable for either the SIP interface or session agent.
- 401 and 407 are preconfigured for all configured SIP interfaces. They are not configured for session agents.
- The format is a comma-separated list of response codes or response code ranges: 404, 484-486.

- Only response codes that fall within the 3xx, 4xx, and 5xx range may be specified.

SIP Route Recursion Configuration

You enable SIP route recursion either in the session agent or the SIP interface configuration.

Configuring a Session Agent for SIP Route Recursion

To configure SIP Route recursion for an existing session agent:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. Select the session agent where you want this feature.

   ```
   ORACLE(session-agent)# select
   <hostname>:
   1: asd          realm=      ip=1.0.0.0
   2: SIPSA        realm=      ip=10.10.102.1
   selection:2
   ORACLE(session-agent)#
   ```

5. **stop-recurse**—Enter list of returned response codes that this session agent will watch for in order to stop recursion on the target's or contact's messages. This can be a comma-separated list or response code ranges.

   ```
   ORACLE(session-agent)# stop-recurse 404,484-486
   ```

6. Save and activate your changes.

Configuring a SIP Interface for SIP Route Recursion

To configure SIP route recursion for an existing SIP interface:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. Select the SIP interface to which you want to apply this feature.

```
ORACLE(sip-interface)# select
<realm-id>:
1: Acme_Realm
selection:1
ORACLE(sip-interface)#
```

5. **stop-recurse**—Enter a list of returned response codes that this SIP interface will watch for in order to stop recursion on the target's or contact's messages. This list can be a comma-separated list of response codes or response code ranges.

```
ORACLE(sip-interface)# stop-recurse 404,484-486
```

6. Save and activate your changes.

SIP Event Package Interoperability

Service providers often deploy a Oracle Communications Session Border Controller on the border of an access network, where it sits between the SIP endpoints (user agents) and the service provider's application server. The application server and the user agents sometimes use various SIP event packages to exchange and maintain state information. The SUBSCRIBE and NOTIFY methods are used to establish subscriptions to the event packages and to report state changes to the subscribing entity.

The SIP global contact option addresses interoperability in the Dialog and Presence event packages that are used in hosted PBX and IP Centrex offerings. State information is passed in the message body of a NOTIFY request; this message body is encoded in an XML format described by the Content-Type header. The Oracle Communications Session Border Controller needs to update certain fields in the body to account for dialog mapping and SIP NAT functionality between the access and service provider realms. Often the subscriptions are established using URIs learned from Contact headers in the user agent registrations or dialog establishment (INVITE/SUBSCRIBE). For this, a Oracle Communications Session Border Controllerrequires a Contact URI that is usable and routable outside of an existing dialog.

The SIP global contact option enables persistent URIs in the Contact headers inserted into outgoing SIP messages. If this option is not used, URIs placed in the Contact header of outgoing messages are only valid within the context of the dialog to which the message is associated.

RFCs associated with this feature are:

- A. B. Roach, *Session Initiation Protocol (SIP)-Specific Event Notification*, RFC 3265, June 2002
- J. Rosenberg, *A Presence Event Package for the Session Initiation Protocol (SIP)*, RFC 3856, August 2004
- J. Rosenberg, et al. *Data Format for Presence Using XML*, http://www.iptel.org/info/players/ietf/presence/outdated/draft-rosenberg-impp-pidf-00.txt, Work In Progress (expired), June 2000

- J.Rosenberg, H. Schulzrinne, R. Mahy, *An INVITE Initiated Dialog Event Package for the Session Initiation Protocol (SIP)*, draft-ietf-sipping-dialog-package-06.txt, Work In Progress, April 2005
- H. Sugano, et al., *Presence Information Data Format (PIDF)*, RFC 3863, August 2004

SIP Event Package Interoperability Configuration

This feature is applicable to the global SIP configuration.

To configure SIP event package interoperability:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

4. **options**—Add SIP event package interoperability support to a new SIP configuration or to an existing SIP configuration:

   If you do not currently have an SIP configuration, you can add the option by typing options, a Space and then global-contact.

   ```
   ORACLE(sip-config)# options global-contact
   ```

   Select the SIP configuration so that you can add SIP event package interoperability support to it. Then, to add this option to a list of options that you have already configured, type options followed by a Space, the plus sign (+), and the global-contact option.

   ```
   ORACLE(sip-config)# select
   ORACLE(sip-config)# options +global-contact
   ```

   If you type options global-contact without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your changes.

SIP Proxy Subscriptions

When the Oracle Communications Session Border Controller operates in dialog mode (i.e., as a B2BUA), it creates and maintains dialog state for subscription dialogs created with SUBSCRIBE/NOTIFY messages and for INVITE-initiated dialogs. Since there can be a very large number of subscriptions per user in a Rich Communication Services (RCS) environment (especially for presence subscriptions), Oracle Communications Session Border Controller resources can quickly become depleted.

To alleviate this consumption of resources, you can configure your **Oracle Communications Session Border Controller** to operate in proxy mode for event packages that you define using the **proxy-sub-event** parameter in the global SIP configuration. When you define event packages in this list and the operation mode for the SIP configuration is dialog or session, the **Oracle Communications Session Border Controller** processes all SUBSCRIBE and NOTIFY requests and responses for the designated event packages in transaction stateful mode.

Topology Hiding

So that it can perform topology hiding, the Oracle Communications Session Border Controller retains necessary routing information (such as the Contact or Record-Route header values) and it encodes certain data from the messages it receives in the messages it sends. To be more specific, the Oracle Communications Session Border Controller encodes the original URI hostport and the ingress realm name into a gr parameter it adds to the URI. The hostport information is replaced with the IP address and port of the SIP interface from which the message is sent (the egress interface). Without this information, subsequent in-dialog messages cannot be routed correctly because the Oracle Communications Session Border Controller does not retain dialog state (i.e., the route-set or remote-target).

For example, the URI sip:td@192.168.24.121:5060 might be encoded as sip:td@192.168.24.121:5060; gr=vjml9qtd175bhmhvhkgp0jov81popvbp000040.

The Oracle Communications Session Border Controller also performs URI encoding on the message body for Content-Type application/pdif+xml. this contains a Presence Information Data Format document (or PIDF) in PUBLISH and NOTIFY requests so subsequent SIP requests using the URIs in the PIDF document can be routed correctly.

The Oracle Communications Session Border Controller performs URI encoding in outgoing SIP messages after SIP=NAT is applied and before outbound HMR occurs. And the system decodes URIs in SIP messages after inbound HMR takes place and before SIP-NAT is applied.

In the event a URI is encoded several times (as is the case in spiral and hairpin calls), the encoded realm+hostport values are separated by a plus sign (+), as in the following:

```
sip:td@192.168.24.121:5060; gr=vjml9qtd175bhmhvhkgp+dhfhb0jov81opvbp
```

When the Oracle Communications Session Border Controller receives any of the following requests, it matches the contents of the request's Event header with the list you configure in the proxy-sub-events parameter:

- PUBLISH
- SUBSCRIBE
- NOTIFY
- REFER

This is provided the operation-mode for the SIP configuration is set to either **session** or **dialog**. If it finds a match, the Oracle Communications Session Border Controller marks the request for processing in transaction-stateful mode rather than in B2BUA mode.

> **Note:**
>
> Although PUBLISH is not a dialog-creating request, topology hiding needs to be applied to the PIDF so that subsequent NOTIFY requests containing portions of the published PIDF can be decoded properly.

When the Oracle Communications Session Border Controller forwards the request, it will have encoded all Contact and Record-Route header information using the ingress realm. The hostport value of the URIs then has egress SIP interface's IP address and port. The Via headers in the requested the Oracle Communications Session Border Controller received are not included in the outgoing request.

Then PUBLISH, SUBSCRIBE, NOTIFY, and REFER responses are compared to the request that was sent to determine if the response should receive transaction-stateful proxy treatment. The Oracle Communications Session Border Controller decodes any encoded Record-Route headers back to their original values for the outgoing response. Any Record-Route headers added downstream from the Oracle Communications Session Border Controller are encoded using the original request's egress realm (meaning the realm from which the response was received). In addition, the Contact header is encoded using the request's egress realm and ingress SIP interface and port.

Feature Interaction

When using this feature, the Oracle Communications Session Border Controller does not keep dialog or subscription state. Therefore the Per-user SUBSCRIBE Dialog Limit feature—configured in the **enforcement-profile** configuration—will not function properly when a subscription is handled in proxy mode.

SIP Proxy Subscription Configuration

This section shows you how to configure a list of SIP event packages to cause the Oracle Communications Session Border Controller to act in proxy mode.

> **Note:**
>
> The operation-mode parameter for the global SIP configuration must be set to either dialog or session in order for this feature to function as designed.

To configure a list of SIP event packages to enable SIP proxy subscriptions:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **proxy-sub-events**—Enter a list of SIP event package names that you want to enable the SIP proxy subscriptions feature. You can enter more than one value by enclosing multiple values in quotations marks, as in the following example.

```
ORACLE(sip-config)# proxy-sub-events presence winfo
```

SIP REGISTER Forwarding After Call-ID Change

This feature addresses the case when an endpoint reboots and performs a third party registration before its old registration expires. During this reregistration, the contact header is the same as it was pre-reregistration. As a consequence of the reboot, the SIP Call-ID changes. In this situation, the Oracle Communications Session Border Controller does not forward the REGISTER to the registrar, because it believes the endpoint is already registered, based on a previous registration from the same Contact: header URI.

To remedy this problem, the Oracle Communications Session Border Controller now keeps track of the Call-ID in its registration cache. The **forward-reg-callid-change** option in the global SIP configuration element forces the Oracle Communications Session Border Controller to forward a REGISTER message to the registrar when the Call-ID header changes in a REGISTER message received from a reregistering UAC.

SIP REGISTER Forwarding Configuration

To configure SIP REGISTER forwarding after a Call-ID change:

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

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

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

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

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

4. **options—**Add this feature to a new or an existing SIP configuration:

   If you do not currently have a SIP configuration, you can add the option by typing options, a Space, and then forward-reg-callid-change.

```
ORACLE(sip-config)# options forward-reg-callid-change
```

For an existing SIP configuration, select the SIP configuration so that you can add this feature to it. Then, to add this option to a list of options that you have already configured, type options, a Space, the plus sign (+), and the forward-reg-callid-change option.

```
ORACLE(sip-config)# options +forward-reg-callid-change
```

If you type options forward-reg-callid-change without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your changes.

SIP Local Response Code Mapping

The SIP local response code mapping feature enhances the SIP response code mapping. The SIP response code map feature lets you establish a table that maps SIP response-received messages (entries) to response-to-send messages (entries).

SIP local response code mapping is used with the SIP responses generated by the Oracle Communications Session Border Controller towards a specific SIP session agent. This feature lets you provision the mapping of the response codes used by the Oracle Communications Session Border Controller when it generates the responses towards a session agent.

You create the SIP local response code map using the existing mapping functionality, and then assigning that map to a session agent or to a SIP interface.

> **Note:**
>
> The configured response map is not used when the Oracle Communications Session Border Controller is acting as proxy for the responses to this session agent.

The parameters **method** and **register-response-expires** enable a SIP registration response mapping feature that allows you to configure the Oracle Communications Session Border Controller to remap a SIP failure response—which it receives from another network device or that it generates locally—to a 200 OK. You might want the Oracle Communications Session Border Controller to perform this type of mapping for circumstances where non-malicious endpoints continually attempt registration, but will stop (and still not be registered) when they receive a 200 OK. This response mapping does not actually register the client with the Oracle Communications Session Border Controller, meaning that there is neither a registration cache entry or a CAM ACL for it.

For the 200 OK it generates, the Oracle Communications Session Border Controller removes any Reason or Retry-After header in the 200 OK and sets the expires time. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-After expires time). You can also set this value using the **register-response-expires** parameter, but the value you set should never exceed the Register request's expires time.

SIP Local Response Code Mapping Configuration

The following instructions explain how to create the SIP response code map and then how to assign it to a specific session agent.

Creating a SIP Response Code Map

To create a SIP local response code map:

1. Access the **sip-response-map** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-response-map
ORACLE(sip-response-map)#
```

2. **name**—Enter the name of the SIP response map you want to configure.

   This value is required and must be unique.

```
ORACLE(response-map)# name busy
```

3. **entries**—Configure the entries for this mapping.

   Typing a question mark will show you the response code entry parameters that you can configure.

```
ORACLE(response-map)# entries
ORACLE(response-map-entries)#
```

   a. **recv-code**—Enter original SIP response code for the recv-mode parameter.

      The valid range is:

      - Minimum—100
      - Maximum—699

```
ORACLE(response-map-entries)# recv-mode 486
```

   b. **xmit-code**—Enter the SIP response code into which you want the original response code to be translated.

      This valid range is:

      - Minimum—100
      - Maximum—699

```
ORACLE(response-map-entries)# xmit-mode 600
```

   c. **reason**—Enter a reason for the translated code into the reason parameter.

      This response comment is sent with the translated code. Make your entry in quotation marks.

```
ORACLE(response-map-entries)# reason "Busy Everywhere"
```

   d. **method**—Enter the name of the received SIP failure response message you want to map to a 200 OK.

> **Note:**
>
> There is no default for this parameter, and leaving the parameter empty turns off the SIP registration response mapping feature.

e. **register-response-expires**—Enter the time you want to use for the expires time what mapping the SIP method you identified in the **method** parameter.

   The maximum is 999999999. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-After expires time). Any value you configure in this parameter (when not using the defaults) should never exceed the Register request's expires time.

4. Note the name that you gave the SIP response code map so that you can use it when you configure a session agent to support SIP response code mapping.
5. Save and activate your changes.

Assigning SIP Response Code Maps to Session Agents

To assign a SIP local response code map to a session agent:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

4. **local-response-map—Enter the name of the configured SIP response map that you want to use for this session-agent** and press Enter.

   ```
   ORACLE(session-agent)# local-response-map busy
   ```

5. Save and activate your configuration.

Assigning SIP Response Code Maps to SIP Interfaces

To apply SIP response codes maps to a SIP interface:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

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

3. Type **sip-interface** and press Enter.

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

4. **local-response-map**—Enter the name of the configured SIP response map that you want to apply to this SIP interface for locally-generated SIP responses. This parameter is blank by default.
5. Save and activate your configuration.

Specific Reason Headers in 503 Response

The Oracle Communications Session Border Controller issues a 503 Service Unavailable SIP response code when it fails to fulfill an apparently valid request because it is undergoing maintenance or is temporarily overloaded and so cannot process the request. This feature changes the reason phrase in the SIP response from the generic “Service Unavailable” to one that specifies the overload condition when exceeding provisioned session capacity, transcoding sessions, or DSP resources.

The Oracle Communications Session Border Controller sends SIP responses with the values of **status-code** and **sip-reason** in the **local-response-map-entries** configuration element as status text when the provisioned session capacity, transcoding sessions, or DSP resource capacity has reached its maximum limit and the capacity has been noted in the **local-error** parameter of the **local-response-map-entries** configuration element. The SBC issues the generic “Service Unavailable” when the capacity has not been noted in the in the **local-response-map-entries** configuration element. Additionally, this feature adds the values of **q.850-cause** and **q.850-reason** in the **local-response-map-entries** configuration element to the reason header when the value of **add-reason header** in the **sip-config** configuration element is **enabled**.

> **Note:**
>
> When SBC reaches the session-capacity, by default SBC sends “503 licensed-session-capacity-reached”.

Specific Reason Headers in 503 Response Configuration

1. Access the **entries** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# local-response-map
ORACLE(local-response-map)# local-response-map-entries
ORACLE(local-response-map-entries)#
```

2. Select the **entries** object to edit.

```
ORACLE(local-response-map-entries)# select


ORACLE(local-response-map-entries)#
```

3. Type **local-error** followed by a space and then one of the following phrases:
   - **transcoding-licensed-session-capacity-reached** — changes the sip-reason field in the SIP response from “Service Unavailable” to "SIP_LOCAL_ERROR_TRANSCODING_LICENSED_SESSION_CAPACITY_REACHED" when there are no more available transcoding licenses.
   - **dsp-resource-limit-reached** — changes the sip-reason field in the SIP response from “Service Unavailable” to "SIP_LOCAL_ERROR_DSP_RESOURCE_LIMIT_REACHED" when there are no more available DSP resources.
4. Type **done** to save your configuration.

Session Agent Ping Message Formatting

You can configure the user portions of the Request-URI and To: headers that define the destination of a session agent ping message, and the From: header that defines the source of a session agent ping message. These headers are sent to Oracle Communications Session Border Controller session agent. This feature is required for interoperability with certain E911 servers.

In the following example of a session agent ping-type message, you can set the user portion of the Request-URI (the text bob in the OPTIONS method line) and the user portion of the From: header (the text bob in the From: header) to the same new value. You can also set the user portion of the To: header (the text anna in the To: header) to its own new value.

```
OPTIONS sip:bob@sip.com SIP/2.0
From: UA1 <sip:bob@sip.com>
To: NUT <sip:anna@gw.sip.com>
Call-ID: 123abc@desk.sip.com
CSeq: 1 OPTIONS
Contact: <sip:UA1@client.sip.com>
Accept: application/sdp
Content-Length: 0
```

If you do not enable this feature, then the session agent ping-type message contains the text ping in all cases.

Session Agent Ping Message Formatting Configuration

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. Select the **session-agent** object to edit.

```
ORACLE(session-agent)# select
<hostname>:
1: 192.168.100.101:1813

selection: 1
ORACLE(session-agent)#
```

3. **ping-from-user-part**—Set the user portion of the From: header that defines the source of a session agent ping message.

```
ORACLE(session-agent)# ping-from-user-part bob
```

4. **ping-to-user-part**—Set the user portions of the Request-URI and To: headers that define the destination of a session agent ping message.

```
ORACLE(session-agent)# ping-to-user-part anna
```

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

SIP Session Agent Continuous Ping

You can configure the Oracle Communications Session Border Controller to use either a keep-alive or continuous method for pinging SIP session agents to determine their health—i.e., whether or not the SBC should route requests to them.

To summarize the two methods:

- keep-alive— SBC sends a ping message of a type you configure to the session agent in the absence of regular traffic.
- continuous—The SBC sends a ping message regardless of traffic state (regular or irregular); the SBC regularly sends a ping sent based on the configured ping interval timer.

By sending ping messages, the SBC monitors session agents' health and can determine whether or not to take a session out of service (OOS), leave it in service, or bring it back into service after being OOS.

When you set it to use the keep-alive mode of pinging, the SBC starts sending a configured ping message to a session agent when traffic for that session agent has become irregular. The SBC only sends the ping if there are no SIP transactions with a session agent over a configurable period of time, to which the session agent's response can have one of the following results:

- Successful response—A successful response is either any SIP response code or any response code not found in the **out-service-response-codes** parameter; these leave the session agent in service. In addition, any successful response or any response in the **ping-in-service-response-codes** parameter can bring a session agent from OOS to in-service status.
- Unsuccessful response—An unsuccessful response is any SIP response code configured in the **out-service-response-codes** parameter and takes the session agent sending it OOS. Because this parameter is blank by default, the SBC considers any SIP response code successful.
- Transaction timeout—A transaction timeout happens when the session agent fails to send a response to the SBC's request, resulting in the session agent's being taken OOS.

Despite the fact that the keep-alive ping mode is a powerful tool for monitoring session agents' health, you might want to use the continuous ping method if you are concerned about the SBC not distinguishing between unsuccessful responses from next-hop session agents and ones from devices downstream from the next-hop session agent. For example, if a SIP hop beyond the session agent responds with a 503 Service Unavailable, the SBC does not detect whether a session agent or the device beyond it generated the response.

When you use the continuous ping method, only the next-hop session agent responds—preventing the request from being sent to downstream devices. The SBC also sends the ping in regular traffic conditions when in continuous ping mode, so it is certain the response comes from the next hop associated with the session agent. And in continuous ping mode, only entries for the **ping-out-service-response-codes** parameter and transaction timeouts bring session agents OOS.

> **Note:**
>
> The SBC also brings a Session Agent from OOS to in-service if it receives any SIP request from the session agent device.

By default, if the SBC does not receive a response to the ping from the session-agent, it marks it as out-of-service. You can configure the number of ping failures that the SBC can receive before it marks the session-agent as out-of-service. This is achieved by configuring the OPTIONS parameter in the session-agent with a ping-failure value, where value is the number of ping response failures. This is true for both the keep-alive and continuous-ping modes.

For example, set the Options parameter by typing **options**, followed by a Space, the option name: **ping-failure-count=N** (where N is the number of ping response failures before the SA is set to OOS) and press Enter. You may prepend the option parameter with a plus sign to add and not replace this option to the existing realm-config option list.

```
ORACLE (session-agent) #options +ping-failure-count=3
```

The session-agent is set out of service after the third ping response failure.

To remove the ping-failure-count option configuration, enter options **-ping**

```
ORACLE (session-agent) #options -ping
```

SIP SA Continuous Ping Configuration

You can set the ping mode in the session agent or session constraints configuration. The default for the ping-send-mode parameter is keep-alive.

Configure Ping Mode for Session Agents

1. Access the session-agent configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. If you are adding to an existing configuration, then you will need to select the configuration you want to edit.
3. **ping-send-mode**—If to want to use continuous ping mode to send ping messages to session agents in regular traffic conditions, set this parameter to **continuous**. If you want to use the keep-alive mode, leave this parameter set to **keep-alive** (default).
4. Save and activate your configuration.

Configure Ping Mode for Session Constraints

1. Access the **session-constraints** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-constraints
ORACLE(session-agent)
```

2. If you are adding to an existing configuration, then you will need to select the configuration you want to edit.
3. **ping-send-mode**—If to want to use continuous ping mode to send ping messages to session agents in regular traffic conditions, set this parameter to **continuous**. If you want to use the keep-alive mode, leave this parameter set to **keep-alive** (default).
4. Save and activate your configuration.

SIP PAI Stripping

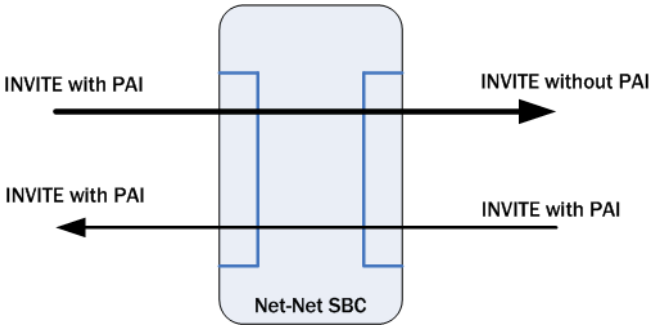
The Oracle Communications Session Border Controller now has the ability to strip P-Asserted-Identity (PAI) headers so that service providers can ensure an extra measure of security against malicious users pretending to be legitimate users. To pretend to represent another account, the malicious users simply send an INVITE with an imitation PAI. This feature allows real-time detection of such fraudulent use.

This feature uses a combination of:

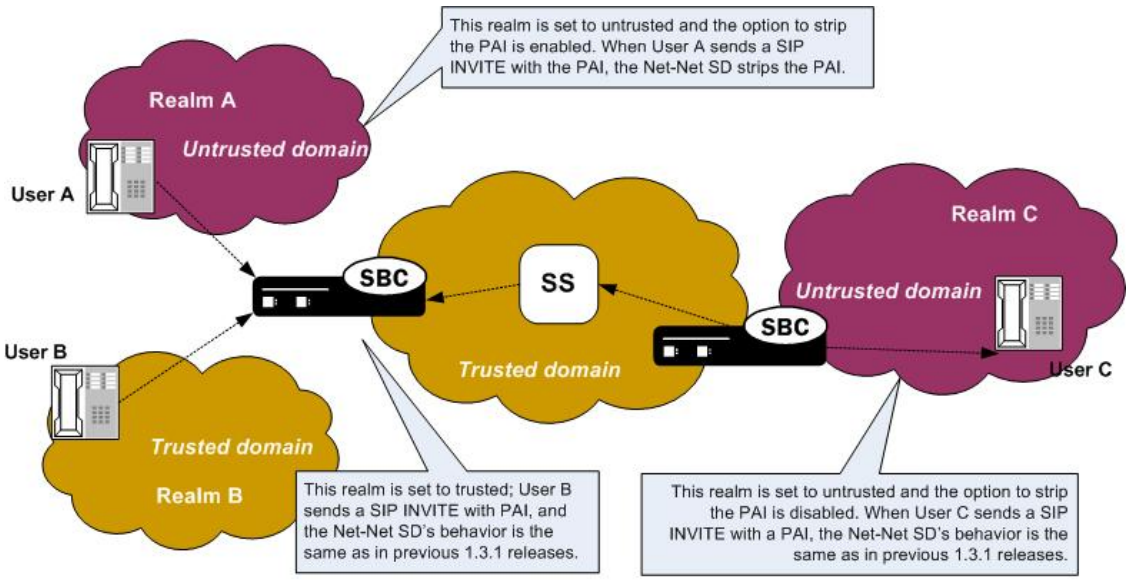
- DoS protection applied on a per-realm basis
- SIP PAI header stripping

The combination of these settings can produce different results for the SIP PAI stripping feature.

- SIP PAI header stripping enabled for an untrusted realm—If the PAI stripping parameter is set to enabled in a realm that is untrusted, then the Oracle Communications Session Border Controller strips the PAI headers from SIP INVITEs that are received from the external address, regardless of the privacy type. The Oracle Communications Session Border Controller then sends the modified INVITE (without the PAI). If the INVITE comes from a trusted realm, then the Oracle Communications Session Border Controller does not strip the PAI header and the system behaves as it does when you are using previous 1.3.1 releases.



- Multiple SIP PAIs in a SIP INVITE—The Oracle Communications Session Border Controller removes all PAIs when there are multiple PAIs set in SIP INVITEs that come from untrusted realms.
- Oracle Communications Session Border Controller behavior bridging trusted and untrusted realms—The following graphics shows you how Oracle Communications Session Border Controllers can be positioned and configured to handle PAI stripping between trusted and untrusted realms.

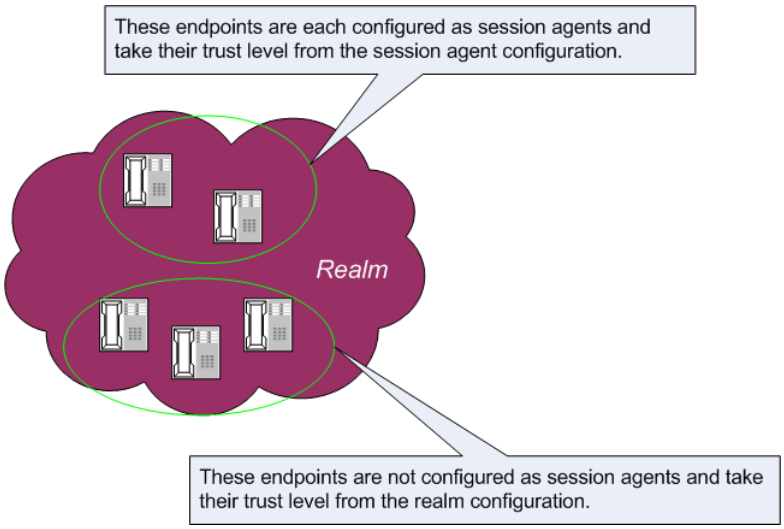


Realm Configuration Settings	REALM A	REALM B	REALM C
Realm designation trusted or untrusted (trust-me)	Disabled	Enabled	Enabled
SIP PAI stripping (pai-strip)	Enabled	Enabled or disabled	Disabled
SBC's behavior	Strip PAI regardless of privacy type	Same as behavior for SIP privacy support in previous 1.3.1 releases	Same as behavior for SIP privacy support in previous 1.3.1 releases

SIP PAI Stripping Configuration

When you configure this feature, please note how the Oracle Communications Session Border Controller behaves when you combine the designation of a realm as trusted/untrusted and SIP PAI stripping is enabled. Enter the choices for the ACLI **trust-me** and **pai-strip** parameters accordingly.

Be aware that trust is also established in the session agent configuration, and that the trust level set in a session agent configuration overrides the trust set in a realm configuration. For example, a realm might have several endpoints, some of which are associated with session agents and some of which are not. The endpoints that have configured session agent will take their trust level from the session agent parameters you set; the other endpoints, ones that are not associated with session agents, take their trust level from the realm parameters you set.



Take this relationship into consideration when you configure SIP PAI header stripping, or this feature will not work as designed.

For the sample configuration cited below, the desired Oracle Communications Session Border Controller behavior is to always strip the PAI regardless of privacy type.

To configure SIP PAI stripping for an existing realm using the ACLI:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-manager path.

   ```
   ORACLE(configure)# media-manager
   ```

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

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

4. Select the realm to which you want to apply this feature.

```
ORACLE(realm-config)# select
identifier:
1: acmePacket <none>           192.168.20.0/24
2: realm1     <none>           0.0.0.0
selection:2
ORACLE(realm-config)#
```

5. **pai-strip**—Enable PAI stripping. The default is **disabled**. Valid values are:
   - enabled | disabled

```
ORACLE(realm-config)# pai-strip enabled
```

6. Save your work using the ACLI **save** or **done** command.

SIP Statuses to Q.850 Reasons

This section explains the Oracle Communications Session Border Controller's ability to map Q.850 cause values with SIP responses, a feature used in SIP calls and calls that require IWF.

RFC 3326 defines a header that might be included in any in-dialogue request. This reason header includes cause values that are defined as either a SIP response code or ITU-T Q.850 cause values. You can configure the Oracle Communications Session Border Controller to support sending and receiving RFC 3326 in SIP messages for:

- Mapping H.323 Q.850 cause values to SIP responses with reason header and cause value
- Mapping SIP response messages and RFC 3326 reason header and cause
- Locally generated SIP response with RFC 3326 reason header and cause

As specified in RFC 3326, the Oracle Communications Session Border Controller sends SIP responses to the softswitch that contain the received Q.850 cause code and the reason.

Though the Oracle Communications Session Border Controller can generate RFC 3326 headers, the default behavior for this feature is disabled. Furthermore, the Oracle Communications Session Border Controller can receive and pass SIP error messages (4xx, 5xx, and 6xx) that contain the SIP reason header with a Q.850 cause code and reason (as specified in RFC 3326). If the system receives an error message without the Reason header, then the Oracle Communications Session Border Controller is not required to insert one.

In calls that require IWF, the Q.850 cause generated in the SIP response are the same as the cause received in the following H.225 messages: Disconnect, Progress, Release, Release Complete, Resume Reject, Status, and Suspend Reject. In addition, the Q.850 cause codes that the Oracle Communications Session Border Controller receives in RFC 3326 headers are passed to the H.323 part of the call unmodified; the H.323 call leg uses this cause code for releasing the call.

SIP-SIP Calls

The SIP Reason header might appear in any request within a dialog, in a CANCEL request, and in any response where the status code explicitly allows the presence of this header field. The syntax of the header follows the standard SIP parameter:

```
Reason: SIP;cause=200;text="completed elsewhere"
Reason: Q.850;cause=16;text="Terminated"
```

This feature attends to the following possible SIP call scenarios:

- When the Oracle Communications Session Border Controller receives a SIP request or SIP response that contains the Reason header, the Oracle Communications Session Border Controller passes it without modification.
- When it generates a SIP response, the Oracle Communications Session Border Controller includes the RFC 3326 Reason header containing a Q.850 cause code and reason. This is the case for all local conditions and for all internally generated error responses (4xx, 5xx, and 6xx) to an initial SIP INVITE.
  Possible local error scenarios are:
  - invalid-message
  - cpu-overloaded
  - media-released
  - media-not-allocated

Configure Reason and Cause Mapping for SIP-SIP Calls

To configure reason-cause mapping for SIP-SIP calls, you must set up the ACLI local-response-map configuration with appropriate entries to generate the SIP response and the Q.850 cause code value used for particular error scenarios. If you want to add a Reason header, you must enable that capability in the global SIP configuration.

In the following procedure use the method parameter and the register-response-expires parameter to enable a SIP registration response mapping feature that allows you to configure the system to remap a SIP failure response to a 200 OK. The failure response can come from another network device or the system can generate the response locally. You might want the system to perform such mapping when a non-malicious endpoint continually attempts registration, but stops when the system sends a 200 OK. The failure response mapping does not register the client with the system, which leaves neither a registration cache entry nor a CAM ACL for the entry.

For the 200 OK it generates, the system removes any Reason or Retry-After header in the 200 OK and sets the expires time. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-After expires time). You can also set this value using the register-response-expires parameter, but the value you set should never exceed the Register request's expires time.

1. Access the **entries** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# local-response-map
   ```

```
ORACLE(local-response-map)# local-response-map-entries
ORACLE(local-response-map-entries)#
```

2. (Optional) Type **?** to see the entire menu for the local response map entries configuration.
3. **local-error**—Set the local error that triggers the use of this local response map. No default.

> **Note:**
>
> The Enterprise and Service Provider systems both display the full list, but some items do not apply to both systems. Such items are noted.

4. Type **local-error ?** to see the entire list of local errors that you can configure.
5. **sip-status**—Set the SIP response code to use. No default value. Range: 100-699.
6. **sip-reason**—Set the SIP reason string you want to use for this mapping. No default value. If the value contains spaces between the characters, you must surround the entry with quotation marks.
7. **q850-cause**—Set the Q.850 cause. No default value. Range: 0-2147483647.
8. **q850-reason**—Set the Q.850 reason string that you want to use for this mapping. No default value. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
9. method—Enter the name of the received SIP failure response message you want to map to a 200 OK.

   No default. Leave the parameter empty to turn off the SIP registration response mapping feature.
10. register-response-expires—Enter the time you want to use for the expires time you identified in the method parameter.

    The maximum is 999999999. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-After expires time). Any value you configure in this parameter (when not using the defaults) should never exceed the Register request's expires time.
11. (Optional) Repeat this process to create the number of local response map entries that you need.
12. Save and activate the configuration for changes to take effect.

Configure the System to Add Reason Headers

To enable the SBC to add the Reason header:

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

   ```
   ORACLE# configure terminal
   ```

2. Type session-router and press Enter.

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

3. Type sip-config and press Enter.

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

4. add-reason-header—Enable this parameter to add the Reason header(s) to responses and CDRs.

   The default value is disabled. The valid values are:

   - enabled | disabled

5. Save and activate your configuration for changes to take effect.

Calls Requiring IWF

For interworking calls between SIP and H.323, you can configure:

- Mappings for SIP status codes to Q.850 values
- Mappings for particular Q.850 cause codes to SIP status codes

If it cannot find the appropriate mapping, then the Oracle Communications Session Border Controller uses default mappings defined in the Default Mappings table below.

The following describes how the Oracle Communications Session Border Controller handles different IWF call scenarios:

- SIP request containing a Reason header—When it receives a request containing a Reason header, the Oracle Communications Session Border Controller determines if the request is a SIP BYE or SIP CANCEL message. RFC 3326 states that the Reason header is mainly used for these types of requests. If there is a Reason header and it contains the Q.850 cause value, then the Oracle Communications Session Border Controller releases the call on the H.323 side using the specified cause value.
- SIP response—When it receives the error response to an initial SIP INVITE, the Oracle Communications Session Border Controller uses its SIP-Q.850 map to determine the Q.850 that it will use to release the call. If there is not a map entry, then the Oracle Communications Session Border Controller uses the default mappings shown in the Default Mappings table.
- Active call released from the H.323 side—If an active call is released from the H.323 side, the Oracle Communications Session Border Controller checks the outgoing realm (the SIP side) to see if the addition of the Reason header is enabled. If it is, then the Oracle Communications Session Border Controller adds the Reason header in the SIP BYE request with the Q.850 value it received from the H.323 side.
- Error during setup of the call on the H.323 side—In the event of an error during setup on the H.323 side of the call, the system needs to send:
  - An error response, if this is a SIP to H.323 call
  - A SIP CANCEL, if this is a H.323 to SIP call and the H.323 side hangs up before the call is answered on the SIP side
    In this case, the Oracle Communications Session Border Controller checks to see if adding the Reason header is enabled in the IWF configuration. If it is, then the Oracle Communications Session Border Controller adds the Reason header with the Q.850 cause value it received from the H.323 side.
- Call released due to a Oracle Communications Session Border Controller error—If the call is released due a Oracle Communications Session Border Controller error and adding the Reason header is enabled in the IWF configuration, the error response to the initial INVITE

contains the Reason header. The Oracle Communications Session Border Controller checks the SIP to Q.850 map configurations to determine whether or not the SIP error response code it is generating is configured. If it is, then the Oracle Communications Session Border Controller maps according to the configuration. If if it not, the Oracle Communications Session Border Controller derives cause mapping from the default table.

Like the configuration for SIP-only calls that enable this feature, you can set a parameter in the IWF configuration that enables adding the Reason header in the SIP requests or responses.

Default Mappings

This table defines the default mappings the Oracle Communications Session Border Controller uses when it cannot locate an appropriate entry that you have configured.

Q.850 Cause Value Number	Q.850 Cause Value Number	SIP Status Number	SIP Status Text	Comments
1	Unallocated number	404	Not found	N/A
2	No route to specified transit network	404	Not found	N/A
3	No route destination	404	Not found	N/A
16	Normal calling clearing	N/A	BYE message	A call clearing BYE message containing cause value 16 normally results in the sending of a SIP BYE or CANCEL request. However, if a SIP response is to be sent to the INVITE request, the default response code should be used.
17	User busy	486	Busy here	N/A
18	No user responding	408	Request timeout	N/A
19	No answer from the user	480	Temporarily unavailable	N/A
20	Subscriber absent	480	Temporarily unavailable	N/A
21	Call rejected	603	Decline (if location filed in Cause information element indicates user; otherwise 403 Forbidden is used)	N/A
22	Number changed	301	Moved permanently (if information in diagnostic field of Cause information element is suitable for generating SIP Contact header; otherwise 410 Gone is used)	N/A
23	Redirection to new destination	410	Gone	N/A
25	Exchange routing error	483	Too many hops	N/A
27	Destination out of order	502	Bad gateway	N/A
28	Address incomplete	484	Address incomplete	N/A

Q.850 Cause Value Number	Q.850 Cause Value Number	SIP Status Number	SIP Status Text	Comments
29	Facility rejected	501	Not implemented	N/A
31	Normal, unspecified	480	Temporarily unavailable	N/A
34	No circuit, channel unavailable	503	Service unavailable	N/A
38	Network out of order	503	Service unavailable	N/A
41	Temporary failure	503	Service unavailable	N/A
42	Switching equipment congestion	503	Service unavailable	N/A
47	Resource unavailable unspecified	503	Service unavailable	N/A
55	Incoming calls barred with CUG	403	Forbidden	N/A
57	Bearer capability not authorized	403	Forbidden	N/A
58	Bearer capability not presently available	503	Service unavailable	N/A
65	Bearer capability not implemented	488	Not acceptable here	N/A
69	Requested facility not implemented	501	Not implemented	N/A
70	Only restricted digital information available	488	Not acceptable here	N/A
79	Service or option not implemented, unspecified	501	Not implemented	N/A
87	User not member of CUG	403	Forbidden	N/A
88	Incompatible destination	503	Service unavailable	N/A
102	Recovery on timer expiry	504	Server time-out	N/A

SIP Status

To configure a SIP status to Q.850 Reason with cause mapping:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

3. Type **sip-q850-map** and press Enter.

```
ORACLE(session-router)# sip-q850-map
ORACLE(sip-q850-map)#
```

4. Type **entries** and press Enter.

```
ORACLE(sip-q850-map)# entries
ORACLE(sip-q850-map-entry)#
```

From here, you can view the entire menu for the SIP status to Q.850 Reason with cause mapping entries configuration by typing a **?**.

5. **sip-status**—Set the SIP response code that you want to map to a particular Q.850 cause code and reason. There is no default, and the valid range is:
   - Minimum—100
   - Maximum—699
6. **q850-cause**—Set the Q.850 cause code that you want to map to the SIP response code that you set in step 5. There is no default. Range: 0-2147483647.
7. **q850-reason**—Set the Q.850 reason corresponding to the Q.850 cause code that you set in step 6. There is no default. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
8. Repeat this process to create the number of local response map entries that you need.
9. Save and activate your configuration for changes to take effect.

To configure a Q.850 cause to a SIP status with reason mapping:

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

```
ORACLE# configure terminal
```

11. Type **session-router** and press Enter.

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

12. Type **sip-q850-map** and press Enter.

```
ORACLE(session-router)# q850-sip-map
ORACLE(q850-sip-map)#
```

13. Type **entries** and press Enter.

```
ORACLE(q850-sip-map)# entries
ORACLE(q850-sip-map-entry)#
```

From here, you can view the entire menu for the Q.850 cause to a SIP response code with reason mapping entries configuration by typing a **?**.

14. **q850-cause**—Set the Q.850 cause code that you want to map to a SIP status with reason. There is no default.
15. **sip-status**—Set the SIP response code to which you want to map the Q.850 cause that you set in step 5. There is no default, and the valid range is:

- Minimum—100
- Maximum—699

16. **sip-reason**—Set the reason that you want to use with the SIP response code that you specified in step 6. There is no default. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
17. Repeat this process to create the number of local response map entries that you need.
18. Save and activate your configuration for changes to take effect.

    To enable the Oracle Communications Session Border Controller to add the Reason header for calls that require IWF:

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

    ```
    ORACLE# configure terminal
    ```

20. Type **session-router** and press Enter.

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

21. Type **iwf-config** and press Enter.

    ```
    ORACLE(session-router)# iwf-config
    ACMEPACKET(iwf-config)#
    ```

22. **add-reason-hdr**—Enable this parameter to add the Reason header. The default is **disabled**. The valid values are:

    - enabled | disabled

23. Save and activate your configuration for changes to take effect.

Trunk Group URIs

The Oracle Communications Session Border Controller's trunk group URI feature, applicable for SIP and IWF signaling services, enables the capabilities related to trunk groups that are described in this section. This implementation follows the IPTEL draft Representing Trunk Groups in Tel/SIP Uniform Resource Identifiers (URIs) (draft-ietf-iptel-trunk-group-06.txt), and also supports more customized approaches.

- For a typical access call flow scenario, when the calling party's call arrives at the Oracle Communications Session Border Controller, the Oracle Communications Session Border Controller formulates a SIP INVITE message that it sends to a softswitch. The Oracle Communications Session Border Controller now supports a new URI contact parameter in the SIP request message so that service providers need to be able to:
  - Determine from where the Oracle Communications Session Border Controller received the call
  - Signal information about the originating gateway from a Oracle Communications Session Border Controller to a softswitch (e.g., an incoming trunk group or a SIP gateway to a Oracle Communications Session Border Controller)
- This feature supports the signaling of routing information to the Oracle Communications Session Border Controller from network routing elements like softswitches. This information tells the Oracle Communications Session Border Controller what egress route (or outgoing trunk groups) it should choose for terminating next hops/gateways. For this purpose, new SIP URI parameters in the Request-URI are defined. Additional URI

parameters include the network context to identify the network in which the originating or terminating gateway resides.

- Especially important for large business applications, this feature can free Oracle Communications Session Border Controller resources by reducing the number of local policy, session agent, and session agent group configurations. By enabling the trunk group URI feature, the Oracle Communications Session Border Controller instead uses a routing scheme based on signaled SIP URI information.

Terminology

The following IPTEL terms are used in the descriptions of and instructions for how to configure this feature:

- Trunk—In a network, a communication path connecting two switching systems used in the establishment of an end-to-end connection; in selected applications, it may have both its terminations in the same switching system
- Trunk group—A set of trunks, traffic engineered as a unit, for the establishment of connections within or between switching systems in which all of the paths are interchangeable except where sub-grouped
- Trunk group name—Provides a unique identifier of the trunk group; referred to as tgrp
- Trunk group context—Imposes a namespace by specifying a domain where the trunk groups are; also referred to simply as context

Trunk Group URI Parameters

Trunk group URI parameters identify originating and terminating trunk group information in SIP requests.

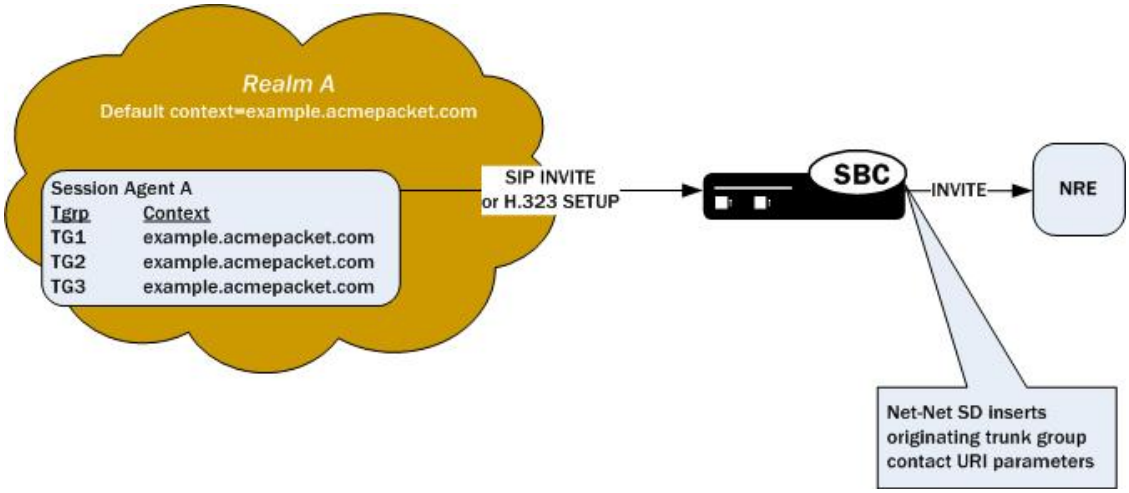
In the absence of official SIP standards for transporting trunk groups between signaling elements, the Oracle Communications Session Border Controller allows you to define URI parameters for use with originating and terminating trunk group URIs.

Originating Trunk Group URI Parameters and Formats

You can configure session agents and session agents groups on the SBC to insert trunk group URI parameters in the SIP contact header. When SIP gateways comply with the IPTEL draft, they include the originating URI parameter in the SIP contact header. For those SIP and H.323 gateways that are not compliant, the SBC inserts SIP trunk group URI parameters on the gateway's behalf.

When there are no applicable session agent or session agent group configurations, the SBC uses the source IP address of the endpoint or gateway as the trunk group name (tgrp) parameter in the originating trunk group URI.

The following diagram shows a scenario where the SBC inserts originating trunk group URI parameters.



There are two available formats for the originating trunk group URIs:

1. In compliance with the IPTEL draft, the first format has two parameters: tgrp (identifier of the specific trunk group) and trunk-context (defines the network domain of the trunk group). These appear in the following formats:
   - tgrp=”trunk group name”
   - trunk-context=”network domain”

     The URI BNF for would appear as it does in the example directly below, where the tgrp is tg55 and the trunk-context is trunk-context is telco.example.com:

     ```
     tel:+15555551212;tgrp=tg55;trunk-context=telco.example.com
     ```

2. The second format is customized specifically for access URIs and contains two provisioned parameters: tgrp (or tgname) and context (or provstring). This appears as tgrp.context, where these definitions apply:
   - tgrp (tgname)—Provisioned trunk group name for the originating session agent; this value must have at least one alphabetical character, cannot contain a period (.), and can contain a hyphen (-) but not as the first or the last character
   - context (provstring)—Name of the originating trunk group context; this value must have at least one alphabetical character in the top label

     This format conforms to format for a hostname in the SIP URI as specified in RFC 3261, such that a trunk group identifier would appear as:

     ```
     custsite2NY-00020.type2.voip.carrier.net
     ```

     where the tgrp is custsite2NY-00020, and the context is type2.voip.carrier.net.

     The BNF for an access URI conforms to the following:

     SIP-URI = "sip:" [userinfo] hostport uri-parameters [headers]

     uri-parameters = *(";" uri-parameter)

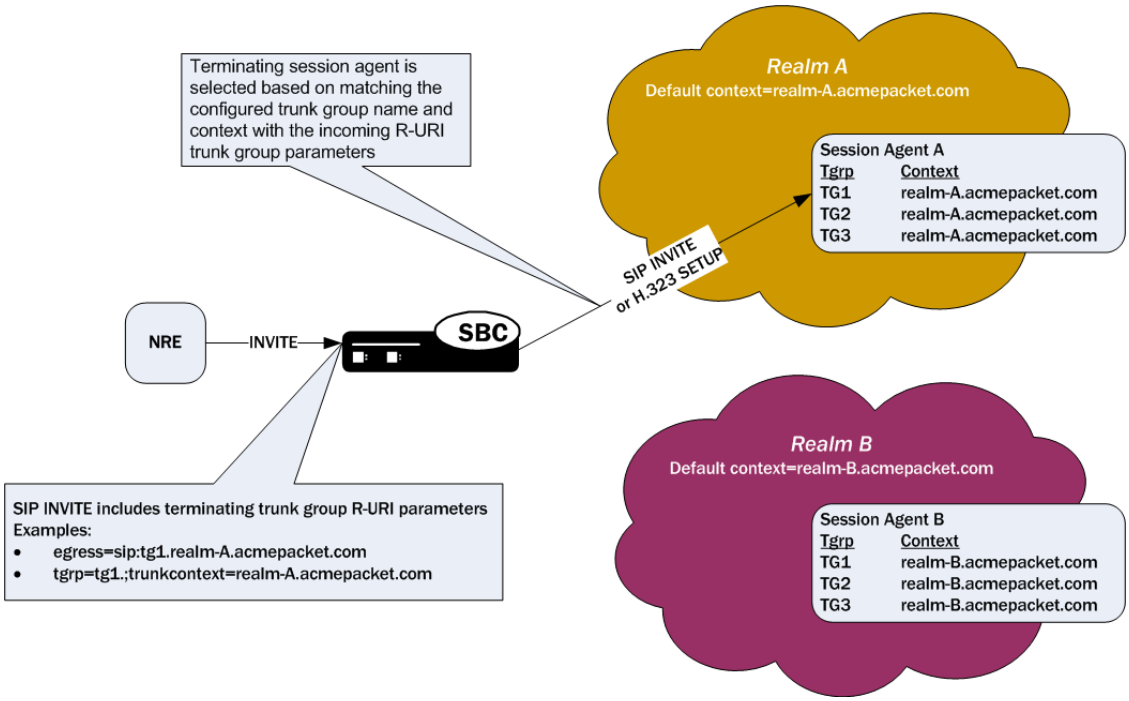
     uri-parameter = transport-param / user-param / method-param

/ ttl-param / maddr-param / lr-param / other-param

```
other-param = accessid / pname [ '=' pvalue ]
accessid = "access=" accessURI
accessURI = scheme tgname ["." provstring]
scheme = "sip:" / token
tgname = ALPHA / *(alphanum) ALPHA *(alphanum / "-") alphanum /
    alphanum *(alphanum / "-") ALPHA *(alphanum) # up to 23 characters
provstring = *(domain ".") toplabel # up to 24 characters
toplabel = ALPHA / ALPHA *( alphanum / "-" ) alphanum
domain = alphanum/ alphanum *( alphanum / "-" ) alphanum
```

Terminating Trunk Group URI Parameters and Formats

Terminating trunk group URI parameters appear in the R-URI, and they can be included in by a network routing element to instruct the SBC which egress trunk groups to use. By matching the trunk group URI parameter with configured session agents or session agent groups, the SBC can locate the terminating gateway. The trunk group name can also be expressed as the IP address of the terminating gateway.



In the absence of official SIP standards for transporting trunk groups between signaling elements, the SBC allows you to define the URI parameters used in terminating trunk groups.

There are two available formats for the terminating trunk group URIs:

1. In compliance with the IPTEL draft, the first format has two parameters: tgrp (which can be either a trunk group name or an IP address) and trunk-context (defines the network domain of the trunk group). These appear in the following formats:
   - tgrp=”trunk group name”
   - trunk-context=”network domain”

An example R-URI with terminating trunk group parameters appears as follows, where the tgrp is TG2-1 and the context is isp.example.net@egwy.isp.example.net:

```
INVITE sip:+15555551212;tgrp=TG2-1;trunk-
context=isp.example.net@egwy.isp.example.net SIP/2.0
```

2. The second format is customized specifically for egress URIs and contains two provisioned parameters: tgrp (or tgname) and context (or tgdomain). This appears as tgrp.context (or tgname.tgdomain), where definitions apply:

   - tgrp (tgname)—Provisioned trunk group name for the originating session agent; this value must have at least one alphabetical character, cannot contain a period (.), and can contain a hyphen (-) but not as the first or the last character
   - context (tgdomain)—Name of the terminating trunk group context; this value can be up to twenty-four characters

   The use of multiple terminating trunk groups is not supported.

   The BNF for a single, egress URI with trunk group information conforms to:

```
SIP-URI = "sip:" [userinfo ] hostport uri-parameters [headers ]
uri-parameters = *( ";" uri-parameter )
uri-parameter = transport-param / user-param / method-param
            / ttl-param / maddr-param / lr-param / other-param
other-param = egressid  / pname [ '=' pvalue ]
egressid = "egress=" egressURI
egressURI = scheme tgname ["." tgdomain]
scheme = "sip:" / token
tgname = ALPHA / *(alphanum) ALPHA *(alphanum / "-") alphanum /
    alphanum *(alphanum / "-") ALPHA *(alphanum) # up to 23 characters
tgdomain = *(domain ".") toplabel # up to 24 characters
toplabel = ALPHA / ALPHA *( alphanum / "-" ) alphanum
domain = alphanum/ alphanum *( alphanum / "-" ) alphanum
```

   For all trunk group URI support, you must set the appropriate parameters in the SIP manipulations configuration and in the session agent or session agent group configurations.

   In the originating trunk group URI scenario, a call arrives at the SBC from a configured session agent or session agent group. If this session agent or session agent group has the appropriate trunk group URI parameters and inbound manipulation rules configured, the SBC then looks to the SIP manipulations configuration and add the trunk group URI information according to those rules. Those rules tell the SBC where and how to insert the trunk group URI information, and the system forwards the call.

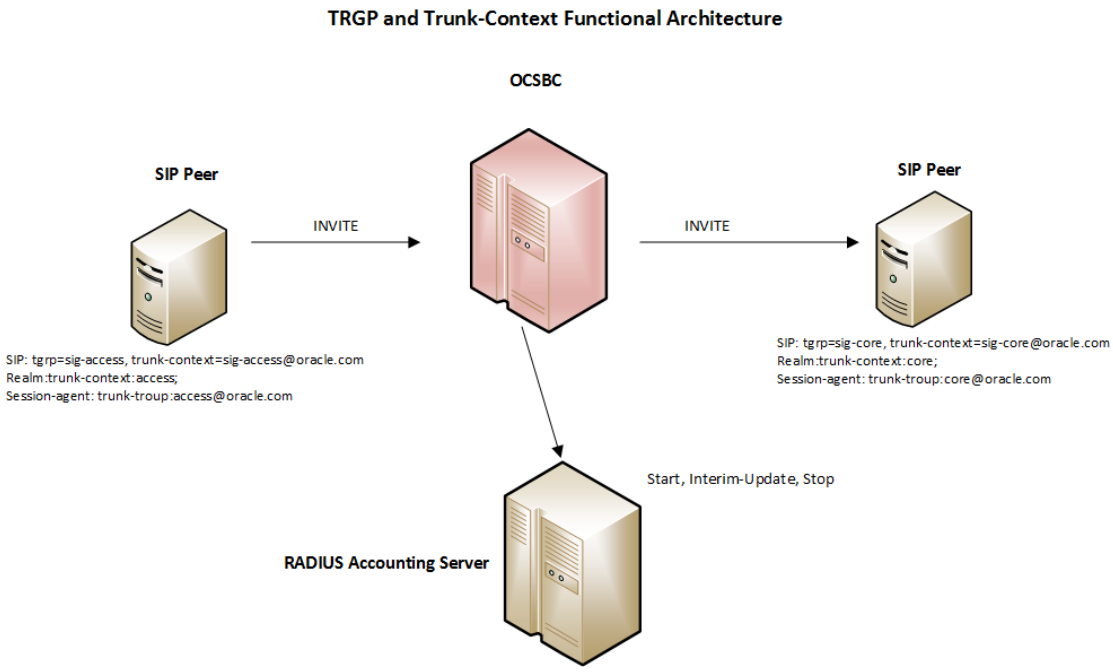
   In the terminating trunk group scenario, a call arrives at the SBC from, for instance, a call agent. This call contains information about what trunk group to use. If the information matches a session agent or session agent group that has outbound manipulation rules configured, the SBC will then look up the SIP manipulations configuration and strip information according to those rules. Those rules tell the SBC where and how to remove the information, and the SBC forwards the call.

Trunk Group Signaling Parameters

The Oracle Communications Session Border Controller supports a dynamic reading of the initial INVITE message of the session Contact header and Request URI tag parameter. This facilitates populating the Trunk Group and Trunk Context parameters into the existing AVPs.

For Ingress messages, if the **trunk-group** and the **trunk-context** parameters are configured in the session agent and the realm-configuration respectively, the Oracle Communications Session Border Controller populates these values in the Acme-Originating-Trunk-Group and the Acme-Originating Trunk-Context AVPs of the START/INTERMIDIATE/STOP accounting records. Similarly for an Egress session, if the trunk-group and the trunk-context parameters are configured in the session agent and the realm-configuration respectively, the Oracle Communications Session Border Controller populates these values in the Acme-Terminating-Trunk-Group and the Acme-Terminating-Trunk-Context AVPs of the START/INTERIM/STOP accounting records.

The following diagram shows the functional architecture of a typical access call flow scenario with Trunk Group and Trunk-Context parameters



The received SIP signaling URI trunk group parameters can be populated into the accounting AVP parameters for ingress and egress legs by adding the **populate-signaling-tgrp** option in the session-agent.

```
ORACLE#(session-agent) options +populate-signaling-tgrp
```

In the absence of the **trunk-group** and **trunk-context** parameters from the session-agent and realm-configuration respectively of the Ingress Session, if the session-agent is configured with the option **populate-signaling-tgrp**, the Oracle Communications Session Border Controller will decode the tgrp and trunk-context parameters received in the initial INVITE message 'Contact header' of the session and cache these values for the session duration. SBC will then populate these received parameter values in Acme-Originating-Trunk-Group and Acme-Originating-Trunk-Context AVPs of START/INTERMEDIATE/STOP accounting records. Similarly for the egress session, the SBC will decode the tgrp and trunk-context parameters received in first INVITE message 'Request-URI' of the session and cache these values for the session duration. SBC will then populate these received parameter values in Acme-Terminating-Trunk-Group and Acme-Terminating-Trunk-Context AVPs of START/INTERIM/STOP accounting records.

Oracle Communications Session Border Controller will cache and populate only if both tgrp and trunk-context and present in the Contact header. In the absence of even one of the parameters the system will discard the received parameter and will not populate the parameter to the corresponding AVP.

SIP Header and Parameter Manipulation

SIP header and parameter manipulation is its own configuration where you can set up rules for the addition, removal, and modification of a SIP header or the elements of a SIP header. For example, you can set up the configuration to add a URI parameter to the URI in a SIP header or replace an FQDN with in IP address. For trunk group URI support, this configuration tells the Oracle Communications Session Border Controller where and how to manipulate the SIP message to use originating (access) and terminating (egress) trunk group URI parameters.

These manipulations can be applied at the realm or at the session agent level.

Trunk Group Routing

You can configure SIP interfaces (using the ACLI **term-tgrp-mode** parameter) to perform routing based on the trunk group information received in SIP requests. There are three options: none, IPTEL, and egress URI.

- If you leave this parameter set to none (its default), the Oracle Communications Session Border Controller will not look for or route based on terminating trunk group URI parameters
- When you set this parameter to either **iptel** or **egress-uri** and the incoming request has the trunk group parameter of this type (IPTEL or egress URI), the Oracle Communications Session Border Controller will select the egress next hop by matching the “tgrp” and trunk context with a configured session agent or session agent group.
  If the received terminating trunk group URI parameters include an IP address, the egress next hop is the IP address specified. The Oracle Communications Session Border Controller determines the egress realm by matching the trunk context it receives with the trunk context you configure for the realm.
- If the incoming request does not have trunk group parameters or it does not have trunk group parameters of the type that you configure, the Oracle Communications Session Border Controller uses provisioned procedures and/or local policy for egress call routing.

The Oracle Communications Session Border Controller returns errors in these cases:

- If the terminating trunk group URI parameters do not identify a local Oracle Communications Session Border Controller session agent or session agent group, then the Oracle Communications Session Border Controller returns a SIP final response of 488 Not Acceptable Here.
- If the Oracle Communications Session Border Controller receives a SIP INVITE with terminating trunk group URI parameters that do not match the specified syntax, the Oracle Communications Session Border Controller returns a 400 final response with the reason phrase Bad Egress=Parameters.

Trunk Group URIs and SIP Registration Caching

For calls where SIP registration caching is used, you will need to set certain parameters that enable the Oracle Communications Session Border Controller to preserve trunk group URI parameters on the outgoing side.

- For SIP-SIP calls, you set the preserve-user-info **option** in the SIP interface configuration.

- For SIP-H.323 calls requiring IWF, you set the preserve-user-info-sa **option** in the session agent configuration.

Trunk Group URI Configuration

Before you configure your Oracle Communications Session Border Controller to support trunk group URIs, you need to determine:

- How you want to manipulate SIP headers (entered in the SIP header manipulations configuration)
- For terminating trunk group routing, the trunk group mode you want to use (none, IPTEL, or egress URI); this decides routing based on trunk group information
- The trunk group name and context to use entered in a session agent or session agent group configuration
- Whether you are using originating or terminating trunk group URIs (entered in the session agent configuration)
- The trunk group context for use in a realm configuration, in case the trunk group name in the session agent or session agent group does not have a context

Precedence Used for Trunk Group Configurations

The Oracle Communications Session Border Controller (SBC) can insert trunk-group/trunk-context URIs into applicable SIP messages when the upstream SIP or H.323 gateways do not. The values inserted are dependent on SBC configuration and use configuration precedence when configurations conflict.

The SBC uses the following precedence when choosing these element's trunk-group and trunk-context configuration values for routing.

1. **session-agent** (SA)
2. **session-group** (SAG)
3. **realm**

> **Note:**
>
> Do not configure a trunk context without a trunk-group.

The examples below present configurations for SA, SAG and realm and the resulting trunk group and context values to be inserted. Examples also include an explanation for the result.

Example 1 (Result—trgp1:contextg)

- SA Configuration: **trunk-group** tgrp1
- SAG Configuration: **trunk-group** tgrp2:context2
- Realm Configuration: **trunk-context** contextg

The SBC selects SA trunk-group value. The SA has no context, so the SBC uses the realm's context value.

Example 2 (Result—trgp1:context1)

- SA Configuration: **trunk-group** tgrp1:context1

- SAG Configuration: **trunk-group** tgrp2:context2
- Realm Configuration: **trunk-context** contextg

The SBC selects the SA trunk-group and context values.

Example 3 (Result—trgp2:context2)

- SA Configuration: **trunk-group** [null]
- SAG Configuration: **trunk-group** tgrp2:context2
- Realm Configuration: **trunk-context** contextg

The SBC selects the SAG trunk-group and context values.

Example 4 (Result—trgp2:contextg)

- SA Configuration: **trunk-group** [null]
- SAG Configuration: **trunk-group** tgrp2
- Realm Configuration: **trunk-context** contextg

The SBC selects the SAG trunk-group's value. The SAG has no context, so the SBC uses the realm's context value.

Example 5 (Result—none)

- SA Configuration: **trunk-group** [null]
- SAG Configuration: **trunk-group** [null]
- Realm Configuration: **trunk-context** contextg

The SBC has no configured trunk-group value and cannot support a context without a group.

Example 6 (Result—ABC)

- SA Configuration: **trunk-group** [null]
- SAG Configuration: **trunk-group** ABC
- Realm Configuration: **trunk-context** [null]

The SBC selects the SAG trunk-group's value, and does not use a context.

Configuring SIP Manipulations

When you configure the SIP header manipulations to support trunk group URIs, take note of:

- The name of the configuration, so that you can use it when you apply the manipulations in a session agent for the inbound or outbound manipulations
- The **new-value** parameter, which specifies the trunk group and trunk group context that you want to manipulate; the possible values that apply to trunk group URI configurations are $TRUNK_GROUP and $TRUNK_GROUP_CONTEXT

Setting the Trunk Group URI Mode for Routing

To set the mode for routing for terminating trunk group URIs:

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

```
ORACLE# configure terminal
```

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

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

3. Type **sip-interface** and press Enter.

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

4. **term-tgrp-mode**—Set the mode that you want to use for routing for terminating trunk group URIs. The default is **none**. Your choices are:
   - **none**—Disables routing based on trunk groups
   - **iptel**—Uses trunk group URI routing based on the IPTEL formats
   - **egress-uri**—Uses trunk group URI routing based on the egress URI format

Configuring a Session Agent for Trunk Group URIs

In a session agent, you can configure the outbound or inbound SIP header manipulation rules to use, as well as a list of trunk group names and contexts. For the trunk group names and contexts, you can use either the IPTEL or the custom format.

To configure a session agent for trunk group URIs:

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

```
ORACLE# configure terminal
```

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

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

3. Type **session-agent** and press Enter.

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

4. **out-manipulationid**—Enter the name of the SIP header manipulations configuration that you want to apply to the traffic exiting the Oracle Communications Session Border Controller via this session agent. There is no default.
5. **in-manipulationid**—Enter the name of the SIP header manipulations configuration that you want to apply to the traffic entering the Oracle Communications Session Border Controller via this session agent. There is no default.
6. **trunk-group**—In either IPTEL or custom format, enter the trunk group names and trunk group contexts to match. If you do not set the trunk group context, then the Oracle Communications Session Border Controller will use the one you set in the realm for this session agent.

   Your ACLI entries for this list must be one of these formats: tgrp:context or tgrp.context.

   To make multiple entries, surround your entries in parentheses and separate them from each other with spaces. For example:

```
ORACLE(session-agent)# trunk-group (tgrp1:context1 tgrp2:context2)
```

7. **options**—If you want to configure trunk group URIs for SIP-H.323 calls that use the IWF and you are using SIP registration caching, you might need to add the preserve-user-info-sa to your list of session agent options.

   If you are adding this option to a new session agent, you can just type **options**, a Space, and **preserve-user-info-sa**.

   If are adding this to an existing session agent, you must type a plus (+) sign before the option or you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign: options +preserve-user-info-sa.

Configuring a Session Agent Group for Trunk Group URIs

In a session agent group, you can configure the outbound or inbound SIP header manipulation rules to use, as well as a list of trunk group names and contexts. For the trunk group names and contexts, you can use either the IPTEL or the custom format.

To configure a session agent group for trunk group URIs:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type **session-group** and press Enter.

   ```
   ORACLE(session-router)# session-group
   ORACLE(session-agent-group)#
   ```

4. **trunk-group**—In either IPTEL or custom format, enter the trunk group names and trunk group contexts to match. If you do not set the trunk group context, then the Oracle Communications Session Border Controller will use the one you set in the realm for this session agent group.

   Your ACLI entries for this list must take one of these formats: tgrp:context or tgrp.context.

   To make multiple entries, surround your entries in parentheses and separate them from each other with spaces. For example:

   ```
   ORACLE(session-agent-group)# trunk-group (tgrp1:context1 tgrp2:context2)
   ```

Setting a Trunk Group Context in a Realm

You can set trunk group contexts at the realm level, which will be used by all session agents and session agent groups if there is no context specified in their configurations.

The realm trunk group URI context accommodates the IPTEL and the custom format.

To configure a trunk group context for a realm:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the session-related configurations.

```
ORACLE(configure)# media-manager
```

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

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

4. **trunk-context**—Enter the trunk group context to use for this realm. There is no default.

Using this Feature with a SIP Interface

If you are using the trunk group URIs feature with SIP interface that has registration caching enabled, then you need to configure the preserve-user-info option for that SIP interface.

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

```
ORACLE# configure terminal
```

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

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

3. Type **session-group** and press Enter.

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

4. **options**—Add support for trunk group URIs with SIP interface that uses registration caching.

   If you are adding this option to a new SIP interface, you can just type **options**, a Space, and **preserve-user-info**.

   If are adding this to an existing SIP interface**, you must type a plus (+) sign before the option or you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign: options +preserve-user-info**.

Example 1 Adding Originating Trunk Group Parameters in IPTEL Format

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to add originating trunk parameters in IPTEL format.

```
sip-manipulation
        name                            add_iptel
        header-rule
                name                            contact
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            tgrp
                        type                            uri-user-param
                        action                          add
```

```
                        match-val-type                  any
                        match-value
                        new-value                       $TRUNK_GROUP
element-rule
                        name                            trunk-context
                        type                            uri-user-param
                        action                          add
                        match-val-type                  any
                        match-value
                        new-value                       $TRUNK_GROUP_CONTEXT
```

Example 2 Adding Originating Trunk Group Parameters in Custom Format

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to add originating trunk parameters in custom format.

```
sip-manipulation
        name                            add_att
        header-rule
                name                            contact
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            egressURI
                        type                            uri-param
                        action                          add
                        match-val-type                  any
                        match-value
                        new-value
"sip:"+$TRUNK_GROUP+"."+$TRUNK_GROUP_CONTEXT
```

Example 3 Removing IPTEL Trunk Group Names

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to remove IPTEL trunk groups names.

```
sip-manipulation
        name                            strip_iptel
        header-rule
                name                            request-uri
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            tgrp
                        type                            uri-user-param
                        action                          delete-element
                        match-val-type                  any
                        match-value
                        new-value
element-rule
                        name                            trunk-context
                        type                            uri-user-param
```

```
                        action                          delete-element
                        match-val-type                  any
                        match-value
                        new-value
```

Example 4 Removing Custom Trunk Group Names

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to remove custom trunk groups names.

```
sip-manipulation
        name                            strip_egress
        header-rule
                name                            request-uri
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            egressURI
                        type                            uri-param
                        action                          delete-element
                        match-val-type                  any
                        match-value
                        new-value
```

Emergency Session Handling

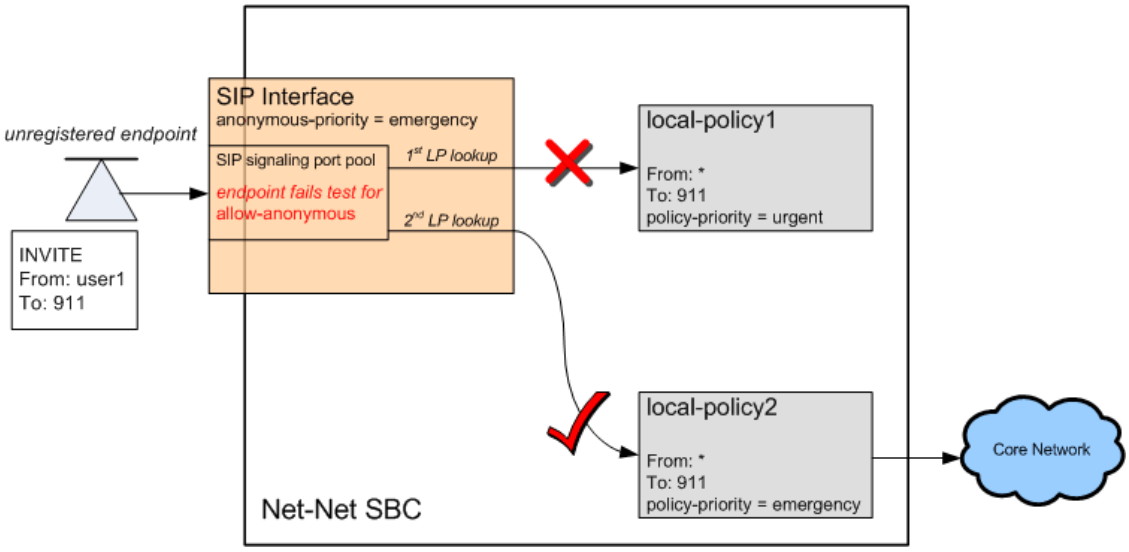
The Oracle Communications Session Border Controller provides a mechanism to handle emergency sessions from non-allowed endpoints. An endpoint is designated as non-allowed if it fails the admission control criteria specified by the allow-anonymous parameter in the SIP Ports configuration element.

When the Oracle Communications Session Border Controller receives a non-allowed emergency request, it performs a local policy lookup for a matching local policy. An emergency local policy could be configured to match if the To: header in a SIP message was addressed to 911.

An emergency policy priority selection criteria has been added to both the SIP interface and the local policy configuration elements. In the SIP interface, the parameter is called anonymous-priority. In the local policy, the parameter is called policy-priority.

For the Oracle Communications Session Border Controller to choose a local policy to route an emergency call, the emergency policy priority value on the local policy must be equal to or greater than the emergency policy priority value on the SIP interface where the emergency message was received. In this scheme, an emergency policy priority value of none is the lowest value and an emergency policy priority value of emergency is the highest.

When a match is made between all existing local policy criteria and the emergency policy priority, the emergency call will be sent to the core network according to the chosen local policy. In addition, the policy priority value of the chosen local policy is inserted into the Priority header of the core-bound SIP message..



Emergency Session Handling Configuration Procedures

Note the value of the allow-anonymous parameter in the SIP interface's SIP Ports for the incoming interface you are configuring. When an incoming emergency call from an unregistered endpoint can not be characterized by this setting, the Oracle Communications Session Border Controllerwill use the following means to route the call.

Set the anonymous-priority parameter in the incoming SIP interface. This parameter specifies that for an INVITE received from an anonymous endpoint, the Oracle Communications Session Border Controller will choose a local policy of equal or greater policy priority for outbound routing.

Next, set the policy-priority parameter located in the local-policy configuration element. Most likely, this local policy will route messages to SIP devices that act on emergency calls. The local policy is selected when its value (or above) matches the anonymous-priority parameter in the sip-interface that receives the incoming phone call from an unregistered endpoint.

The enumerated values for both the anonymous-priority and policy-priority are: none, normal, non-urgent, urgent, emergency.

Emergency Session Handling Configuration

To set the anonymous priority for a message received in a SIP interface:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

4. Type **select** and the number of the SIP interface you want to configure.

```
ORACLE(sip-interface)# select 1
```

5. **anonymous-priority—Set the policy priority for this SIP interface. It is used to facilitate emergency sessions from unregistered endpoints. This value is compared against the policy-priority parameter in the local-policy configuration element. The default is none. The valid values are:**
   - none | normal | non-urgent | urgent | emergency

     This completes the configuration.

     ```
     ORACLE(sip-interface)# anonymous-priority emergency
     ```

6. Save your work using the ACLI **done** command.

Setting Policy Priority

To set the policy priority for a local policy:

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

```
ORACLE# configure terminal
```

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

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

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

```
ORACLE(session-router)# local-policy
ORACLE(local-policy)#
```

4. Type **select** and the number of the local policy you want to configure.

```
ORACLE(local-policy)# select 1
```

5. **policy-priority—Enter the policy priority for this local policy. It is used to facilitate emergency sessions from unregistered endpoints. This value is compared against the anonymous-priority parameter in the sip-interface configuration element. The default is none. The valid values are:**
   - none | normal | non-urgent | urgent | emergency

     This completes the configuration.

     ```
     ORACLE(local-policy)# policy-priority emergency
     ```

6. Save your work using the ACLI **done** command.

Fraud Prevention

The Oracle Communications Session Border Controller can constrain outgoing SIP messages to a maximum size in bytes in order to support fraud prevention techniques. If a message does exceed the configured size, it is dropped. A SIP message can be constrained from 0 to 65535 bytes, with a default value of 4096 bytes.

Fraud Prevention Configuration

To set a maximum SIP message size:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

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

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

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

4. Type **select** to configure the existing sip config.

   ```
   ORACLE(sip-config)# select
   ```

5. **sip-message-len**—Set the size constraint in bytes of a SIP message. The default is **4096**. The valid range is:

   - Minimum—0
   - Maximum—65535

     This completes the configuration.

     ```
     ORACLE(sip-config)# sip-message-len 5000
     ```

6. Save your work using the ACLI **done** command.

Early Media Support

The Oracle Communications Session Border Controller (SBC) supports early media features, including SIP early media suppression, the Private Early Media (PEM) header, and multiple dialog management. This support complies with 3GPP TS 24.628, TS 24.182 and RFC 5009 behavior for sessions supporting early media.

Early media are the RTP/RTCP packets sent to or from the caller before a session is fully established by the receipt of a 200 OK. There are a variety of devices, including IVRs, that may generate this media. When the SBC receives an INVITE message with SDP, it can forward

media packets conforming to that SDP to the calling endpoint as soon as it forwards the INVITE to the next hop. It can also forward media packets received from the calling side toward the called endpoint as soon as it receives SDP in a SIP response to the INVITE. These responses are usually provisional messages. This allows for any early media to be played, such as remote ringback or announcements.

Early media can be unidirectional or bidirectional, and can be generated by the caller, the callee, both, or by interim AS components. Important early media concepts for which the SBC provides feature support includes:

- Private Early Media (PEM) Header Support
- Early Media Suppression
- Early Media Support for Multiple Early Dialog Scenarios
- Selecting SDP within Multi-Dialog Call Scenarios

P-Early-Media SIP Header Support

Version S-CZ7.2.0 provides support for the SIP P-Early-Media that can be used in SIP INVITES, PRACKS, and UPDATES to request and authorize the use of early media. It offers an alternative to policy-based early media support.

The P-Early-Media SIP header is defined in RFC 5009, Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media. RFC 5009 defines the use of the P-Early-Media header within SIP messages in certain SIP networks to authorize the cut-through of backward (server-to-client) and/or forward (client-to-server) early media when permitted by the early media policies of the networks involved. The P-Early-Media header field is intended for use in a SIP network, such as a 3GPP IMS that has the following characteristics: its early media policy prohibits the exchange of early media between end users; it is interconnected with other SIP networks that have unknown, untrusted, or different policies regarding early media; and it has the capability to gate (enable/disable) the flow of early media to/from user equipment.

P-Early-Media SIP Header

The P-Early-Media SIP header is defined in RFC 5009, Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media. RFC 5009 defines the use of the P-Early-Media header within SIP messages in certain SIP networks to authorize the cut-through of backward (server-to-client) and/or forward (client-to-server) early media when permitted by the early media policies of the networks involved. The P-Early-Media header field is intended for use in a SIP network, such as a 3GPP IMS that has the following characteristics: its early media policy prohibits the exchange of early media between end users; it is interconnected with other SIP networks that have unknown, untrusted, or different policies regarding early media; and it has the capability to gate (enable/disable) the flow of early media to/from user equipment.

A SIP network containing both PSTN gateways and SIP end devices, for example, can maintain such an early media policy by gating "off" any early media with a SIP end device acting as UAS, gating "on" early media with a SIP end device acting as UAC, and gating "on" early media at each PSTN gateway. This is what we have been doing for years with the SIP early media suppression feature, which allows determining who can send early media and in what direction.

Unfortunately, in SIP interconnection scenarios there is no means of assuring that the interconnected network is implementing a compatible early media policy, thus allowing the exchange of user data within early media under some circumstances. For example, if a network "A" allows all early media with user equipment as UAC and an interconnected network

"B" allows all early media with user equipment as UAS, any session established between user equipment as UAC in "A" and user equipment as UAS in "B" will allow bidirectional user data exchange as early media.

The P-Early-Media header is used for the purpose of requesting and authorizing requests for backward and/or forward early media. It's sent from UAS to UAC to indicate authorization for early media.

P-Early-Media-Header Usage

The syntax of the P-Early-Media header field is as follows.

```
P-Early-Media = "P-Early-Media" HCOLON[ em-param *(COMMA em-param) ]
       em-param  = "sendrecv" / "sendonly" / "recvonly" / "inactive" /
"gated" /
      "supported" / token
```

The P-Early-Media header is used for requesting and authorizing requests for backward and/or forward early media. The P-Early-Media header field in an INVITE request contains the "supported" parameter. If P-CSCF is part of the trusted domain, then it must decide whether to insert or delete the P-Early-Media header field before forwarding the INVITE. The P-CSCF upon receiving the P-Early-Media header field in a message towards the UAC needs to verify that the early media request comes from an authorized source. If a P-Early-Media header field arrives from either an untrusted source, a source not allowed to send backward early media, or a source not allowed to receive forward early media, then it may remove the P-Early-Media header field or alter the direction parameter(s) of the P-Early-Media header field before forwarding the message, based on local policy.

The P-Early-Media header field with the "supported" parameter in an INVITE request indicates that the P-CSCF on the path recognizes the header field. The P-Early-Media header field includes one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or "inactive", following the convention used for SDP stream directionality. Each parameter applies, in order, to the media lines in the corresponding SDP messages establishing session media. The parameter value "sendrecv" indicates a request for authorization of early media associated with the corresponding media line, both from the UAS towards the UAC and from the UAC towards the UAS. The value "sendonly" indicates request for authorization of early media from the sender to the receiver and not in the other direction. The value "recvonly" indicates a request for authorization of early media from the receiver, and not in the other direction. The value "inactive" indicates either a request that no early media associated with the corresponding media line be authorized, or a request for revocation of authorization of previously authorized early media. Each parameter applies, in order, to the media lines in the corresponding SDP lines. Unrecognized parameters are discarded and non-direction parameters are ignored. If there are more direction parameters than media lines, the excess are silently discarded. If there are fewer direction parameters than media lines, the value of the last direction parameter applies to all remaining media lines. The P-Early-Media header field in any message within a dialog towards the sender of the INVITE request can also include the non-direction parameter "gated" to indicate that a network entity on the path towards the UAS is already gating the early media, according to the direction parameter(s).

The P-Early-Media Header and the Rx Interface

As defined in 3GPP TS 24.229, IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 both the P-CSCF and IBCF may add, remove, or modify, the P-Early-Media header field within forwarded SIP requests and responses according to procedures in RFC 5009.

The P-CSCF can use the P-Early-Media header field for the gate control procedures, as described in 3GPP TS 29.214. In the current implementation, if you set the configuration option **early-media-allow** to **none**, the SBC sends the Flow-Status AVP (511) in any AAR request set to disable until there is a final response.

Although you can configure the SBC to provide PEM header information within the Flow-Status AVP (511) to provide early media status over the Rx interface, there are two conditions that must be in place before it can populate this AVP with PEM header flags.

The SBC does not include PEM header feature status in the Rx Flow-Status AVP unless:

1. You enable the **get-flow-status-from-sdp** option in the **sip-config**.

   ```
   ORACLEACMEPACKET#(media-manager) options +get-flow-status-from-sdp
   ```

2. There has been an SDP exchange. Note that provisional responses without SDP, which occur prior any SDP exchange, do not generate update data within the Rx flow-status AVP even if they include PEM status.

Functional Design

Acceptance of and authorization for early media is accomplished with two new ACLI parameters -- **p-early-media-header** and **p-early-media-direction**, which are added to SIP interface configuration in Version S-CZ7.2.0 and later releases.

The **p-early-media-header** parameter will enable the feature when the value is set to either “add” or “modify”. The **p-early-media-header** and **p-early-media-direction** should be configured on egress interface of the incoming message. The values for parameter **p-early-media-direction** are “sendrecv, sendonly, recvonly, inactive”. It is a list and each configured value corresponds to the m-line in the SDP. If the number of configured values is more than the number of m-lines in the SDP, the excess configured values are ignored. If the number of configured value is less than the number of m-lines in the SDP, the last configured value is used for all the m-lines.

The following illustrations show the ingress and egress sip-interface configuration.

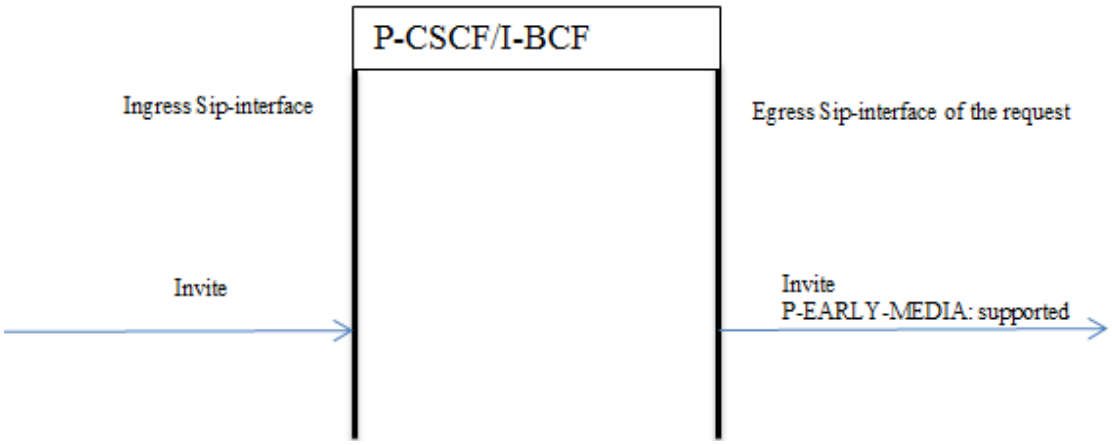
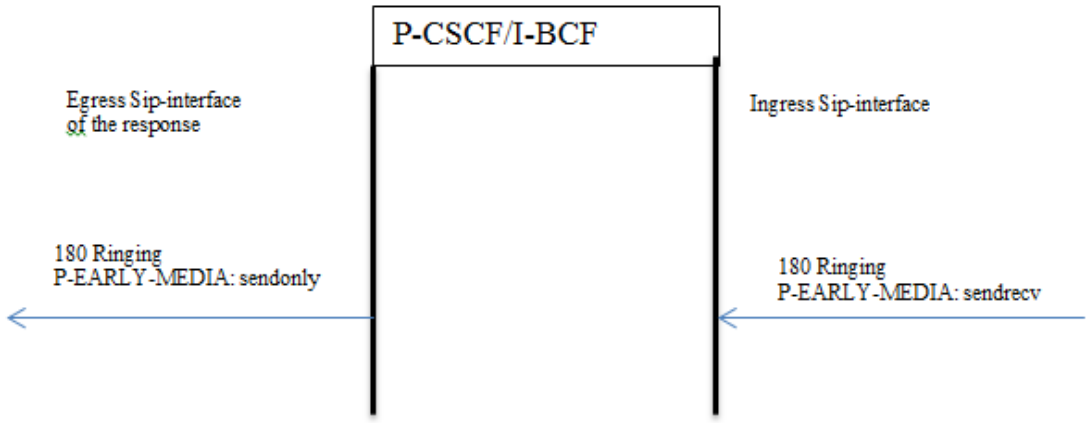


Figure 5-2 180 RINGING Specifying p-early-media Support



P-Early-Media headers in Re-Invites are ignored. If the SDP contains Content-Disposition: early-session the P-Early-Media header is ignored.

Endpoint is considered trusted or untrusted based on the configuration on the ingress sip-interface of the P-CSCF. Sip-interface has the configuration parameter “trust-mode”. If the “trust-mode” is set to “none” then nobody is trusted in sip-interface. By default the value is “all”. Possible values are <all, agents-only, realm-prefix, registered, none>.

For multiple dialogs due to forking, P-CSCF will identify the media associated with a dialog, and then setup early media flow for the selected media. The configuration elements restricted-latching in realm-config, and latching in media-router should be enabled.

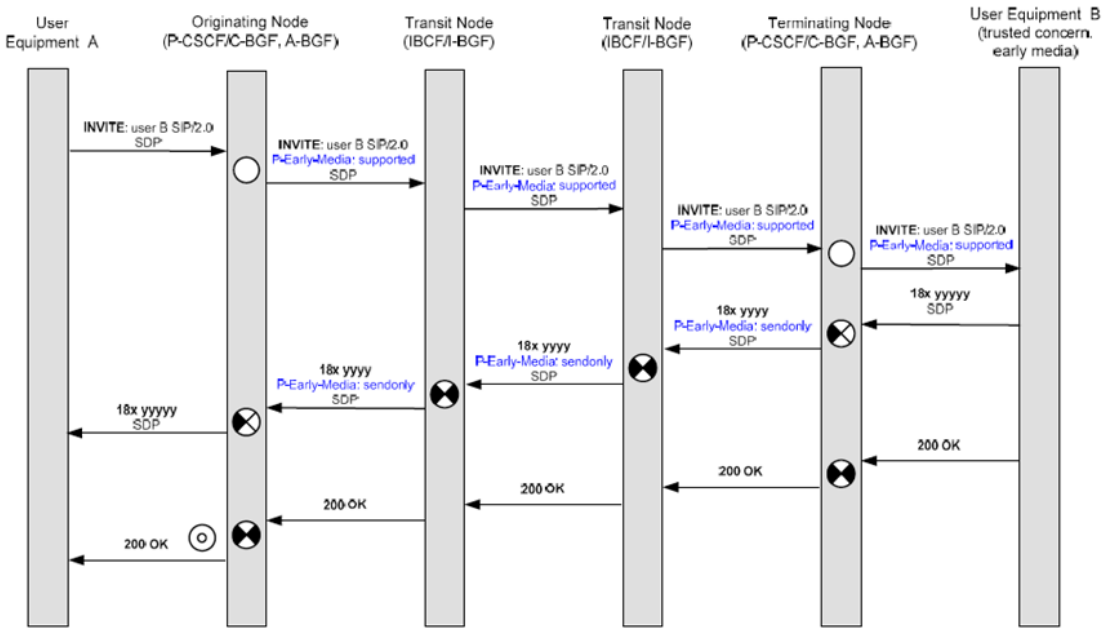
The following 3 tables detail early media implementation details.

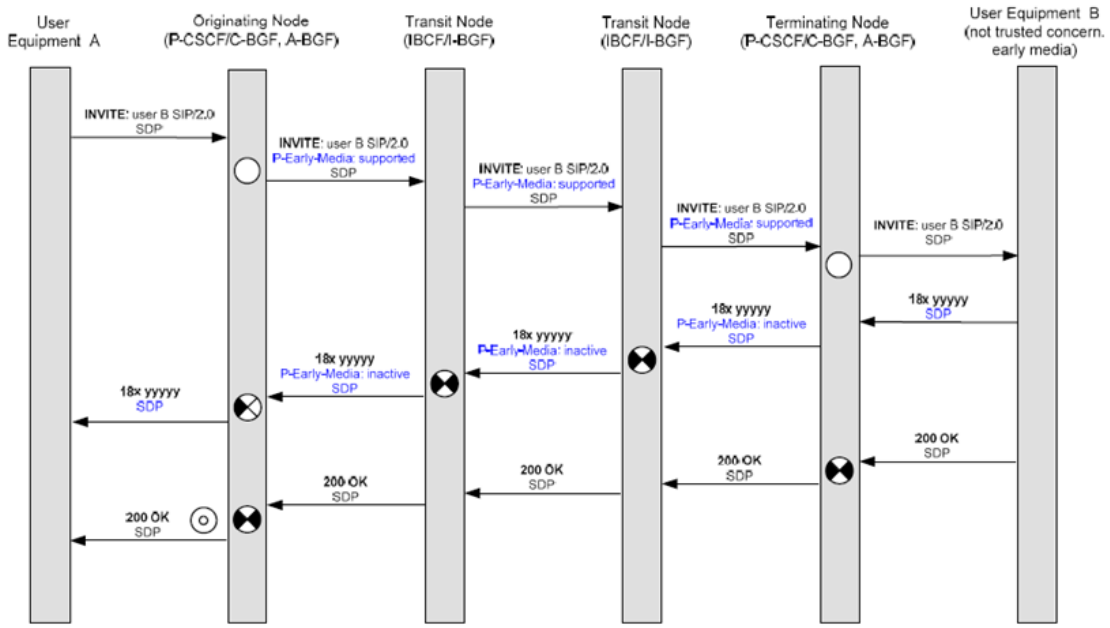
P-Early-Media Trusted to Trusted

This table illustrates the P-CSCF case when messages are received from trusted endpoints and forwarded to trusted endpoints.

Message Parameters configured on egress interface	**Request (Invite, UPDATE, PRACK) w/o header**	**Request (Invite, UPDATE, PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"**	**Response (18x, 200OK(UPDATE/ PRACK)) w/o header**	**Response (18x, 200OK(UPDATE/ PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"**
p-early-media-header-"add" p-early-media-direction -"sendonly"	Add header with "supported" value. Setup flows based on SDP value	Add "supported" value if not present. Setup the flows based on the value in the SDP value.	Setup flows based on local config PEM value. Add the PEM header based on local config value. Status Flow AVP in AAR message updated.	Setup the flows based on the value in the incoming PEM header.

Message Parameters configured on egress interface	Request (Invite, UPDATE, PRACK) w/o header	Request (Invite, UPDATE, PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"	Response (18x, 200OK(UPDATE/ PRACK)) w/o header	Response (18x, 200OK(UPDATE/ PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"
p-early-media-header-"modify" p-early-media-direction -"sendonly"	Add header with "supported" value. Setup flows based on SDP value	Add "supported" value if not present. Setup the flows based on the SDP value	Setup flows based on local config PEM value. Add the PEM header based on local config value. Status Flow AVP in AAR message updated.	Setup flows based on local config PEM value. Modify the PEM header based on local config value. Status Flow AVP in AAR message updated.
p-early-media-header-"add" No p-early-media-direction	Add header with "supported" value. Setup flows based on SDP value	Add "supported" value if not present. Setup the flows based on the SDP.	Setup flows based on local config PEM value. Add the PEM header based on default value. Status Flow AVP in AAR message updated.	Setup flows based on local config PEM value. Modify the PEM header based on default value. Status Flow AVP in AAR message updated.
p-early-media-header-"modify" No p-early-media-direction	Add header with "supported" value. Setup flows based on SDP value.	Add header with "supported" value if not present. Setup the flows based on the SDP.	config PEM value. Add the PEM header based on default value. Status Flow AVP in AAR message updated.	Setup flows based on local config PEM value. Modify the PEM header based on default value. Status Flow AVP in AAR message updated.





P-Early-Media Untrusted to Trusted

This table illustrates the P-CSCF case when messages are received from untrusted endpoints and forwarded to trusted endpoints.

Message Parameters configured on egress interface	Request (Invite, UPDATE, PRACK) w/o header	Request (Invite, UPDATE, PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"	Response (18x, 200OK(UPDATE/ PRACK)) w/o header	Response (18x, 200OK(UPDATE/ PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"
p-early-media-header-"add" p-early-media-direction -"sendonly"	Add header with "supported" value. Setup flows based on SDP value	Discard the header. Add the header with supported value. Setup flows based on SDP value.	Setup flows based on local config PEM value. Add the PEM header based on local config value. Status Flow AVP in AAR message updated.	Discard the header. Setup flows based on local config PEM value. Add the PEM header based on local config value. Status Flow AVP in AAR message updated.
p-early-media-header-"modify" p-early-media-direction -"sendonly"	Add header with "supported" value. Setup flows based on SDP value	Discard the header. Add the header with supported value. Setup flows based on SDP value.	Setup flows based on local config PEM value. Add the PEM header based on local config value. Status Flow AVP in AAR message updated.	based on local config PEM value. Add the PEM header based on local config value. Status Flow AVP in AAR message updated.
p-early-media-header-"add" No p-early-media-direction	Add header with "supported" value. Setup flows based on SDP value.	Discard the header. Setup flows based on SDP value.	Setup flows based on local config PEM value. Add the PEM header based on default value. Status Flow AVP in AAR message updated.	Discard the header. Setup flows based on local config PEM value. Add the PEM header based on default value. Status Flow AVP in AAR message updated.
p-early-media-header-"modify" No p-early-media-direction	Add header with "supported" value. Setup flows based on SDP value	Discard the header. Setup flows based on SDP value.	Setup flows based on local config PEM value. Add the PEM header based on default value. Status Flow AVP in AAR message updated.	Discard the header. Setup flows based on local config PEM value. Add the PEM header based on default value. Status Flow AVP in AAR message updated.

P-Early-Media Trusted to Untrusted

This table illustrates the P-CSCF case when messages are received from trusted endpoints and forwarded to untrusted endpoints.

Message Parameters configured on egress interface	Request (Invite, UPDATE, PRACK) w/o header	Request (Invite, UPDATE, PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"	Response (18x, 200OK(UPDATE/ PRACK)) w/o header.	Response (18x, 200OK(UPDATE/ PRACK) with header with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"
p-early-media-header-"add" p-early-media-direction -"sendonly"	Setup flows based on SDP value.	Discard the header. Setup flows based on SDP value.	Setup flows based on SDP value.	Discard the header. Setup flows based on local config PEM value. Status Flow AVP in AAR message updated.
p-early-media-header-"modify" p-early-media-direction -"sendonly"	Setup flows based on SDP value.	Discard the header. Setup flows based on SDP value.	Setup flows based on SDP value.	Discard the header. Setup flows based on local config PEM value. Status Flow AVP in AAR message updated.
p-early-media-header-"add" No p-early-media-direction	Setup flows based on SDP value.	Discard the header. Setup flows based on SDP value.	Setup flows based on SDP value.	Discard the header. Setup flows based on default PEM value. Status Flow AVP in AAR message updated.
p-early-media-header-"modify" No p-early-media-direction	Setup flows based on SDP value.	Discard the header. Setup flows based on SDP value.	Setup flows based on SDP value.	Discard the header. Setup flows based on default PEM value. Status Flow AVP in AAR message updated.

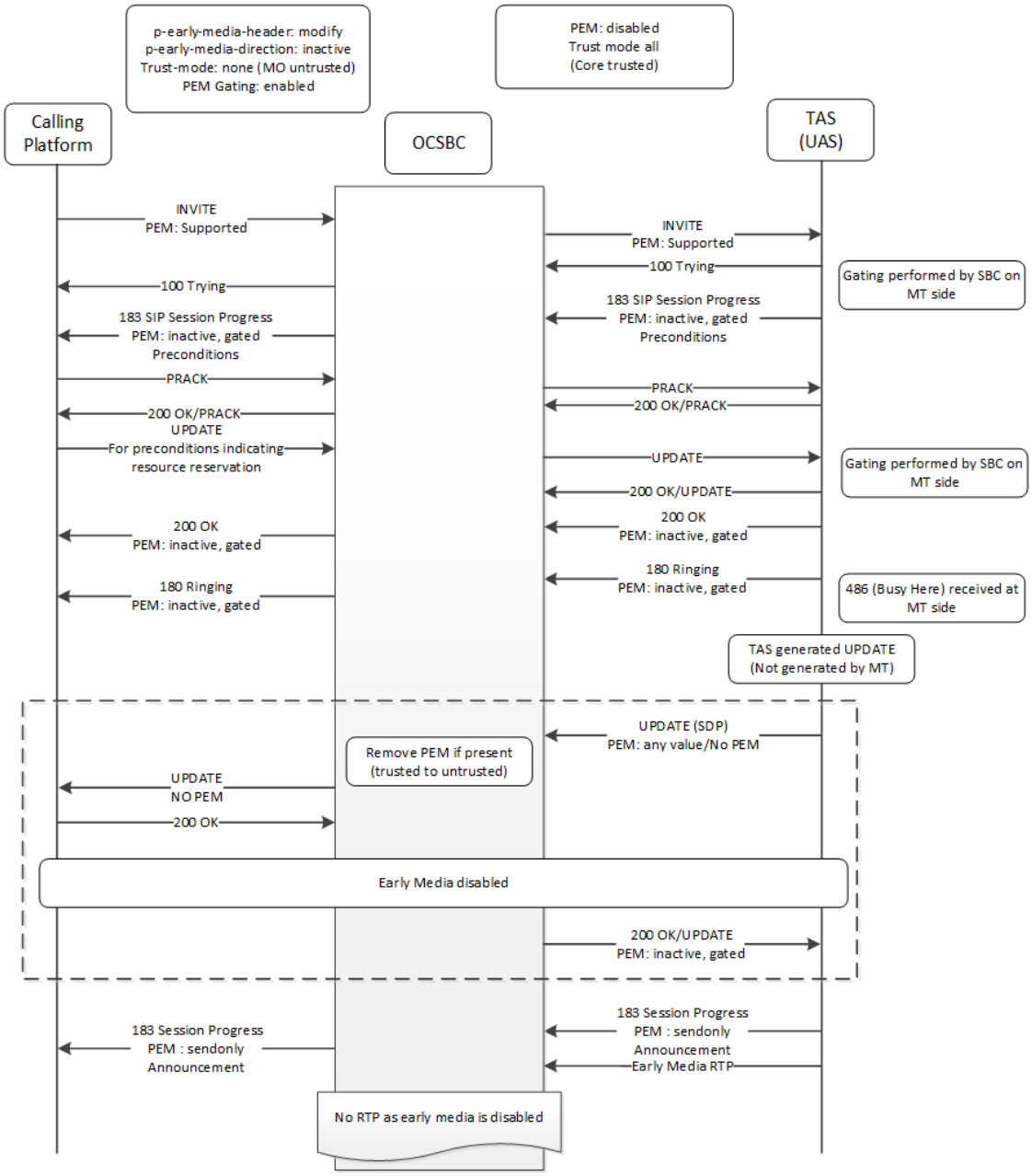
Bypassing Early Media Gating

You can configure the SBC to bypass gating and forward early media to untrusted domains. This feature resolves early media problems for situations including PEM gating when an UPDATE goes from the trusted side towards the untrusted side and the system prevents an early media announcement to play through the subsequent 18x. In this case, the default SBC behavior would be to gate the early media. To configure this feature, you enable the **pass-pem-in-update** option on the ingress **sip-interface**.

The SBC supports the SIP P-Early-Media (PEM) header and function used within SIP INVITE, PRACK, and UPDATE requests and responses to authorize the use of early media. Standard and default SBC behavior includes preventing early media signaling from entering networks that are not authorized or are not trusted. In some deployments, however, administrators may want early media forwarded even if the network is not trusted.

The SBC uses two key **sip-interface** parameters to control early media, including **p-early-media-header** and **p-early-media-direction**. Both of these parameters are effectively disabled by default, which gates early media from operating across the interface. In addition, you can set the **trust-mode** according to your setup (none, all, agents and so on), which establishes multiple security behaviors on the interface, including gating early media from a trusted source.

The call flow below presents an example of the issues this feature resolves. In this flow, the SBC closes the gate on early media when it receives the 200 OK to the UPDATE from the mobile originating (MO) side, depicted within the dashed box. This is default SBC behavior, resulting in PEM being inactive on the access side, and, in this call flow, prevents the presentation of early media on the MO leg.



An example of a problem case solved by this features includes the SBC receiving an UPDATE with a PEM header arriving from the trusted mobile terminating (MT) leg/core side. Without this feature, the SBC follows its default behavior, which includes not forwarding the PEM header towards the untrusted access side:

- If you set the **p-early-media-header** access **sip-interface** to **modify** and the **p-early-media-direction** is inactive, the 200 OK response to the update from the MO effectively closes the gate.

- The SBC closes the gate when receiving the 200 OK to the UPDATE from the MO side, which is by design (PEM inactive on access), thereby gating early media on the MO leg. In this case, the access is always untrusted, and the core is always trusted.

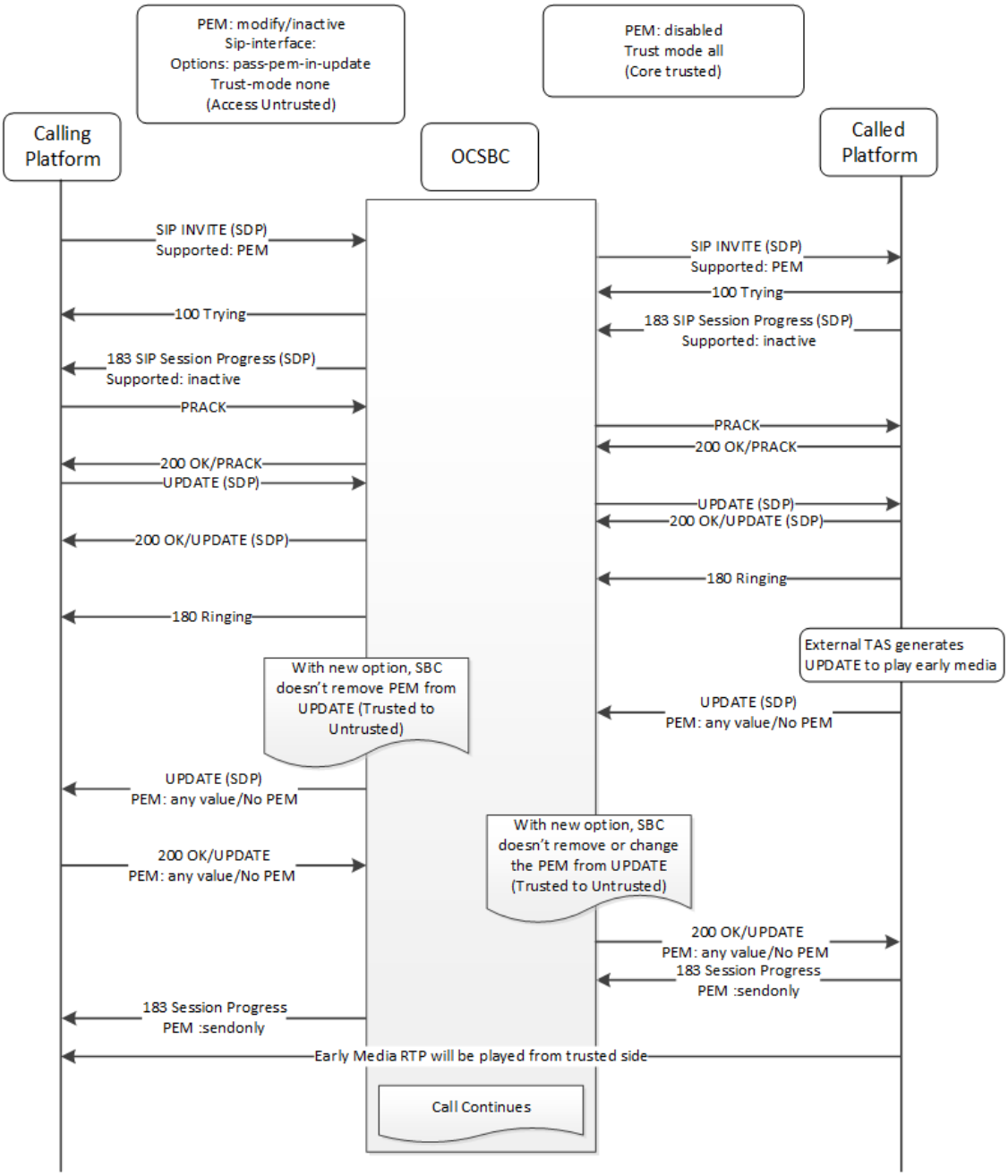
When you configure the **pass-pem-in-update** option, the SBC forwards the PEM header as received, ignoring other PEM header configuration. The applicable messages that can trigger this feature include the SBC:

- Receiving an UPDATE request from the trusted side and forwarding it towards the untrusted side.
- Receiving a 200 OK/UPDATE from the untrusted side and forwarding it towards the trusted side.

```
ORACLE(sip-interface)# +options pass-pem-in-update
```

The image below depicts the SBC supporting PEM across trusted and untrusted networks in both directions. You have configured the **trust-mode** to **none**, the **p-early-media-header** to **modify**, and the **p-early-media-header** to **inactive** on the access **sip-interface**. You have also set the **p-early-media-header** to **disabled** and the **trust-mode** to **all** on the egress **sip-interface**. Both of these configurations prevent early media across those interfaces.

But you have also enabled the **pass-pem-in-update** option on the ingress **sip-interface**, which would be applied to the UPDATE coming from the core (access interface egress), based on the egress **sip-interface** configuration. This configuration supersedes the other configurations, allowing the SBC to accept early media, effectively bypassing what would be the standard SBC behavior of rejecting it.

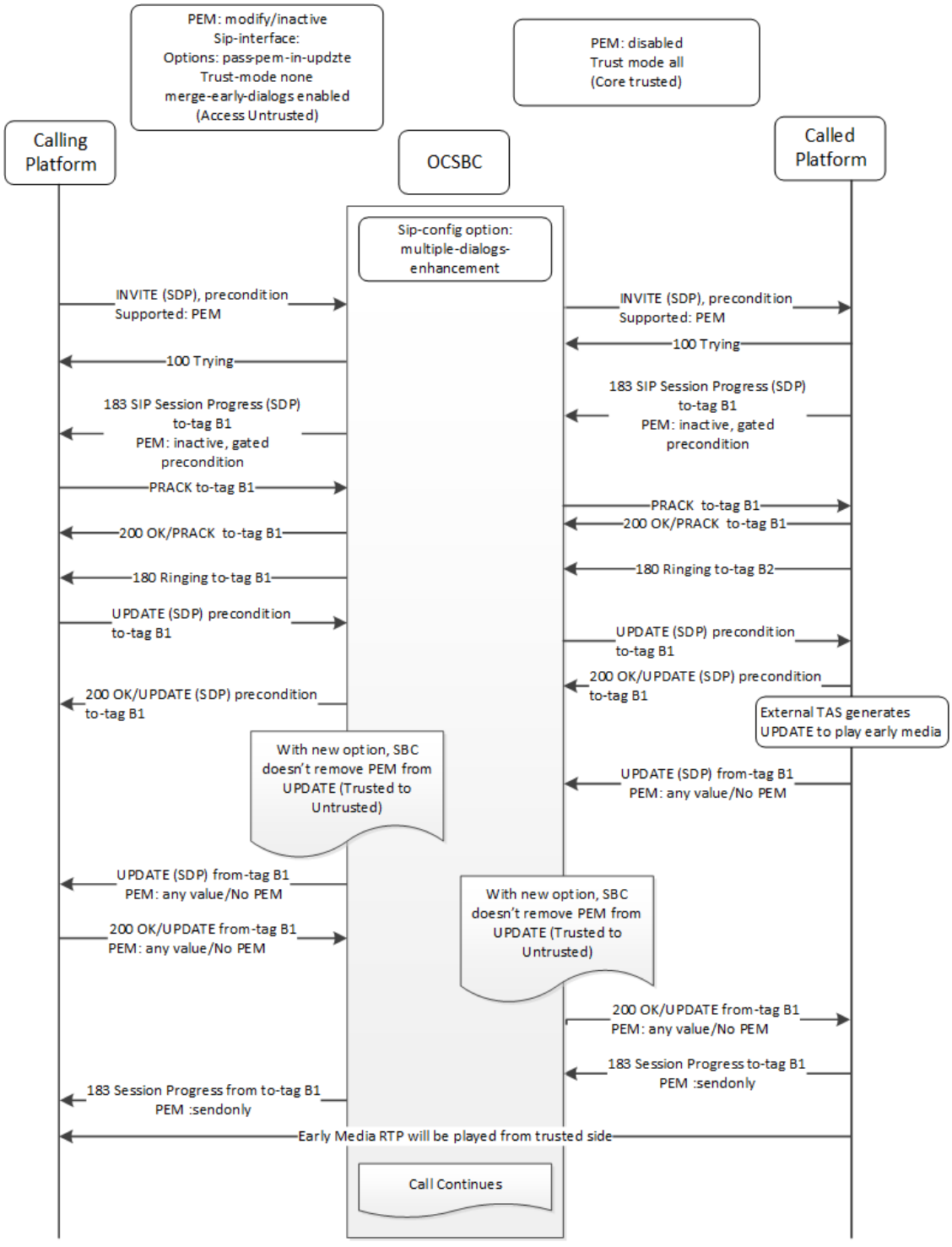


This feature is compatible with the multiple early dialogs feature within the context of dialog merging, and many-to-many dialogs. You establish these configurations with the **merge-early-dialogs** configuration and the **multiple-dialogs-enhancement** option respectively. Applicable cases wherein the SBC receives an UPDATE from the trusted side allows early media include:

- multiple-early-dialogs in many-to-many mode cases:
  - UN-TRUSTED (ingress side) to TRUSTED (egress side)
  - UN-TRUSTED (egress side) to TRUSTED (ingress side)
- multiple-early-dialogs in merge mode cases, with the ingress configured as the merge realm:
  - UN-TRUSTED (ingress side) to TRUSTED (egress side)
  - TRUSTED (ingress side) to UN-TRUSTED (egress side)

Note:

This is supported in E2E precondition cases because the SBC responds to an UPDATE message from the calling party locally in the context of the merge.



P-Early-Media ACLI Configuration

Use the following procedure to configure P-Early-Media SIP header support.

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

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

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. Use the **p-early-media-header** parameter to enable P-Early-Media SIP header support. This parameter is disabled by default.
   - **disabled**—(the default value) disables support
   - **add**—enables support and allows the Oracle Communications Session Border Controller to add the P-Early-Media header to SIP messages.
   - **modify**—enables support and allows the Oracle Communications Session Border Controller to modify or strip the P-Early-Media header in SIP messages.
   - **support**—adds additional PEM support, including enforcing PEM from trusted sources, preventing system modification of PEM direction, not adding PEM if absent from SIP replies, and adding PEM if it is not advertised in the initial INVITE.

```
ACMEPACKET(sip-interface)# p-early-media-header add
ACMEPACKET(sip-interface)#
```

4. Use the **p-early-media-direction** parameter to specify the supported directionalities.
   - **sendrecv**—send and accept early media
   - **sendonly**—send early media
   - **recvonly**—receive early media
   - **inactive**—reject/cancel early media

```
ACMEPACKET(sip-interface)# p-early-media-direction sendrecv,sendrecv
ACMEPACKET(sip-interface)#
```

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

SIP Early Media Suppression

This section explains how to configure SIP early media suppression, which lets you determine who can send early media and in what direction. Early media are the RTP/RTCP packets sent from the called party to the caller, or vice versa, before a session is fully established (before a

200 OK is received). When the Oracle Communications Session Border Controller receives an INVITE message with SDP, it can forward media packets to the calling endpoint as soon as it forwards the INVITE to the next hop. It can also forward media packets received from the calling endpoint to the called endpoint as soon as the Oracle Communications Session Border Controller receives SDP in a SIP response to the INVITE, usually a provisional message. This allows for any early media to be played, such as remote ringback or announcement.

Early media can be unidirectional or bidirectional, and can be generated by the caller, the callee, or both.

With early media suppression, you can block early media until the call is established. You can define which outbound realms or next hop session agents are allowed to send or receive early media. early media suppression only applies to RTP packets. RTCP packets received by Oracle Communications Session Border Controller are still forwarded to their destination in both directions, unless an endpoint is behind a NAT and the media manager has not been enabled for RTCP forwarding.

> **Note:**
>
> To use early media suppression, you cannot configure media release of any kind: same-realm, same-network, or multiple-system media release.

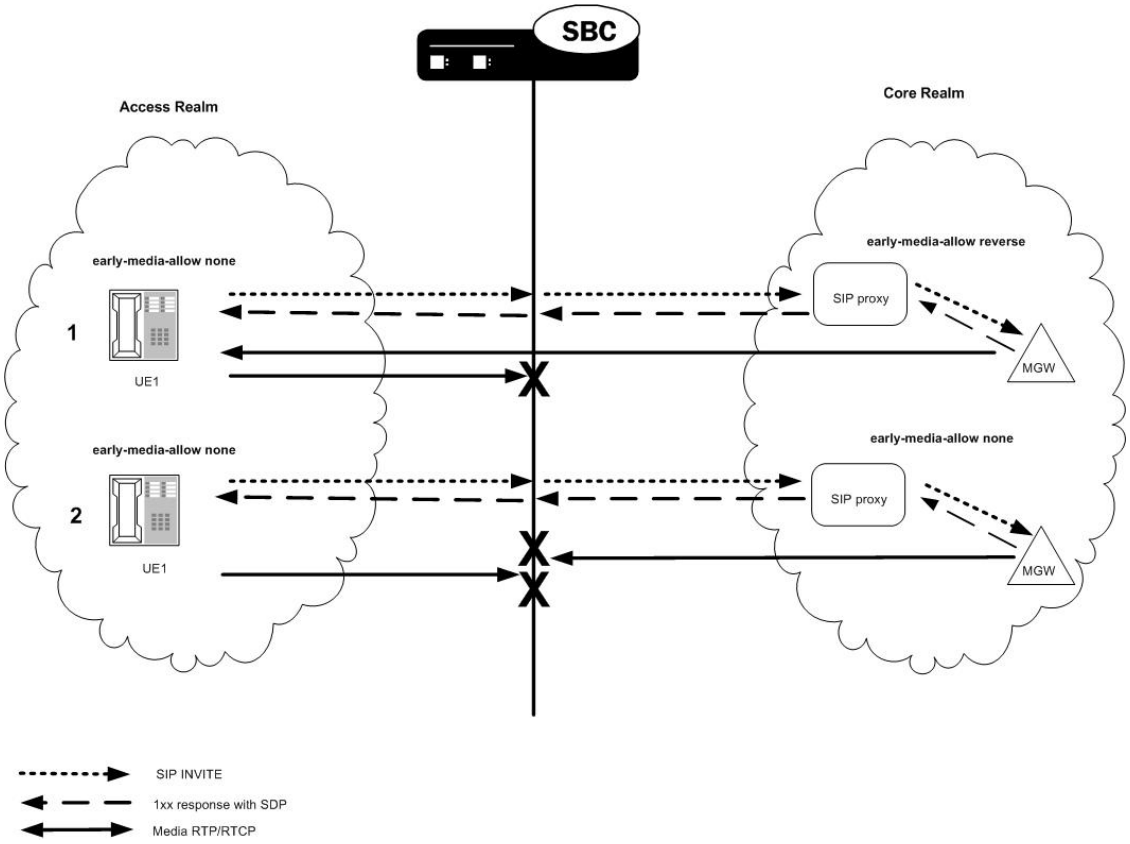
With the SIP-based addressing, early media suppression is based on the outbound SIP interface realms and the value of their early-media-allow parameter. When the Oracle Communications Session Border Controller forwards a SIP Invite out a SIP interface, the outbound realm is chosen based on the SIP layer information, such as the session agent for the next-hop or the address prefix of the next-hop SIP device. The matching realm's early-media-allow parameter value then applies to either allow all, block all, or block one-way early media until a 200 ok is received. At that point bidirectional media is allowed. The decision is based on SIP-layer addressing of next-hops.

You configure a rule for a realm or a session agent to use early media suppression. An early media suppression rule specifies whether you want to prevent early media in any direction, allow early media going to the calling endpoint in the reverse direction, or allow early media in both directions. The forward direction is when the packets flow from the caller to the called party. The reverse direction is when the packets flow from the called party to the caller.

The early media suppression rule is applied to a session. When the Oracle Communications Session Border Controller initiates a new session, it first checks whether the next hop is a session agent and if so, whether an early media suppression rule has been configured it. If an early media suppression rule is found, the Oracle Communications Session Border Controller enforces it. If the next hop is not a session agent or no early media suppression rule is configured, the Oracle Communications Session Border Controller checks whether an early media suppression rule has been configured for the outbound realm. If it finds one, it enforces it.

Example

The following illustration shows two examples of early media suppression.



1. Caller UE1 makes a call to the PSTN media gateway (MGW). The INVITE traverses from UE1 to the Oracle Communications Session Border Controller through the softswitch to the MGW. The Oracle Communications Session Border Controller allows early media from the core to reach UE1.
2. The PSTN MGW makes a call to UE1. The INVITE traverses to the Oracle Communications Session Border Controller and to UE1. The Oracle Communications Session Border Controller blocks all early media to and from UE1 until a 200 OK is received.

Early Media Suppression Support

The Oracle Communications Session Border Controller supports suppressing early media in the following directions no matter which side makes the SDP offer, until it receives 200 OK for an INVITE:

- Forward direction based on the outbound realm or next-hop session agent
- Forward and reverse directions based on the outbound realm or next-hop session agent.

The Oracle Communications Session Border Controller allows all media when a 200 OK response is received for the INVITE, regardless of whether the 200 OK response contains SDP.

Call Signaling

The Oracle Communications Session Border Controller media manager performs early media suppression according to an early media suppression rule. No change has been made to call signaling. For SIP, the Oracle Communications Session Border Controller still forwards SDP

received in an INVITE request or response after performing a NAT to the media connection address. After which, the Oracle Communications Session Border Controller is ready to receive media packets from the endpoints. If an early media suppression rule has been configured, the Oracle Communications Session Border Controller drops the packets going in the direction being specified by the rule.

For a H.323 to SIP call, early media suppression rule does not change how the Oracle Communications Session Border Controller performs H.225/Q.931 call signaling and starts the H.245 procedure (if required) to establish logical channels for early media on the H.323 leg of the call.

Suppression Duration

When early media suppression is enabled in a session, the block lasts until the session is established. For a SIP to SIP call or an H.323 to SIP call, a session is established when the system receives a 200 OK response to the INVITE. A 200 OK response to the INVITE terminates early media suppression, even when it does not contain a SDP. (A 200 OK response to a PRACK or an UPDATE request does not terminate early media suppression.) After a session is established, the Oracle Communications Session Border Controller can receive a change in media session (for example, a re-INVITE with a new SDP) without an early media suppression rule blocking the media.

About the Early Media Suppression Rule

An early media suppression rule is configured in the form of a permission. It specifies whether early media is allowed in both directions, the reverse direction only or not at all. Reverse direction media is media sent in the upstream direction towards the calling endpoint.

Session Agent Rule

The next-hop session agent's early media suppression rule is applied regardless of whether the media packet's source or destination address is the same as the session agent's address. For example, if the session's next hop session agent is 10.10.10.5 but the SDP in a 183 response specifies 10.10.10.6 as its connection address.

Rule Resolution

When the call's next hop is a session agent and both the outbound realm of the call and the session agent have an early media suppression rule, the session agent's early media suppression rule takes precedence. If the session agent's early media suppression rule has not been configured, the outbound realm's early media suppression rule is used, if configured.

Selective Early Media Suppression

Normally, the Oracle Communications Session Border Controller performs early media blocking based on destination realm. Calls to such realms are prohibited from sending and receiving RTP until a SIP 200 OK response is received, and you can set the direction of the blocked media.

While decisions to block early media are customarily based on SIP-layer addressing, there are cases when the Oracle Communications Session Border Controller can reject early media based on the SDP address in the SDP answer for a 1XX or 2XX response. By comparing the SDP address with the realm prefix or additional prefix address, it can block early media for matching realms. For these cases, you define global or signaling realms—ones that are not tied to SIP interfaces, but which establish additional address prefixes and rules for blocking early media.

This way, the Oracle Communications Session Border Controller blocks all early media for SIP interface realms, but can accept it for global realms that reference media or PSTN gateways. This configuration allows early media for calls destined for the PSTN, and blocks it for user-to-user and PSTN-to-user calls.

Selective early media suppression addresses the fact that some service providers need to allow early media for certain user-to-user and PSTN-to-user calls to support, for example, custom ringback tones. The enhancements also address the fact that Oracle Communications Session Border Controllers can themselves lose the ability to decide whether or not early media should be blocked when confronted with hairpinned call flows, or with traffic that traverses multiple Oracle Communications Session Border Controllers.

To address this need, you can configure realm groups. Realm groups are sets of source and destination realms that allow early media to flow in the direction you configure. For example, you can set up realm groups to allow media from PSTN realms to user realms so that users can listen to PSTN announcements, but prohibit early media from user realms to PSTN realms.

> **Note:**
>
> The source and destination realms you add to your lists need to be a global signaling realm matching the caller's SDP address prefix or a SIP realm.

Configuring the Realm

To configure the realm:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ```

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

   ```
   ORACLE(media-manager)# realm
   ORACLE(realm)#
   ```

4. If configuring an existing realm, enter the select command to select the realm.
5. **early-media-allow**—Enter the early media suppression rule for the realm. The valid values are:
   - **none**—No early media is allowed in either direction
   - **both**—Early media is allowed in both directions
   - **reverse**—Early media received by Oracle Communications Session Border Controller in the reverse direction is allowed

     There is no default value. If you leave this parameter blank, early media is allowed in either direction. You can use the following command to clear this parameter:

     ```
     early-media-allow ()
     ```

6. Save and activate your configuration.

   For example:

```
realm-config
        identifier                      access1
        addr-prefix                     192.168.1.0/24
        network-interfaces
                                        media:0
        mm-in-realm                     enabled
        mm-in-network                   enabled
        msm-release                     disabled
        qos-enable                      disabled
        max-bandwidth                   0
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        in-translationid
        out-translationid
        class-profile
        average-rate-limit              0
        access-control-trust-level      none
        invalid-signal-threshold        0
        maximum-signal-threshold        0
        deny-period                     30
        early-media-allow               none
        last-modified-date              2006-02-06 13:09:20
```

Configuring Session Agents

If you do not configure early media suppression for a session agent, the early media suppression for the outbound realm is used, if configured.

To configure session agents:

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

```
ORACLE# configure terminal
```

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

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

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

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

4. If configuring an existing session agent, enter the select command to select the session agent.

5. **early-media-allow**—Enter the early media suppression rule for the session agent. The valid values are:

   - **none**—No early media is allowed in either direction
   - **both**—Early media is allowed in both directions
   - **reverse**—Early media received by Oracle Communications Session Border Controller in the reverse direction is allowed

     There is no default value. If you leave this parameter blank, early media is allowed in either direction. You can use the following command to clear this parameter:

     ```
     early-media-allow ()
     ```

6. Save and activate your configuration.

   For example:

```
session-agent
        hostname                        cust1
        ip-address                      192.168.1.24
        port                            5060
        state                           enabled
        app-protocol                    SIP
        app-type
        transport-method                UDP
        realm-id                        access1
        description
        carriers
        constraints                     disabled
        max-sessions                    0
        max-outbound-sessions           0
        max-burst-rate                  0
        max-sustain-rate                0
        time-to-resume                  0
        ttr-no-response                 0
        in-service-period               0
        burst-rate-window               0
        sustain-rate-window             0
        req-uri-carrier-mode            None
        proxy-mode
        redirect-action
        loose-routing                   enabled
        send-media-session              enabled
        response-map
        ping-method
        ping-interval                   0
        media-profiles
        in-translationid
        out-translationid
        trust-me                        disabled
        early-media-allow               reverse
        last-modified-date              2006-05-06 13:26:34
```

Configuring Realm Groups

To configure a realm group for selective early media suppression:

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

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

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-group** and press Enter.

```
ORACLE(media-manager)# realm-group
ORACLE(realm-group)#
```

4. **name**—Enter the name of the realm group.
5. **source-realm**—Enter the list of one or more global/SIP realms that you want to designate as source realms for the purpose of blocking early media; this is the realm identifier value for the realms you want on the list. Values in this list refer to calling SDP realms; this parameter has no default. To enter more than one realm in the list, list all items separated by a comma and enclose the entire entry in quotation marks:

```
ORACLE(realm-group)# source-realm Private, Public
```

   To add a realm to the list, use the plus sign (+) in front of each new entry.

```
ORACLE(realm-group)# source-realm +Private
```

   You can also remove single items in the list by using the minus sign (-) directly in front of the realm identifier.

```
ORACLE(realm-group)# source-realm -Private
```

6. **destination-realm**—Enter the list of one or more global/SIP realms that you want to designate as destination realms for the purpose of blocking early media; this is the realm identifier value for the realms you want on the list. Values in this list refer to called SDP realms; this parameter has no default. To enter more than one realm in the list, list all items separated by a comma and enclose the entire entry in quotation marks:
7. ORACLE(realm-group)# **source-realm Private, Public**

   To add a realm to the list, use the plus sign (+) in front of each new entry.

```
ORACLE(realm-group)# destination-realm +Private
```

   You can also remove single items in the list by using the minus sign (-) directly in front of the realm identifier.

```
ORACLE(realm-group)# destination-realm -Private
```

8. **early-media-allow-direction**—Set the direction for which early media is allowed for this realm group. Valid values are:
   - none—Turns off the feature for this realm group by blocking early media

- reverse—Allows early media to flow from called to caller
- both (default)—Allows early media to flow to/from called and caller

9. Save and activate your configuration.

SDP-Response Early Media Suppression

This section explains how to configure SDP-response early media suppression, which can be used when the Oracle Communications Session Border Controller is deployed after a softswitch or proxy in the signaling path. In this deployment, user endpoints and gateways communicate directly with the softswitch or proxy, which in turn sends call signaling to the Oracle Communications Session Border Controller. The call signaling gets sent back to the same or different softswitch or proxy. Because the Oracle Communications Session Border Controller does not communicate with the endpoints or gateways that are the media terminators, early media suppression for this deployment must use SDP-based addressing rather than the SIP-based addressing (described in the SIP Early Media Suppression section in this technical notice).

Using this feature lets you configure specific IP addresses for which early media should not be suppressed, based on SDP addressing. The Oracle Communications Session Border Controller checks the SDP addresses in SIP responses against these IP address or address ranges to determine on which media gateway a call terminates.

SIP-Based Addressing

With the SIP-based addressing described in the SIP Early Media Suppression section, early media suppression is based on the outbound SIP interface realms and the value of their early-media-allow parameter. When the Oracle Communications Session Border Controller forwards a SIP Invite out a SIP interface, the outbound realm is chosen based on the SIP layer information, such as the session agent for the next-hop or the address prefix of the next-hop SIP device. The matching realm's early-media-allow parameter value then applies to either allow all, block all, or block one-way early media until a 200 ok is received. At that point bidirectional media is allowed. The decision is based on SIP-layer addressing of next-hops.

SDP-Based Addressing

SDP-response early media suppression follows the same sequence described for SIP-based addressing with one exception. A provisional response with SDP media can make the Oracle Communications Session Border Controller select a new early-media-allow rule from another realm, based on the addressing inside the responding SDP.

When the SDP-response early media suppression feature is enabled, the Oracle Communications Session Border Controller searches the outbound SIP interface's realms for a matching address prefix with the connection address in the responding SDP. If it finds a match, it uses the early-media-allow parameter value of that realm until the 200 OK message is received, then bidirectional media is allowed regardless. If the Oracle Communications Session Border Controller does not find a match, it searches all of the global realms for one. If it finds a match, the Oracle Communications Session Border Controller uses that realm's early-media-allow parameter value. If it does not find a match in the global realm(s), the Oracle Communications Session Border Controller continues to use the previous early-media-allow parameter value.

Global Realms

Global realms are realms that are not parents or children of any other realms, do not have defined SIP interfaces and ports (or any signaling interface or stack), and are configured to use

the network interface lo0:0. They are special realms, applicable system-wide, and are currently only used for this feature. The only global realm configuration parameters applicable to early media suppression are:

- addr-prefix
- additional-prefixes
- early-media-allow
- network-interface (which must be set to lo0:0)

Additional Prefixes

You can specific additional prefixes in addition to that of the addr-prefix parameter you configure for a realm. For example, you can configure a global realm with additional address prefixes to specify the IP addresses (or ranges of addresses) of the media gateways that are allowed to send and receive early media. This overrides the SIP interface realm's early media blocking settings.

You can also enter additional prefixes in non-global realms. These additional prefixes function the same as would multiple values in the addr-prefix parameter (which only takes one value), except addresses in additional-prefixes are not used for SIP NATs.

Using the SDP-Response Early Media Suppression Rule

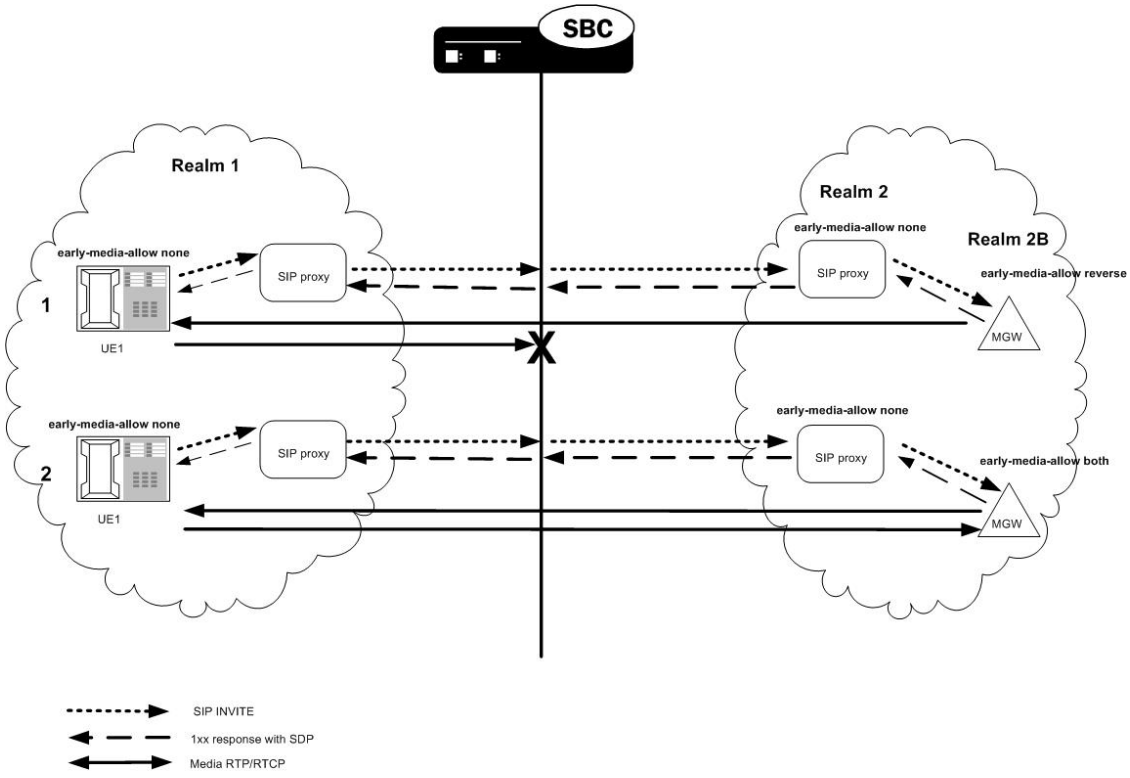
To use SDP-response early media suppression, you must add the early-media-sdp-realms option to the SIP interface configuration that interfaces with the next-hop device, such as the supported softswitch.

When the Oracle Communications Session Border Controller receives a provisional response that includes SDP from the called endpoint, and the early-media-sdp-realms option is active in the outgoing SIP interface of the call, it first searches the realms that apply to the outgoing SIP interface. If it does not find a realm, the Oracle Communications Session Border Controller searches the global realms. If the search yields a new realm that is not the SIP interface realm, its early media suppression rule (if any) replaces the existing one. Only the early media suppression rule of the new realm is applied to the call. Other realm properties from the outbound realm remain applicable to the call. If no new realm is found, the early media policy of the outgoing SIP interface realm is applied.

TheOracle Communications Session Border Controller allows media when the SDP media connect address in a response matches one of a configured list of IP address ranges defined in a realm and the realm has early media allowed. You need to configure specific a IP address or address range to specify which media gateways should not be suppressed based on SDP media addresses. The IP addresses are checked against the SDP being received. The decision for suppression is based on whether the matching realm allows early media. The early media will be suppressed if the matching realm does not allow early media or if there is no match and the outbound SIP interface ream does not allow early media.

Example

The following illustration shows two examples of SDP-response early media suppression.



Configuring SDP-Response Early Media Suppression

To configure SDP-response early media suppression:

1. Add the early-media-sdp-realms option to the SIP interface that interfaces with the softswitch.
2. Configure the SIP interface realm with an early media suppression rule that blocks all early media.
3. Configure either or both of the following:
   - One or more of the SIP realm's child realms, each with an early media suppression rule that allows all or reverse direction early media and a list of additional prefixes that specifies the IP addresses of the media gateways, or a range of IP addresses that includes the media gateways. Early media is allowed from these gateways only for calls that signals through this SIP interface.
   - One or more realms that has the network interface equal to lo0:0, an early media suppression rule that allows all or reverse direction early media and a list of additional prefixes that specifies the IP addresses of the media gateways, or a range of IP addresses that includes the media gateways. Early media is allowed from these gateways regardless of interface.

Configuring the SIP Interface

To configure a SIP interface:

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

```
ORACLE# configure terminal
```

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

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

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

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

From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. If configuring an existing interface, enter the select command to select the interface.
5. **options**—Enter early-media-sdp-realms as the option. If adding to an existing list of options, use a preceding plus (+) sign.

```
options +early-media-sdp-realms
```

6. Continue to the next section to configure the outbound realm.

For example:

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# options + early-media-sdp-realms
ORACLE(sip-interface)# done
sip-interface
        state                           enabled
        realm-id                        access1
        sip-port
                address                         192.168.1.30
                port                            5060
                transport-protocol              UDP
                allow-anonymous                 all
        carriers
        proxy-mode                      Proxy
        redirect-action
        contact-mode                    maddr
        nat-traversal                   none
        nat-interval                    30
        registration-caching            disabled
        min-reg-expire                  300
        registration-interval           3600
        route-to-registrar              disabled
        teluri-scheme                   disabled
        uri-fqdn-domain
        options                         early-media-sdp-realms
        trust-mode                      all
        last-modified-date              2006-05-10 18:27:31
```

Configuring a Realm

To configure a realm:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ```

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

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

4. If configuring an existing realm, enter the select command to select the realm.
5. **early-media-allow**—Enter the early media suppression rule for the realm. The valid values are:
   - **both**—Early media is allowed in both directions
   - **reverse**—Early media received by Oracle Communications Session Border Controller in the reverse direction is allowed
   - **none**—Early media is blocked
6. **additional-prefixes**—Enter a single or a comma-delimited list of IP address prefixes to use in addition to the value of the **addr-prefix** parameter.

   ```
   <IPv4> [/<number of bits>]
   ```

   <IPv4> is a valid IPv4 address and <number of bits> is the number of bits to use to match an IP address with the address prefix. Not specifying <number of bits> implies that all 32 bits are used for matching.

   Enclose the list between quotes if there is any space between a comma and the next address prefix.

   You can add and remove address prefixes to and from the list:

   - add-additional-prefixes adds one or more additional prefixes

     ```
     add-additional-prefixes 192.168.201.69
     ```

   - remove-additional-prefixes removes one or more additional prefixes

     ```
     remove-additional-prefixes 192.168.201.69
     ```

     If using multiple address prefixes, enter a comma-delimited list.
7. Save and activate your configuration.

   For example:

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)# additional-prefixes 192.168.200.0/24,192.168.201.68
   ORACLE(realm-config)# done
   ```

```
realm-config
        identifier                      early-media
        addr-prefix                     0.0.0.0
        network-interfaces
                                        media2:0
        mm-in-realm                     disabled
        mm-in-network                   enabled
        msm-release                     disabled
        qos-enable                      disabled
        max-bandwidth                   0
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        in-translationid
        out-translationid
        in-manipulationid
        out-manipulationid
        class-profile
        average-rate-limit              0
        access-control-trust-level      none
        invalid-signal-threshold        0
        maximum-signal-threshold        0
        untrusted-signal-threshold      0
        deny-period                     30
        symmetric-latching              disabled
        pai-strip                       disabled
        trunk-context
        early-media-allow               reverse
        additional-prefixes             192.168.200.0/24
192.168.201.69
        last-modified-date              2006-05-11 06:47:31
```

Early Media Support for Multiple Early Dialog Scenarios

By default, the SBC supports the P-Early-Media (PEM) header for authorizing early media, and multiple early dialogs with PEM headers. With additional configuration, it can expand on early media support functions related to multiple early dialogs and PEM header management that target specific environments or deployments, such as SIP/SIPI interworking and access VoLTE deployments.

Specific configurable early media support functions include:

- Hiding the multiple early dialogs by merging them into a single dialog when deployed in a SIP/SIPI environment. With some legacy equipment (MGCs, PBXs), this becomes necessary because SIP-I does not support multiple early dialog
- Resolution and collation of many-to-many dialog scenarios
- Mapping a SIP UPDATE towards the right early dialog when deployed within an access VoLTE environment
- Appropriate media latching when presented with multiple latching choices

Configuration of these enhanced multiple early dialog support, including PEM header, dialog merging, and many-to-many dialog support includes:

- Set the **sip-config** option **multiple-dialogs-enhancement**. By default this option is not configured.
- In the **realm-config**, enable the **merge-early-dialogs** parameter on the caller side to merge early media dialogs.
- In addition, you can set the **p-early-media-header** configuration in **sip-interface** to **support** to trigger behavior that supports merged early dialogs.

All of this configuration supports real-time configuration, not requiring a reboot.

> **Note:**
>
> The SBC uses its interaction with the PCRF, via the SIP-Forking-Indication AVP, to reserve the highest QoS requested for that media component by any forked responses. This does not require configuration.

Related Feature Dependencies

The complexity of these enhancements and their related functions can conflict, supercede or depend on other aspects of your SBC configuration. Configuration dependencies include:

- The **realm-config** must have:
  - **hide-egress-media-update** enabled to maintain RTP consistency
  - A **codec-policy** set (xcoding enabled)
- The **sip-config** must have the **multiple-dialogs-enhancement** option enabled.
- If the **media-manager** has latching enabled, **sdp-restrict** is disabled.
- You must not set the **dont-save-early-dialog-sdp** option.

Recall that the SBC requires the **100-rel-interworking** option set on the ingress **sip-interface** to support PRACK interworking, which is needed for early dialog support.

> **Note:**
>
> Legacy SDP management (SCz7.2.0 and below) uses the last SDP seen as the source for SDP. You can enable this legacy behavior by setting the **dont-save-early-dialog-sdp** option in the **sip-config**. Do not use this unless directed by Oracle.

```
ORACLE(sip-config)# options +dont-save-early-dialog-sdp
```

Merge Function within Early Dialog Support

Dialog merge functionality allows the SBC to manage media, handle potential transcoding change requirements, and accommodate complex PRACK and UPDATE scenarios for merged (single caller, multiple callee) call flows. You configure this support in conjunction with other configuration depending on your deployment. This merge requires that any forking proxy send only one final response to the SBC, either a 200 OK or an error response. You configure the

merge-early-dialogs parameter on the caller realm to support the merging of multiple early dialogs into a single dialog.

The SBC aggregates all early dialogs on the calling party side using the same early dialog, identified via the presence of the same To-Tag identifier, consistent CSeq, RSeq, SDP session and so forth. If there are provisional responses containing SDP, the associated provisional responses towards the ingress leg contain the same unique SDP, as created by the SBC.

The final response sent by the SBC also belongs to the same dialog (same To-Tag identifier). If a former provisional response sent contained SDP, the final response contains the exact same SDP. If PRACK is activated for this dialog, however, the response would not contain SDP.

There is no need to drop messages coming from the calling party side, although the first early dialog received from the calling side determines the codec used for the call. The SBC also performs transcoding and PRACK interworking on a per call leg basis. Furthermore, the SBC does not increment the SDP version on the merged leg unless there is a renegotiation on that leg.

Operational behavior of this merging includes:

- The SBC maintains consistency of RTP parameters regardless of whether or not there is transcoding. These parameters include Seq Number, Timestamp, SSRC and so forth. This support includes on-the-fly transcoding setup as new early dialogs are received.
- There is only one dialog on the caller side even if there are multiple early dialogs on the called side.
- The SBC forwards every provisional message received to the caller with a single to-tag.
- The SBC sends an ACK for 200OK/INVITE to the correct dialog, which is the dialog with the to-tag in the 200OK/INVITE message.
- The SBC sends a BYE message to the correct dialog, which is the dialog with the to-tag in the 200OK/INVITE message.
- The SBC maintains the same SDP Version Number in all messages to the caller.

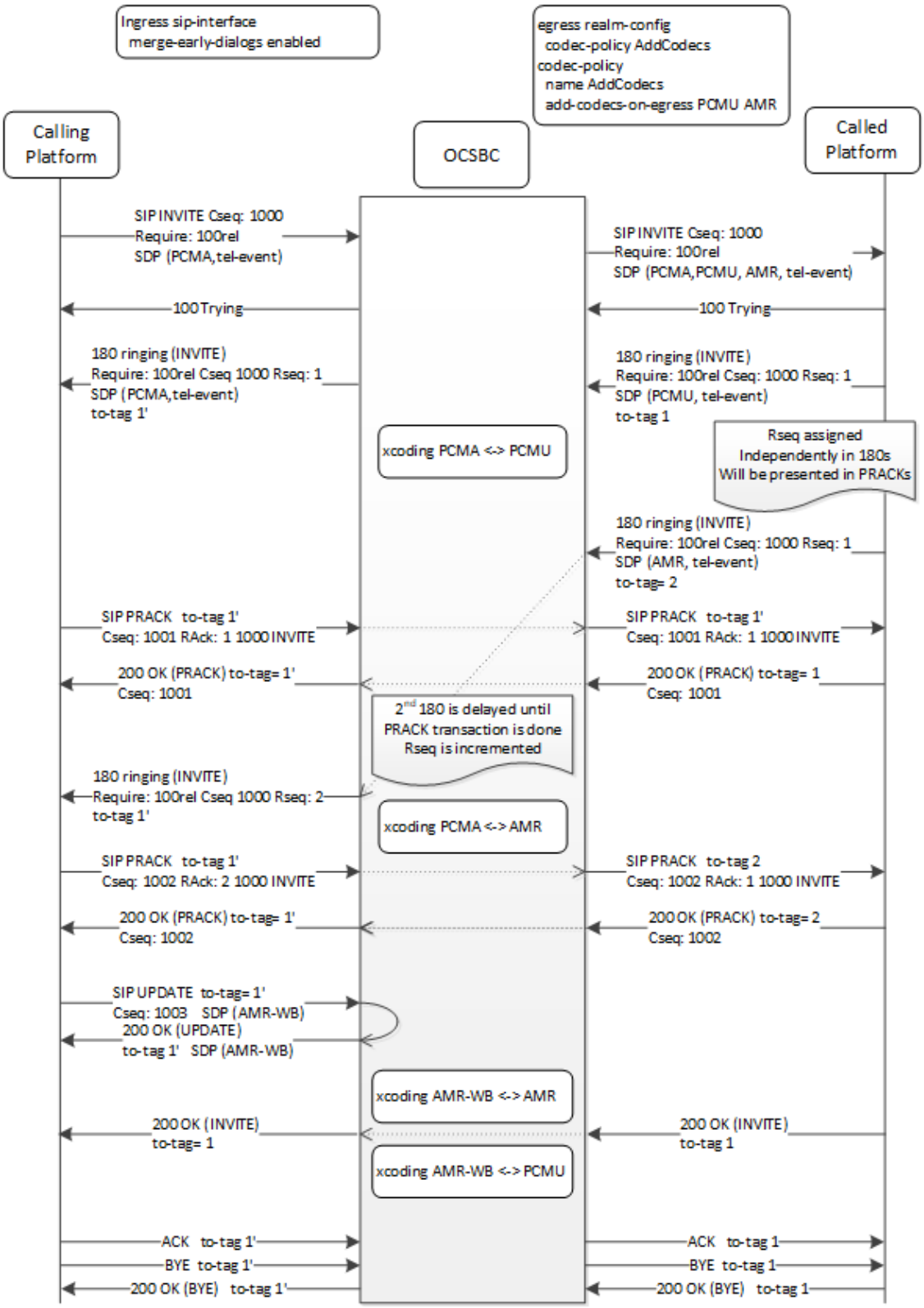
Consider the following configuration rules:

- You should configure HMU to maintain RTP consistency.
- When enabled, dialog-transparency disables this merging.
- You should configure transcoding resources and at least one codec policy on the ingress and/or egress side. This could be as simple as a codec policy configured to allow all.

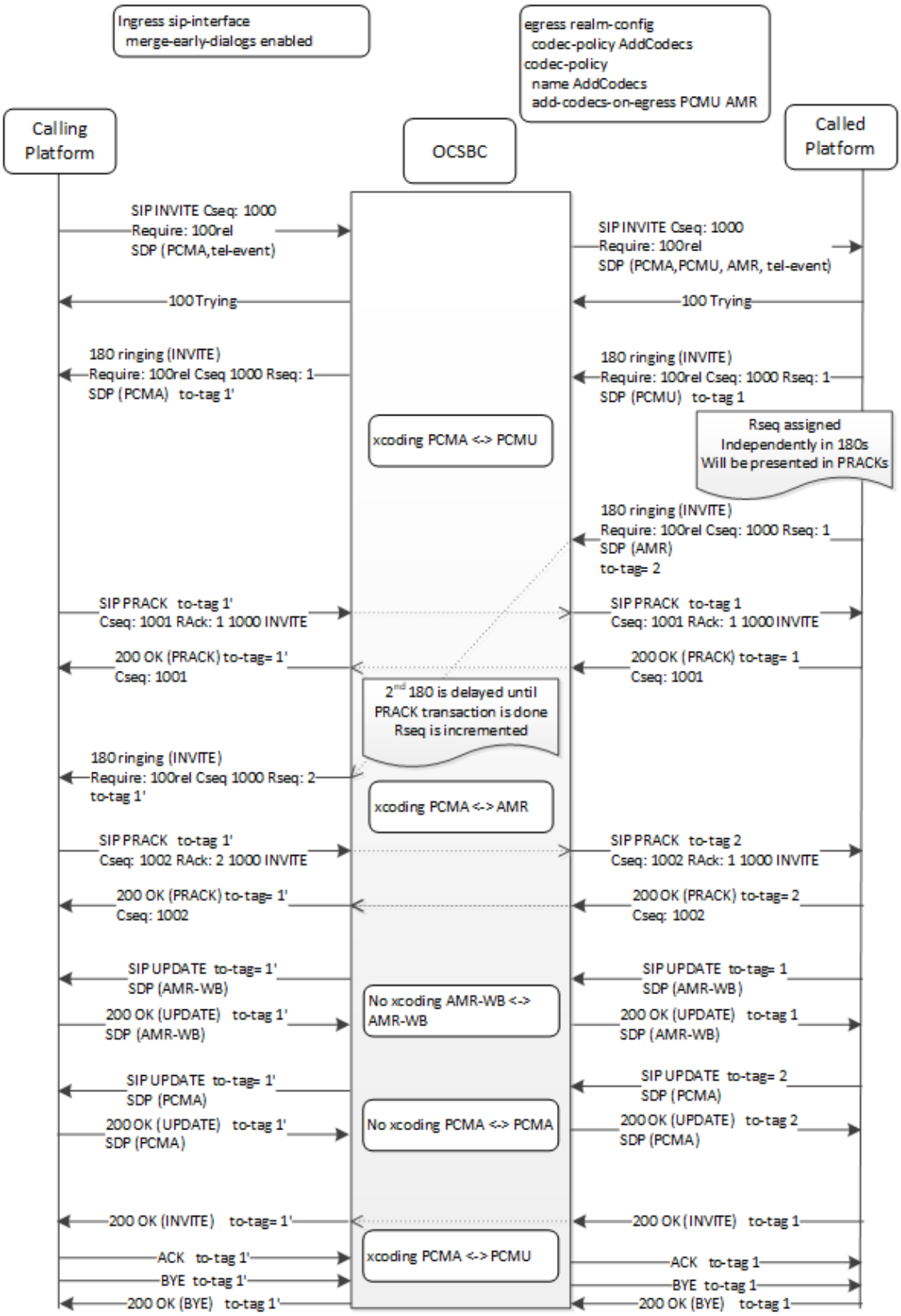
This merge function does not support:

- Offerless call
- Preconditions interworking
- SRVCC
- multiple audio or video m-line
- p-early-media-header with 'add' and 'modify' options

The call flow below presents an example of the SBC supporting a merge scenario with an UPDATE message received from the calling party. The SBC hairpins a 200OK in response to the UPDATEs locally towards the merged (single) dialog side.



The call flow below presents an example of the SBC supporting a merge scenario with an UPDATE message received from the called party. The SBC forwards the UPDATEs and associated 200OKs maintaining the same To-Tag for the calling platform.


Ingress sip-interface
merge-early-dialogs enabled
egress realm-config
codec-policy AddCodecs
codec-policy
name AddCodecs
add-codecs-on-egress PCMU AMR
Calling Platform
OCSBC
Called Platform
SIP INVITE Cseq: 1000
Require: 100rel
SDP (PCMA,tel-event)
SIP INVITE Cseq: 1000
Require: 100rel
SDP (PCMA,PCMU, AMR, tel-event)
100 Trying
100 Trying
180 ringing (INVITE)
Require: 100rel Cseq 1000 Rseq: 1
SDP (PCMA) to-tag 1'
180 ringing (INVITE)
Require: 100rel Cseq: 1000 Rseq: 1
SDP (PCMU) to-tag 1
xcoding PCMA <-> PCMU
Rseq assigned
Independently in 180s
Will be presented in PRACKs
180 ringing (INVITE)
Require: 100rel Cseq: 1000 Rseq: 1
SDP (AMR)
to-tag= 2
SIP PRACK to-tag 1'
Cseq: 1001 RAck: 1 1000 INVITE
SIP PRACK to-tag 1
Cseq: 1001 RAck: 1 1000 INVITE
200 OK (PRACK) to-tag= 1'
Cseq: 1001
200 OK (PRACK) to-tag= 1
Cseq: 1001
2nd 180 is delayed until
PRACK transaction is done
Rseq is incremented
180 ringing (INVITE)
Require: 100rel Cseq 1000 Rseq: 2
to-tag 1'
xcoding PCMA <-> AMR
SIP PRACK to-tag 1'
Cseq: 1002 RAck: 2 1000 INVITE
SIP PRACK to-tag 2
Cseq: 1002 RAck: 1 1000 INVITE
200 OK (PRACK) to-tag= 1'
Cseq: 1002
200 OK (PRACK) to-tag= 2
Cseq: 1002
SIP UPDATE to-tag= 1'
SDP (AMR-WB)
SIP UPDATE to-tag= 1
SDP (AMR-WB)
No xcoding AMR-WB <-> AMR-WB
200 OK (UPDATE) to-tag 1'
SDP (AMR-WB)
200 OK (UPDATE) to-tag 1
SDP (AMR-WB)
SIP UPDATE to-tag= 1'
SDP (PCMA)
SIP UPDATE to-tag= 2
SDP (PCMA)
200 OK (UPDATE) to-tag 1'
SDP (PCMA)
No xcoding PCMA <-> PCMA
200 OK (UPDATE) to-tag 2
SDP (PCMA)
200 OK (INVITE) to-tag= 1'
200 OK (INVITE) to-tag 1
ACK to-tag 1'
BYE to-tag 1'
200 OK (BYE) to-tag 1'
xcoding PCMA <-> PCMU
ACK to-tag 1
BYE to-tag 1
200 OK (BYE) to-tag 1

Many-to-Many Support within Early Dialog Support

The SBC provides early dialog support for many-to-many dialog scenarios, which are defined by working with multiple early dialogs established by and for both the callee and the caller. The SBC establishes a one-to-one correspondence between to-tag identifiers for the dialogs it sends and receives. This support applies to both reliable and unreliable dialogs (with or without PRACK). It also provides this support for both transcoding and non-transcoding call flows.

While processing responses within different dialogs, the SBC matches the dialogs of corresponding offers and answers and updates the flows to maintain SDP selection for the subsequent session. The ensuing behavior depends on whether the call is reliable, meaning it includes PRACKs, or unreliable:

- If the flow is reliable, the SBC does not restore the SDP for the final response in the same dialog, resulting in no SDP in the outgoing final response. If the final response, such as a 200 OK, arrives within a dialog that is different and doesn't include SDP, the SBC participates in changing the dialog switching and a media update.
- If the flow is unreliable and a final response arrives within the same dialog and without SDP, the SBC restores the SDP from the matching dialog, updates the media flows and then sends the final response with SDP inserted.

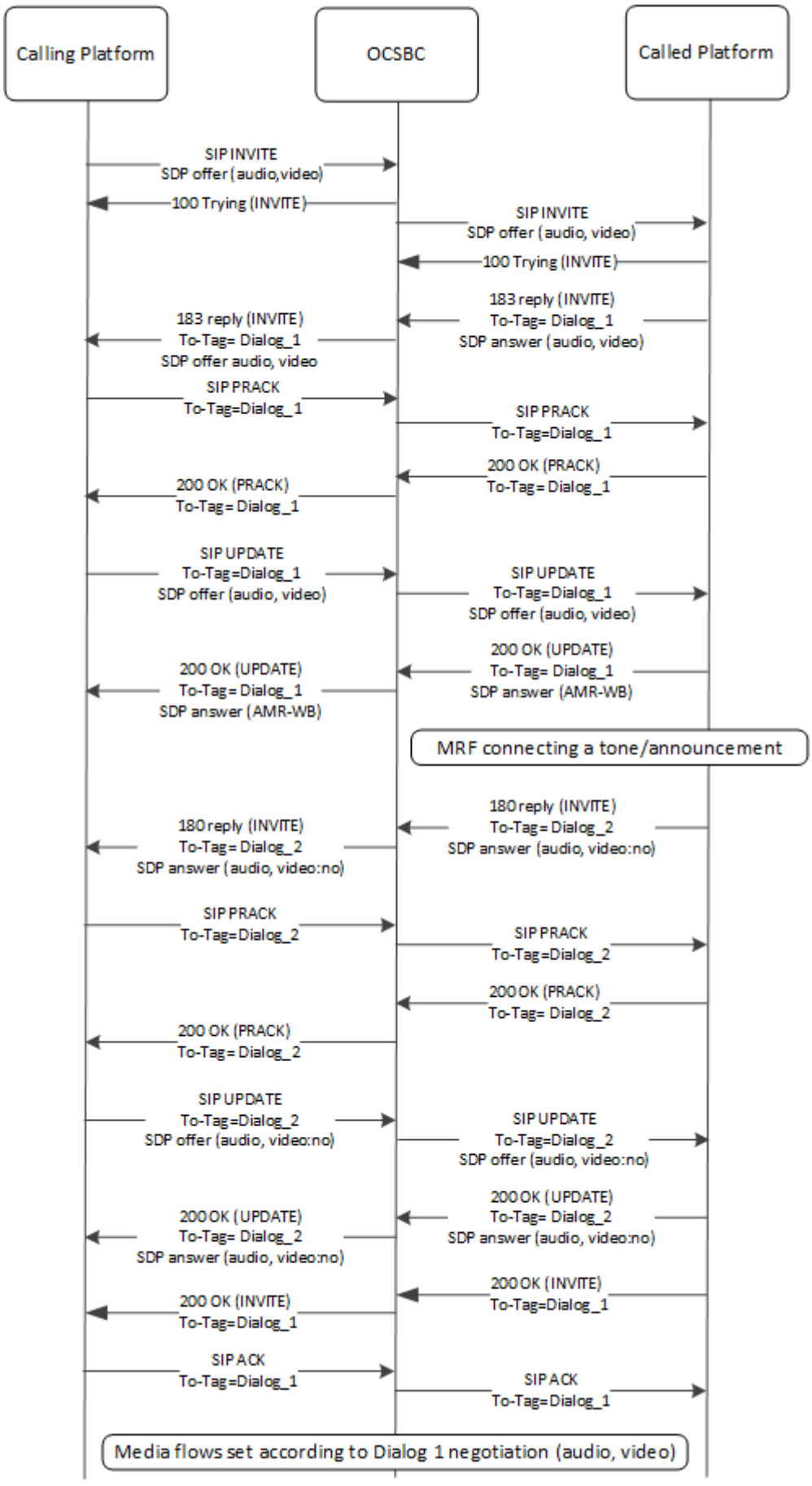
The SBC maintains the SDP version consistent across dialogs on both sides throughout the call.

Many-to-many cases require that you disable the **merge-early-dialogs** config. Also, the UAC must maintain same IP and port in the SDP c line across all dialogs.

This many-to-many configuration does not support the following:

- Offerless calls with **add-sdp-invite**
- Preconditions interworking
- PRACK IWF
- multiple audio or video m-line
- **p-early-media-header** with **support** option

The call flow below presents the SBC supporting a Multiple Early dialog scenario, wherein both the called and calling plarforms are initiating dialogs using reliable exchanges.


Calling Platform
OCSBC
Called Platform
SIP INVITE
SDP offer (audio,video)
100 Trying (INVITE)
SIP INVITE
SDP offer (audio, video)
100 Trying (INVITE)
183 reply (INVITE)
To-Tag= Dialog_1
SDP answer (audio, video)
183 reply (INVITE)
To-Tag= Dialog_1
SDP offer audio, video
SIP PRACK
To-Tag=Dialog_1
SIP PRACK
To-Tag=Dialog_1
200 OK (PRACK)
To-Tag= Dialog_1
200 OK (PRACK)
To-Tag= Dialog_1
SIP UPDATE
To-Tag=Dialog_1
SDP offer (audio, video)
SIP UPDATE
To-Tag=Dialog_1
SDP offer (audio, video)
200 OK (UPDATE)
To-Tag= Dialog_1
SDP answer (AMR-WB)
200 OK (UPDATE)
To-Tag= Dialog_1
SDP answer (AMR-WB)
MRF connecting a tone/announcement
180 reply (INVITE)
To-Tag= Dialog_2
SDP answer (audio, video:no)
180 reply (INVITE)
To-Tag= Dialog_2
SDP answer (audio, video:no)
SIP PRACK
To-Tag=Dialog_2
SIP PRACK
To-Tag=Dialog_2
200 OK (PRACK)
To-Tag= Dialog_2
200 OK (PRACK)
To-Tag= Dialog_2
SIP UPDATE
To-Tag=Dialog_2
SDP offer (audio, video:no)
SIP UPDATE
To-Tag=Dialog_2
SDP offer (audio, video:no)
200 OK (UPDATE)
To-Tag= Dialog_2
SDP answer (audio, video:no)
200 OK (UPDATE)
To-Tag= Dialog_2
SDP answer (audio, video:no)
200 OK (INVITE)
To-Tag=Dialog_1
200 OK (INVITE)
To-Tag=Dialog_1
SIP ACK
To-Tag=Dialog_1
SIP ACK
To-Tag=Dialog_1
Media flows set according to Dialog 1 negotiation (audio, video)

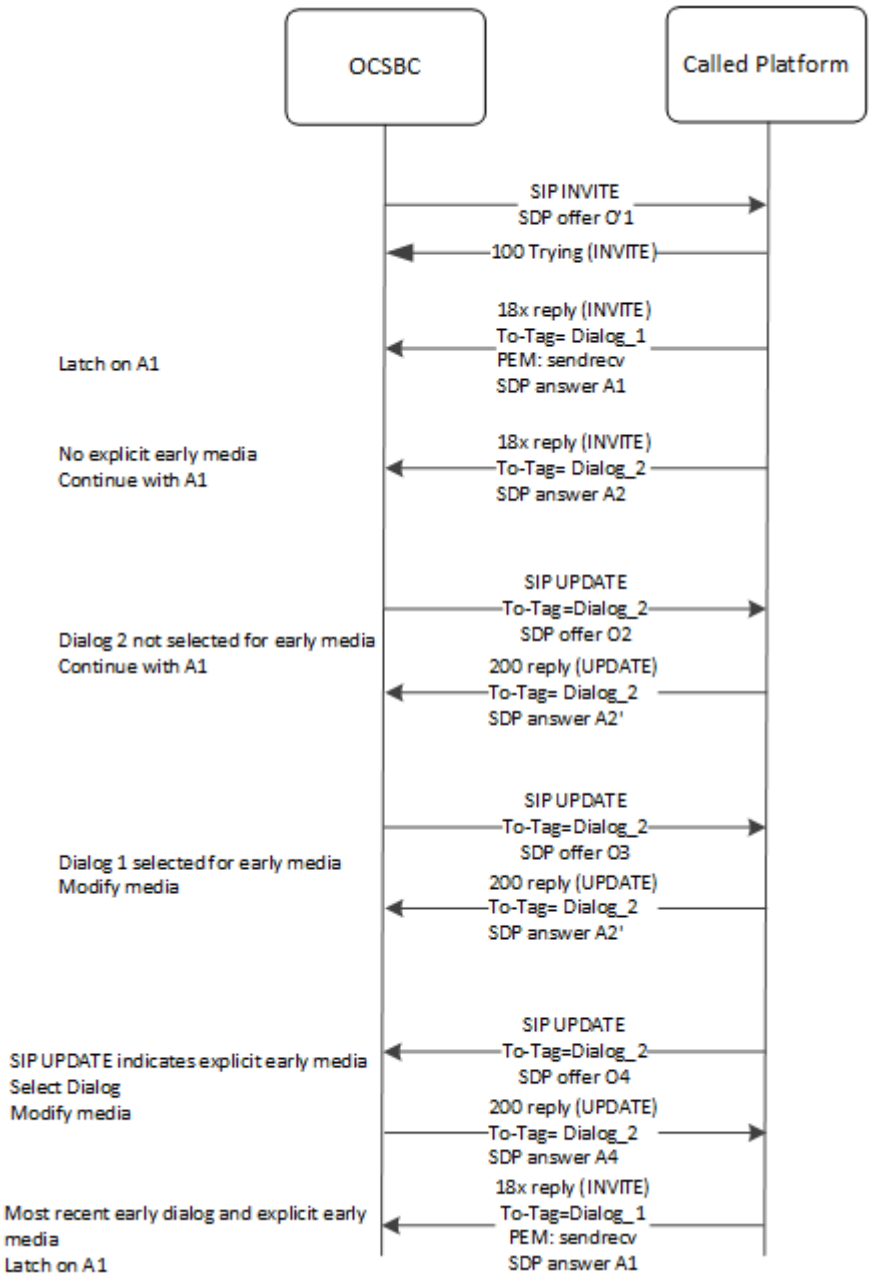
PEM Header Support for Early Dialogs

PEM support on the SBC is fully documented in the section P-Early-Media SIP Header Support. Note that PEM header support is disabled by default, which causes the SBC to ignore all PEM headers and not enforce directionality. Within the context of multiple dialogs, an additional behavior supports early multiple dialog merging.

When you set the **sip-interface**, **p-early-media-header** parameter to **support**, you enable PEM processing in replies without interfering with the signaling, thereby supporting multiple early dialog merge. This attribute value causes the SBC to perform the following:

- The SBC enforces received PEM from trusted sources if they are not restricted by some other condition.
- The SBC does not modify PEM direction; the SBC ignores the direction attribute on the applicable **sip-interface**.
- If a PEM header is absent from a SIP reply, the SBC does not add it.
- If PEM support is not advertised in the initial SIP INVITE, the SBC adds it.
- The SBC forwards PEM in SIP replies to any calling party (trusted or untrusted) that advertises PEM support.

Regarding the early dialog media selection process, the SBC is expected to retain only the last received early media with P-Early-Media set to sendonly or sendrecv. This support extends to call flows that include many-to-many dialog presentation, within which both the UAC and UAS initiate multiple dialogs, using UPDATES for example to trigger a media change. The call flow below presents an example of the applicable behavior.



Configuring Multiple Early Dialog Support

You enable the parameters below to enable specific components of multiple early dialog support. Refer to Related Feature Dependencies to ensure you consider complementary and conflicting configuration properly for your deployment.

Use the following procedure to configure default multiple early dialog support.

1. Navigate the **sip-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
```

2. Apply the **multiple-dialogs-enhancement** option.

```
ORACLE(sip-config)#options +multiple-dialogs-enhancement
```

3. Use **done**, **exit** to save the configuration.
4. Navigate to the **realm-config** configuration element.

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

   Select the desired **realm-config** or create a new one.

5. Enable the **merge-early-dialogs** parameter.

```
ORACLE(realm-config)#merge-early-dialogs enable
```

6. Use **done**, **exit** to save the configuration.
7. Navigate to the **sip-interface** configuration element.

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

   Select the desired **sip-interface** or create a new one.

8. Enable the **p-early-media-header** parameter.

```
ORACLE(sip-interface)#p-early-media-header enable
```

9. Use **done**, **exit** configuration mode, and perform a subsequent **verify-config** and **activate-config** to complete the configuration.

   These configurations support real-time configuration, and therefore do not require a reboot.

Selecting SDP within Multi-Dialog Call Scenarios

By default, the Oracle Communications Session Border Controller saves SDP presented in a series of early dialogs using To tags to differentiate between dialogs. If the session continues with a 200OK that does not include SDP, the Oracle Communications Session Border Controller refers to the To tag to identify the dialog from which the Oracle Communications Session Border Controller selects the SDP for the media flow. This complies with 3GPP TS 24.628, TS 24.182 and RFC 5009 behavior for sessions supporting early media.

Consider a SIP dialog with early media that proceeds by establishing call scenarios in which the final 200 OK does not include any SDP. This messaging may include multiple 183 (Session Progress) messages with SDP that differ from each other and include different To tags, establishing multiple early dialogs. In these scenarios, the Oracle Communications Session Border Controller uses the saved SDP from the dialog that matches the dialog indicated in the 200 OK's To tag to anchor the media. If the 200 OK includes SDP, the Oracle Communications Session Border Controller uses that SDP to anchor the media.

> **Note:**
>
> The user can disable this behavior, for example, to use the functional behavior in Oracle Communications Session Border Controller version S-CZ7.2.0 and below, which is to use the last SDP seen as the source for SDP. Deployments that rely on this behavior must revert to it using the **sip-config**'s **dont-save-early-dialog-sdp** option parameter.

```
ORACLE(sip-config)# options +dont-save-early-dialog-sdp
```

SDP Compliance Enforcement

You can configure the SBC to enforce SDP compliance on incoming messages and reject non-compliant messages and change the non-compliant SDP in ensuing messages. By default, the SBC forwards response message even if the Content-Length is greater than the SDP size and the SDP does not have mandatory parameters. You enable the **sip-strict-compliance** option when the SBC is operating in environments where it is expected to monitor and validate these aspects of SDP.

When you enable the **sip-strict-compliance** option in the **sip-config**, the SBC:

- Ignores/drops any response message if the SDP Content-Length header value is greater than the SDP size. The system also increments the "Invalid Responses" statistic.
- Forwards any response message without any SDP if the message does not include mandatory SDP parameters. Mandatory parameters include:
  - version
  - origin
  - session-name
  - connection
  - timer

  The SBC provides an "SDP Stripped Responses" statistic in the sip-errors HDR group to track SDP stripped responses system-wide. Enable this HDR group to obtain statistics on SDP responses the SBC has removed because the original message violated the mandatory SDP parameter requirements.

SDP Compliance Enforcement Configuration

To enforce SDP compliance for SIP sessions:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

If you are adding support for this feature to a preexisting configuration, you must select (using the ACLI **select** command) the single instance **sip-config** element.

4. **options**—Set the options parameter by typing options, a Space, and then the option name **sip-strict-compliance**. Then press Enter.

```
ORACLE(sip-config)# options +sip-strict-compliance
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

SIP Duplicate SDP Suppression

Using the **strip-dup-sdp** option in the SIP configuration, you can enable your Oracle Communications Session Border Controller to suppress a duplicate SDP answer in the reliable responses (1xx and 2xx) in an INVITE transaction.

During INVITE transactions in certain networks, SDP answers in reliable provisional responses (1xx) can cause interoperability issues. This issue occurs when the UAS includes the SDP answer in subsequent 1xx responses or in the final 2xx response, and the UAC views that inclusion as a protocol violation. The UAC does so based on the fact that the SDP answer was reliably delivered to it in a 1xx response, and so it views additional SDP information as unnecessary in and ensuing reliable 1xx or 200 OK responses.

RFCs 3216 and 3262 do not specifically call out the UAS's including SDP information in this way as a protocol violation. Still, the system allows you set enable the **strip-dup-sdp** option as a means of preventing the UAC from terminating sessions. With this option enabled, the Oracle Communications Session Border Controller removes the SDP answer in subsequent reliable provisional or final 200 OK responses if it is identical to the SDP answer previously received.

SIP Duplicate SDP Suppression Configuration

To enable duplicate SDP suppression for SIP sessions:

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

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

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

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

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

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name **strip-dup-sdp**. Then press Enter.

```
ORACLE(sip-config)# options +strip-dup-sdp
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

SIP SDP Address Correlation

SIP SDP address correlation ensures that when the Oracle Communications Session Border Controller receives a request containing SDP, the L3 source address of the request is compared against the address in the c-line of the SDP. When the addresses match, the session proceeds as it normally would. If there is a mismatch, the call is rejected with the default 488 status code. You can also configure the code you want to use instead of 488.

This functionality works only with non-HNT users. The value c=0.0.0.0 is an exception and is always processed.

SIP SDP Address Correlation Configuration Address Checking

The **sdp-address-check**, in the **enforcement-profile** element can be set to enable the SDP address correlation.

To enable SDP address checking:

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

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

2. Type **session-router** and press Enter.

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

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

```
ORACLE(session-router)# enforcement-profile
ORACLE(enforcement-profile)#
```

4. Use the ACLI **select** command so that you can work with the enforcement profile configuration to which you want to add this parameter.

```
ORACLE(enforcement-profile) select
```

5. **sdp-address-check**—Enable or disable SDP address checking on the Oracle Communications Session Border Controller. The default for this parameter is **disabled**.

```
ORACLE(enforcement-profile)# sdp-address-check enabled
```

6. Save and activate your configuration.

   If a mismatch occurs and you want to reject the call with a status code other than 488, you set the code you want to use in the local response code map entries.

SIP SDP Address Correlation Configuration Mismatch Status Code

To apply a new status code to a SDP address correlation mismatch:

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

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

2. Type **session-router** and press Enter.

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

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

```
ORACLE(session-router)# local-response-map
ORACLE(local-response-map)#
```

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

```
ORACLE(local-response-map)# entries
ORACLE(local-response-map-entry)#
```

   From here, you can view the entire menu for the local response map entries configuration by typing a **?**.

5. **local-error**—Enter sdp-address-mismatch for which to apply the new status code.
6. **sip-status**—Set the SIP response code to use.
7. **sip-reason**—Set the SIP reason string you want to use for this mapping.

```
ACMEPACKET(local-response-map-entry)# local-error sdp-addressmismatch
ACMEPACKET(local-response-map-entry)# sip-status 403
ACMEPACKET(local-response-map-entry)# sip-reason sdp address mismatch
```

8. Save and activate your configuration.

SIP SDP Address Correlation Configuration Enforcement Profile

To apply an enforcement profile to a realm:

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

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

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

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

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

4. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this realm.

```
ORACLE(realm-config)# enforcement-profile profile1
```

5. Save and activate your configuration.

SDP Insertion for (Re)INVITEs

If your network contains some SIP endpoints that do not send SDP in ReINVITEs but also contains others that refuse INVITEs without SDP, this feature can facilitate communication between the two types. The Oracle Communications Session Border Controller can insert SDP into outgoing INVITE messages when the corresponding, incoming INVITE does not contain SDP.

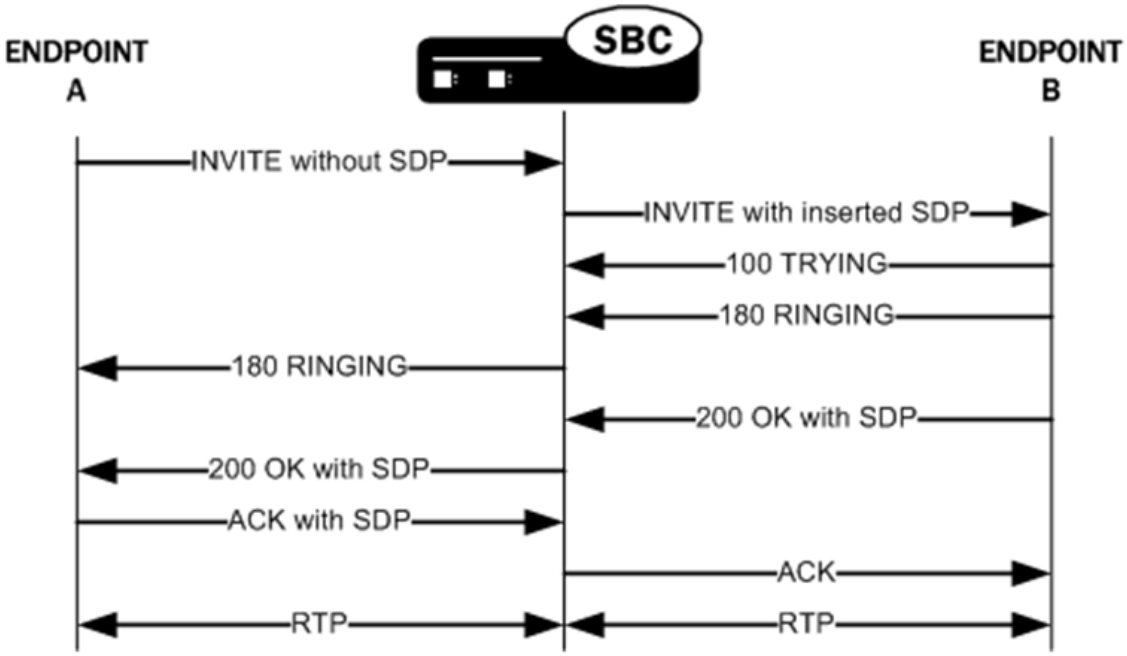
You can also use this feature when the network devices used in H.323-SIP interworking do not include SDP in the INVITEs sent to SIP endpoints. In this case, the Oracle Communications Session Border Controller can insert SDP in the outgoing INVITE messages it forwards to the next hop.

This feature works for both INVITEs and ReINVITEs.

This section explains how the SDP insertion feature works for INVITEs and ReINVITEs. The examples used this section are both pure SIP calls. Even when you want to use this feature for IWF calls, though, you configure it for the SIP side.

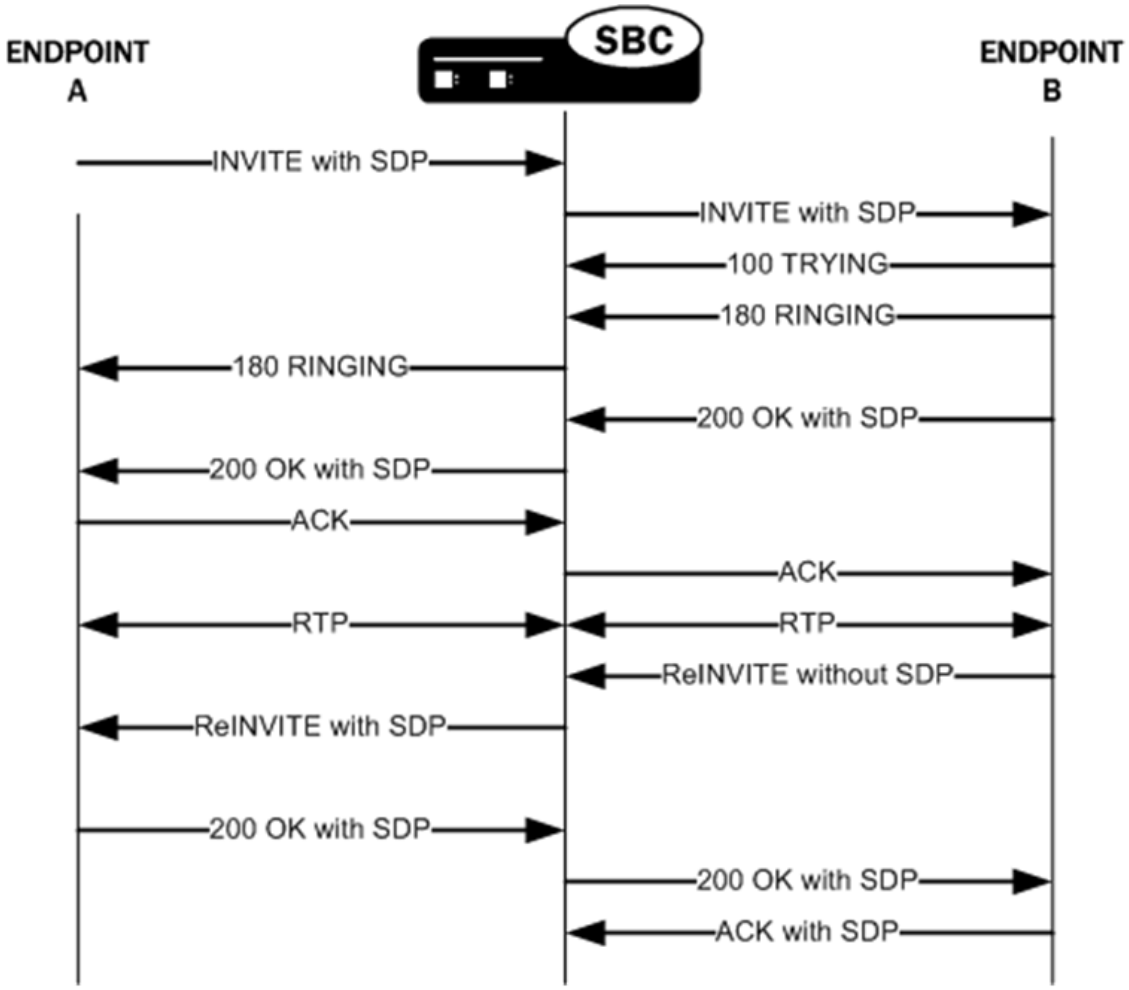
SDP Insertion for SIP INVITES

With the parameters mentioned above appropriately configured, the SBC inserts SDP into an outgoing INVITE when the corresponding incoming INVITE has none. Because no SDP information is available for the session, the SBC uses a media profile from a list of them you configure and then apply for SDP insertion.



SDP Insertion for SIP ReINVITEs

The section explains SDP insertion for ReINVITEs, using a case where SIP session has been established with an initial INVITE containing SDP. In the diagram below, you can see the initial INVITE results in a negotiated media stream. But after the media stream is established, Endpoint B sends a ReINVITE without SDP to the SBC. In this case, the SBC inserts the negotiated media information from the initial INVITE as the ReINVITE's SDP offer. For subsequent ReINVITEs with no SDP, the SBC inserts the negotiated media information from the last successful negotiation as the ReINVITE's SDP offer. It then sends this ReINVITE with inserted SDP to the next hop signaling entity.



SDP Insertion Configuration

This section shows you how to configure SDP insertion for the calls cases described above.

Configuring SDP Insertion for SIP INVITEs

To work properly, SDP insertion for SIP invites requires you to set a valid media profile configuration.

To enable SDP insertion for INVITEs:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-interface** and press Enter.

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

4. **add-sdp-invite**—Change this parameter from disabled (default), and set it to **invite**. To include an SDP in both the INVITE and ReINVITE, change this parameter to **both**.
5. **add-sdp-profile**—Enter a list of one or more media profile configurations you want to use when the system inserts SDP into incoming INVITEs that have no SDP. The media profile contains media information the Oracle Communications Session Border Controller inserts in outgoing INVITE.

   This parameter is empty by default.
6. Save and activate your configuration.

Configuring SDP Insertion for SIP ReINVITEs

In this scenario, the Oracle Communications Session Border Controller uses the media information negotiated early in the session to insert after it receives an incoming ReINVITE without SDP. The Oracle Communications Session Border Controller then sends the ReINVITE with inserted SDP to the next hop signaling entity. You do not need the media profiles setting for ReINVITEs.

To enable SDP insertion for ReINVITEs:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-interface** and press Enter.

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

4. **add-sdp-invite**—Change this parameter from disabled (default), and set it to **reinvite**. To include an SDP in both the INVITE and ReINVITE, change this parameter to **both**.
5. Save and activate your configuration.

SDP Version Change without SDP Body Change

When an SRTP call is made through the Oracle Communications Session Border Controller and a UE sends a reINVITE, there may be no change in the SDP contents. When the other UE responds, some devices will increment the o= line's session version, despite no SDP change. Normally the change in SDP version indicates that the SDP has changed, which would otherwise require the Oracle Communications Session Border Controller to modify the media flow set-up. In order to leave the media flows unchanged when session version changes but the rest of the SDP does not, an option is configured in the **media-manager-config**.

If the Oracle Communications Session Border Controller attempts to modify these media flows when unnecessary, calls could be dropped by the system disrupting media packet sequence number synchronization.

The **ignore-reinv-session-ver** option is set to handle this situation. When configured, the Oracle Communications Session Border Controller compares the current SDP received from a UE against the previously-received SDP from the same UE. If the SDP in the newer and older messages is the same, the Oracle Communications Session Border Controller ignores any changes to the session-version and will not modify the media flow portion of the call.

> **Note:**
>
> When the SDP change is only a reordering of SDP lines without any other change, the option has no effect.

SDP Version Change Configuration

To configure media over TCP:

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

   ```
   ORACLE# configure terminal
   ```

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

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter to begin configuring media over TCP.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **options** — Set the options parameter by typing **options**, a Space, the option name **ignore-reinv-session-ver** with a “plus” sign in front of it, and then press Enter.

   ```
   ORACLE(media-manager-config)# options +ignore-reinv-session-ver
   ```

   If you type the option without the “plus” sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration’s options list, you must prepend the new option with a “plus” sign as shown in the previous example.

   Save and activate your configuration.

Enhanced SIP Port Mapping

This section explains how to configure SIP port mapping feature to support:

- Anonymous requests from endpoints
- Cases where endpoints dynamically change transport protocols between UDP and TCP

Anonymous Requests

If a SIP endpoint sends an INVITE message with a From header that is anonymous, the Oracle Communications Session Border Controller can find the registration cache entry by using the Contact and Via headers. In cases such as instant messaging (IM), where there is no Contact header, the Oracle Communications Session Border Controller can use the Via header.

The Oracle Communications Session Border Controller's checks whether the reg-via-key option is configured for the access-side SIP interface where a REGISTER is received. If the option is enabled, the Oracle Communications Session Border Controller makes the via-key by adding the IP address from the Via header to the firewall address (if there is a firewall present between the Oracle Communications Session Border Controller and the endpoint).

When an INVITE arrives at a SIP interface where this option is enabled, the Oracle Communications Session Border Controller determines whether the From header is anonymous or not. If it is anonymous, then the Oracle Communications Session Border Controlleruses the Via-key to find the registration entry.

Anonymous SIP Requests Configuration

To enable support for anonymous SIP requests:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

4. Type **options +reg-via-key** and press Enter.

   ```
   ORACLE(sip-interface)# options +reg-via-key
   ```

   If you type **options reg-via-key** without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

SIP Registration Via Proxy

The Oracle Communications Session Border Controller supports a number of features that require it to cache registration information for UAs (endpoints) registering and receiving requests through it. For those features to operate correctly, the Oracle Communications Session Border Controller must act as the outbound proxy through which these endpoints register.

In order to support deployments where a proxy sits between the Oracle Communications Session Border Controller and the endpoints, the Oracle Communications Session Border Controller must consider the bottom Via header it receives from endpoints when constructing and matching cache registration entries. And when you use SIP port mapping, the system must use the bottom Via header as a way to determine the endpoint uniquely so that it can have a unique mapping port when the SIP interface is configured with the reg-via-key=all option.

Using the reg-via-proxy option, you can enable your Oracle Communications Session Border Controller to support endpoints that register using an intervening proxy. You can set this option either for a realm or for a SIP interface. If you use it for a SIP interface, add to the SIP interface pointing toward the proxy and endpoints—the access side.

Considerations for Reg-Via-Key and Port Mapping

When you set the **reg-via-proxy** option, the Oracle Communications Session Border Controller includes the bottom Via header from received requests in the registration cache Via Key. The system also uses it for determining whether or not the request matches a registration cache entry. Each unique bottom Via received a unique mapping port when you turn SIP port mapping on and set the SIP interface with the **reg-via-key=all** option.

Request Routing

So that requests addressed to the corresponding registered contact are routed to the proxy, the Oracle Communications Session Border Controller includes the intervening proxy (i.e., the top Via) in the routing information for the registration cache when you set **reg-via-proxy**. To carry out this routing scheme, the system adds a Path header (if none is present) to the REGISTER. But it removes the Path header prior to sending the REGISTER to the registrar.

Note that when the received REGISTER contains a Path header, the Oracle Communications Session Border Controller uses it for routing requests toward the endpoint and includes it in the forwarded REGISTER request—as is the case when you do not enable SIP registration via proxy.

SIP Registration Via Proxy Configuration

To configure SIP registration via proxy:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

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

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

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

4. Type **options +reg-via-proxy** and press Enter.

```
ORACLE(sip-interface)# options +reg-via-proxy
```

If you type options reg-via-proxy without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

Dynamic Transport Protocol Change

The Oracle Communications Session Border Controller also uses the IP address and port in the Contact and Via headers. This is useful for cases when endpoints dynamically change transport protocols (TCP/UDP), and the port number used for sending an INVITE might not be the same one used to send a Register message.

If you do not enable this feature, when an endpoint registered with the Oracle Communications Session Border Controller used UDP for its transport protocol, a call fails if that endpoint subsequently initiates the call using TCP. The Oracle Communications Session Border Controller checks for the Layer 3 IP address and port, and it rejects the call if the port is changed.

With the new option reg-no-port-match added to the SIP interface configuration, the Oracle Communications Session Border Controller will not check the Layer 3 port in the INVITE and REGISTER messages.

Dynamic Transport Protocol Change Configuration

To enable dynamic transport protocol change:

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

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

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

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

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

4. Type **options +reg-no-port-match** and press Enter.

```
ORACLE(sip-interface)# options +reg-no-port-match
```

If you type options reg-no-port-match without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

SIP Privacy Extensions

This section explains how you can configure privacy services to be applied only when the source is trusted and the destination is considered untrusted. (Prior to this release, the Oracle Communications Session Border Controller always applied the privacy services, unless the source and the destination were both trusted.)

The Oracle Communications Session Border Controller considers all user endpoints and nodes outside the core as untrusted.

The Oracle Communications Session Border Controller acts as the boundary device between the trusted platform and the untrusted Internet, to implement privacy requirements. When it receives a message, the Oracle Communications Session Border Controller checks whether the source is trusted. It evaluates the level of privacy requested in a Privacy header, if present.

Depending on whether the source is trusted or untrusted, the Oracle Communications Session Border Controller can do different things when passing the message to the outgoing side. It also checks whether the destination is trusted.

Privacy Types Supported

The Oracle Communications Session Border Controller supports the following Privacy types:

- user: user-level privacy function provided. Any non-essential informational headers are removed, including the Subject, Call-Info, Organization, User-Agent, Reply-To, and In-Reply-To. Possibly the original value of the From header is changed to anonymous.
- header: headers that cannot be set arbitrarily by the user (Contact/Via) are modified. No unnecessary headers that might reveal personal information about the originator of the request are added. (The values modified must be recoverable when further messages in the dialog need to be routed to the originator.)
- id: third-party asserted identity kept private with respect to SIP entities outside the trust domain with which the user authenticated.

The following SIP headers can directly or indirectly reveal identity information about the originator of a message: From, Contact, Reply-To, Via, Call-Info, User-Agent, Organization, Server, Subject, Call-ID, In-Reply-To and Warning.

user

The Oracle Communications Session Border Controller supports the Privacy type user. It can remove non-essential information headers that reveal user information by:

- Setting the SIP From header and display information to anonymous
- Removing the Privacy header
- Removing Proxy-Require option tag = privacy (if present)
- Removing the following headers:
  Subject

  Call-Info

  Organization

  User-Agent

  Reply-To

In-Reply-To

header

The Oracle Communications Session Border Controller also supports the Privacy type header. It modifies SIP headers that might reveal the user identity by:

- Stripping the Via header
- Replacing the Contact header
- Stripping Record-Route
- Removing the Privacy header
- Removing Proxy-Require option tag = privacy (if present)

In general, the B2BUA behavior of the Oracle Communications Session Border Controller by default provides header privacy for all sessions.

id

The Oracle Communications Session Border Controller also supports the Privacy type id. It keeps the Network Asserted Identity private from SIP entities outside the trusted domain by:

- Stripping only P-Asserted-Identity
- Removing the Privacy header and Proxy-Require option-tag = privacy
- Setting the From header to anonymous (for the backward compatibility)

Examples

The following examples show the actions the Oracle Communications Session Border Controller performs depending on the source and target of the calls.

Calls from Untrusted Source to Trusted Target

When calls are from an untrusted source to a trusted target and PPI is included in the INVITE sent to IP border elements, the Oracle Communications Session Border Controller maps the PPI information to PAI in the outgoing INVITE to the trusted side (even if the Privacy header is set to id or to none). The Privacy and From headers get passed on unchanged.

IP border elements must pass PAI (if received in the ingress INVITE) and the From and Privacy headers to the egress side just as they were received on the ingress side.

The Oracle Communications Session Border Controller maps the PPI to PAI by default, if the outgoing side is trusted. To change this behavior, you need to configure the disable-ppi-to-pai option.

Calls from Trusted to Untrusted

When calls are from a trusted source to an untrusted target, and the Privacy header is set to id, the Oracle Communications Session Border Controller strips PAI, makes the From header anonymous, and strips the Privacy header.

If the Privacy header is set to none, the Oracle Communications Session Border Controller does not change the From header and passes on the Privacy header, if there is one.

Calls from Trusted to Trusted

When calls are going from trusted source to trusted target acting as a peer network border element and PPI is included, the Oracle Communications Session Border Controller maps PPI to PAI. The Privacy header remains the same as signaled and the Oracle Communications Session Border Controller passes the From header and the PAI without changes.

Configuring SIP Privacy Extensions

Prior to this release the session agent's trust mode provided this functionality. Now you configure SIP interface's trust-mode as none, which means nothing is trusted for this SIP interface.

You also configure the disable-ppi-to-pai parameter disable the changing of the P-Preferred header to the P-Asserted-Identity header, if the outgoing side is trusted.

Trust Mode

To configure the trust mode:

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

   ```
   ORACLE# configure terminal
   ```

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

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

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

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

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. If configuring an existing interface, enter the select command to select the interface.
5. **trust-mode**—Select the trust mode for this SIP interface. The default value is **all**. The valid values are:
   - **all**—Trust all previous and next hops except untrusted session agents
   - **agents-only**—Trust only trusted session agents
   - **realm-prefix**—Trusted only trusted session agents or address matching realm prefix
   - **registered**—Trust only trusted session agents or registered endpoints
   - **none**—Trust nothing
6. Save and activate your configuration.

The following example shows the trust-mode set to none. The remaining SIP interface options are omitted for brevity.

```
sip-interface
        state                           enabled
        realm-id                        access1
        sip-port
                address                         192.168.1.30
                port                            5060
                transport-protocol              UDP
                allow-anonymous                 all
        carriers
        proxy-mode                      Proxy
        redirect-action
        contact-mode                    maddr
        nat-traversal                   none
        nat-interval                    30
        registration-caching            disabled
        min-reg-expire                  300
        registration-interval           3600
        route-to-registrar              disabled
        teluri-scheme                   disabled
        uri-fqdn-domain
        options
        trust-mode                      none
```

Disabling the PPI to PAI Change

To disable the changing of PPI to PAI:

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

   ```
   ORACLE# configure terminal
   ```

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

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

3. Type **sip-config** and press Enter. The system prompt changes.

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

   From this point, you can configure SIP configuration parameters. To view all sip-config parameters, enter a **?** at the system prompt.

4. If configuring an existing SIP configuration, enter the select command to select it.
5. **options**—Enter **disable-ppi-to-pai**. If adding to an existing list of options, use a preceding plus (+) sign.

   ```
   options +disable-ppi-to-pai
   ```

6. Save and activate your configuration.

SIP Registration Cache Limiting

Using SIP registration cache limiting for SIP endpoint access deployments, you can restrict the size of the SIP registration cache for the global SIP configuration.

You can implement this feature if you have been seeing issues where, either due to network failure scenarios or incorrect sizing of system capabilities, the Oracle Communications Session Border Controller and/or the SIP registrar cannot support the number of registering endpoints. Although the Oracle Communications Session Border Controller protects itself and the registrar against SIP REGISTER floods, conditions can still occur where too many legitimate endpoints attempt to register with the registrar via the Oracle Communications Session Border Controller.

By enabling SIP registration cache limiting, you restrict the number of legitimate endpoints that can register. The Oracle Communications Session Border Controller rejects any endpoints beyond the limit you set. If you do not want to use this feature, simply leave the reg-cache-limit parameter set to its default of 0, meaning there is no limit to the entries in the SIP registration cache.

When you limit the number of registered endpoints allowed in the Oracle Communications Session Border Controller's registration cache, the Oracle Communications Session Border Controller analyzes each registration before starting to process it. First, the Oracle Communications Session Border Controller checks the contact header to determine if it is already in the list of contacts for the user. If it finds the contact in its cache list, the Oracle Communications Session Border Controller treats the registration as a refresh; it treats any other headers as new. Note that the Oracle Communications Session Border Controller checks the message prior to making any changes to the cache because it must either accept or reject the message as a whole.

The Oracle Communications Session Border Controller adds the number of new contacts to the number already present in the cache, and rejects any registration with a contact that would cause it to exceed its limit. Rejection causes the Oracle Communications Session Border Controller to send a response communicating that its registration cache is full. The default response is the 503 Registration DB-Full message, but you can use the SIP response mapping feature to use another message if required.

You can set an option in the global SIP configuration that defines the value in the Retry-After header. The Oracle Communications Session Border Controller sends this header as part of its rejection response when the registration cache is full. Another option sets the percentage of the registration cache size which, if exceeded, causes the Oracle Communications Session Border Controller to send an alarm.

About Registration Cache Additions Modifications and Removals

When it receives a REGISTER message with new contact information for a user, the Oracle Communications Session Border Controller considers it an addition to the cache and augments the number of registration cache entries. Then the Oracle Communications Session Border Controller forwards the message to the registrar, and—when and only when the registrar returns both the original and new contacts in the 200 OK—the registration cache count stays the same. However, if the registrar returns only the new contact (making this a case of modification), then the Oracle Communications Session Border Controller removes the old contact information and subtracts accordingly from the number of registration cache entries.

Thus the Oracle Communications Session Border Controller does not know whether a REGISTER might result in an addition or a modification until it receives a response from the registrar. For this reason, the Oracle Communications Session Border Controller first assumes

it is to make an addition, and then updates the registration cache and count when it has the necessary information from the registrar.

The registration cache count does not reflect removals during the rejection check because the Oracle Communications Session Border Controller ignores registration messages or expires headers with their expires values set to zero when it counts new entries. The fact that removals take place after additions and modifications means that messages which remove one contact while adding another might be rejected. That is, the addition might exceed the registration cache limit before any removal can take place to make room for it.

Registration Cache Alarm Threshold

A percentage of the registration cache limit, the registration cache alarm threshold is a configurable value you can set to trigger an alarm when the registration cache is reaching its limit. When exceeded, this threshold triggers the generation of an alarm and SNMP trap. When registrations fall back beneath the threshold, the Oracle Communications Session Border Controller clears the alarm and sends a clear trap.

This alarm is Major in severity, and its text reads as follows:

```
Number of contacts <registration count> has exceeded the registration cache
threshold <threshold %> of <registration cache limit value>.
```

Notes on Surrogate Registration

The Oracle Communications Session Border Controller does not, under any circumstances, reject surrogate registrations on the basis of the registration cache limit. However, surrogate registrations generate contacts, and so they do add to the global registration count. In the case where the surrogate registrations add to the registration count to the extent the count exceeds the limit you configure, you will have more registrations in the cache than the configured limit.

Monitoring Information

You can monitor how many entries are in the SIP registration cache using the ACLI **show registration** command and referring to the Local Contacts statistics.

SIP Registration Cache Limiting Configuration

This section shows you how to configure the registration cache limit, and how to set the options controlling retry times and thresholds for alarm purposes.

To configure SIP registration cache limiting:

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

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

2. Type **session-router** and press Enter.

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

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

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

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

4. **registration-cache-limit**—Set the registration cache limit, or the maximum number of SIP registrations that you want to keep in the registration cache. The minimum and default value for this parameter is 0, and you can set it to a maximum value of 999999999. Leaving this parameter set to 0 means there is no limit on the registration cache (and therefore leaves this feature disabled).

5. **options**—Set the options parameter by typing options, a Space, the option name **reg-cache-lim-retry-after=X** (where X is the value added to the Retry-After header) with a plus sign in front of it. This option defaults to 1800, and you can enter values from 0 to 999999999.

   You can configure the alarm threshold option the same way, substituting the option name **reg-cache-alarm-thresh=X** (where X is the percentage of registration cache limit that triggers an alarm). This option defaults to 95, and you can enter value from 0 to 100.

```
ORACLE(sip-config)# options +reg-cache-lim-retry-after=2500
ORACLE(sip-config)# options +reg-cache-alarm-thresh=90
```

   If you type options and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

SIP Registration Overload Protection

You can configure your Oracle Communications Session Border Controller for SIP Registration overload protection, which augments the Oracle Communications Session Border Controller's protection methods. Working with the Oracle Communications Session Border Controller's access control and registration caching functions, this new feature guards against benign avalanche restarts. The avalanche is caused by events where many endpoints lose power or connectivity at once, are restored to service, and then flood the Oracle Communications Session Border Controller as they attempt to register again.

Normally, the Oracle Communications Session Border Controller handles SIP registration by creating a temporary registration cache for the endpoint's address of record (AoR) and forwards the REGISTER request to the registrar. To challenge the endpoint's registration, the registrar sends back either a 401 Unauthorized or 407 Proxy Authorization Required response. When it receives the 401 or 407, the Oracle Communications Session Border Controller saves the challenge context in anticipation of receiving a second REGISTER with the endpoint's authentication credentials. The Oracle Communications Session Border Controller forwards the second REGISTER (with authentication credentials) to the registrar, and then the registrar confirms registration with a 200 OK. Both REGISTER requests are subject to the system's access control rules, set either for the ingress realm or the ingress session agent. The Oracle Communications Session Border Controller also honors the maximum registration sustain rate constraint for session agents; this applies when the incoming REGISTER is from a session agent and the outgoing REGISTER is sent to a session agent.

When you enable SIP Registration overload protection, the Oracle Communications Session Border Controller temporarily promotes the endpoint to the trusted level when it receives the 401 or 407 response (to the first REGISTER) from the registrar. This ensures that the second REGISTER (containing authentication credentials) can reach the Oracle Communications Session Border Controller. Temporary promotion lasts only for the amount of time remaining before the REGISTER server transaction expires plus the time allotted in the transaction expiration parameter in the SIP configuration. Before the temporary promotion expires, there is enough time for any necessary retransmissions of the first REGISTER and for the second REGISTER to take place. The following situations might also occur:

- If the Oracle Communications Session Border Controller receives a 401 or 407 to the second REGISTER request, it resets its access control level for the endpoint's address to the default level; it then treats additional REGISTER requests from the same context at the default access control level.
- If the Oracle Communications Session Border Controller receives a 200 OK response to the REGISTER message, it extends the promotion time to the expiration period for the registration cache.

If the Oracle Communications Session Border Controller is able to find the temporary registration cache and the saved challenge context when the second REGISTER arrives, it forwards the REGISTER without checking the maximum registration sustain rate constraint for ingress and egress session agents—thereby ensuring that the REGISTER with authentication credentials is sent to the registrar. So when you use this feature, you should set the maximum registration sustain rate constraint of the session agent (representing the registrar) at half the registrar's maximum registration sustain rate. Additional REGISTER requests with the same challenge context are subject to the maximum registration sustain rate constraint.

SIP Registration Overload Protection Configuration

When you configure this feature, be sure to set the **reg-overload-protect** option in your global SIP configuration:

To enable SIP Registration overload protection on your Oracle Communications Session Border Controller:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

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

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

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

4. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**reg-overload-protect**), and then press Enter.

   ```
   ORACLE(sip-config)# options +reg-overload-protect
   ```

If you type either of these options without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

> **Note:**
>
> Note that the sip-config option "cache-challenges" (enabled by default) must not have been disabled for SIP Registration Overload Protection to work properly. If you have disabled cache-challenges, re-evaluate the reason you disabled it. If registration overload protection supersedes your reason for disabling cache-challenges, re-enable the option as shown below.

```
ACMEPACKET(sip-config)# options +cache-challenges=yes
```

Note that the configuration syntax above is equivalent to the following, which uses the "-" character to remove the option.

```
ACMEPACKET(sip-config)# options -cache-challenges
```

5. Save and activate your configuration.

SIP Registration Overload Protection for IMS-AKA

The SIP Registration Overload Protection (SROP) feature supports registrations via IMS-AKA. From the endpoint's perspective, overload protection for IMS-AKA endpoints is the same as for other endpoints. From the Oracle Communications Session Border Controller's perspective, however, overload protection functions differently with IMS-AKA, using ACLs to manage connectivity with the endpoint. De-registration support remains the same, either explicitly by UE signaling or by registration timeout. The ingress realm must be set to low or medium trust level.

SROP, with or without IMS-AKA, requires that the user enable both **reg-overload-protect** and **cache-challenges** in the sip-config.

For IMS-AKA SROP, the system creates one ACL entry for the path between the endpoint’s secure client port and the SBC's secure server port. This ACL allows both TCP as well as UDP signaling traffic for the path between the endpoint’s secure server port and the SBC's secure client port. The system temporarily promotes these ACLs to trusted when it receives the registrar's challenge (401 or 407) to an IMS-AKA endpoint's REGISTER request. The promotion period is temporary.

After receiving the 200 OK, the system updates the flow session timer with the registration expire timer and the registration timers. In addition, it updates the ACL entry with the new timer value. This value is the sum of the registration expire timer plus the remaining registration cache linger value.

The temporary ACL promotion is equal to the remaining expiration time of the REGISTER server transaction plus the value of the transaction-expire parameter of the sip-interface/sip-config. If the registration does not finish within this temporary promotion, the system allows the ACL session to expire and removes the entries.

> **Note:**
>
> The Oracle Communications Session Border Controller does not replicate this ACL's security associations to a standby system.

If signaling before the registration is complete shows the endpoints have changed their secure ports, the system deletes the ACLs initially set up for the registration. There are two ways for the system to delete these entries:

- The user can set the **remove-old-secured-acl-entries** option on the ingress sip-interface. This option causes the system to delete the entry as soon as it detects the secure ports are changed.
- If the option above is not set, the system deletes the entries when the session-timer expires.

> **Note:**
>
> The system adds the **Reset UE Ipp** keyword with the IP and port of the UE as an ACL entry to the log.sipd file each time it creates these ACLs.

SIP Instance ID in Registration Cache

As defined in RFC 5626, Managing Client-Initiated Connections in the Session Initiation Protocol (SIP), the `+sip.instance` uniquely identifies a specific user agent instance. The instance-id does not change even when the User Agent is rebooted or power cycled. Using the instance-id from the `+sip-instance` parameter allows the SBC to locate a registered contact even when the IP address of the UA has changed.

The +sip.instance is a Contact header field parameter that contains a globally unique identifier, generally a Universally Unique Identifier (UUID) Uniform Resource Name (URN) that identifies a specific user agent client.

As defined in RFC 5626, Managing Client-Initiated Connections in the Session Initiation Protocol (SIP), the parameter contains an "instance-id" that uniquely identifies a specific user Agent instance. The instance-id does not change even when the User Agent is rebooted or power cycled (see Section 3.1 of RFC 5626). Using the instance-id from the +sip-instance parameter allows the SBC to locate a registered contact even when the IP address of the UA has changed.

An example REGISTER message containing a `+sip-instance` parameter and assigned value is shown below.

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/TCP 192.0.2.2;branch=z9hG4bK-bad0ce-11-1036
Max-Forwards: 70
From: Bob <sip:bob@example.com>;tag=d879h76
To: Bob <sip:bob@example.com>
Call-ID: 8921348ju72je840.204
CSeq: 1 REGISTER
Supported: path, outbound
Contact: <sip:line1@192.0.2.2;transport=tcp>; reg-id=1;
```

```
;+sip.instance="<urn:uuid:00000000-0000-1000-8000-000A95A0E128>"
Content-Length: 0
```

The user agent calculates a `+sip.instance` generally using a time-stamp, a unique string -- such as a MAC address, and/or a randomly generated number. After calculating the value, the user agent saves it to persistent storage, thus ensuring that the value is unchanged by subsequent power cycles.

SIP Instance ID and the Registration Cache

Every assignment of a new IP address to a mobile user agent client results in a new REGISTER request. The receipt of the REGISTER request, in turn, generates an additional entry in the registration cache. The implementation can be simplified by including the +sip.instance value in the criteria used to match incoming REGISTER requests with existing sessions maintained in the registration cache. When a match is found, the SBC re-uses the existing registration cache rather than adding a new one.

If such a +sip.instance match is found and the contact address has changed, the SBC determines if digest authentication has been used in previous associated REGISTER requests. If so, the SBCs forwards the REGISTER request to the registrar for authentication of the new location.

SIP Instance ID Configuration

Because this configuration is dynamically performed at the **sip-config** level, it applies to all SIP interfaces and takes effect immediately

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

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

2. Select the **sip-interface** object to edit.

   ```
   ORACLE(sip-interface)# select
   <RealmID>:
   1: realm01 172.172.30.31:5060

   selection: 1
   ORACLE(sip-interface)#
   ```

3. **match-sip-instance**—Set this parameter to enabled to use the +sip-instance-id when matching incoming calls with the registration cache. Valid values are:

   - **disabled**—(the default) disables the use of the +sip-instance-id when matching incoming calls with the registration cache
   - **enabled**—enables the use of the +sip-instance-id when matching incoming calls with the registration cache

   ```
   ACMEPACKET(sip-config)# match-sip-instance enabled
   ACMEPACKET(sip-config)#
   ```

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

SIP Request Method Throttling

You can configure throttling mechanisms for SIP INVITEs and REGISTERs using session agent constraints. However, you might want to throttle other types of SIP methods, and for those methods you should use the rate constraints configuration available both in the session constraints (which you then apply to a SIP interface or a realm) and the session agent configurations.

Oracle recommends you use session agent constraints for session-rate INVITE throttling and registration-rate for REGISTER throttling.

For SIP access deployments, you can configure rate constraints for individual method types along with a set of burst and sustain rates. These constraints can help to avoid overloading the core network. In addition, they restrain the load non-INVITE messages use, thus reserving capacity for INVITE-based sessions and Registrations

When you configure SIP request method throttling, you must exercise care because it is possible to reject in-dialog requests. Therefore, Oracle recommends you do NOT configure constraints—although the configuration allows you to and will not produce error messages or warnings if you set them—for the following SIP method types:

- ACK
- PRACK
- BYE
- INFO
- REFER

However, the Oracle Communications Session Border Controller is likely to throttle NOTIFY requests despite their being part of a Subscribe dialog.

Therefore, the methods you will most likely configure for throttling are:

- NOTIFY
- OPTIONS
- MESSAGE
- PUBLISH
- REGISTER

The Oracle Communications Session Border Controller counts Re-INVITEs and challenged responses against the throttle limit, but does not check to determine if the constraints have been exceeded for either.

You can configure separate constraints—inbound and outbound values for burst and sustain rates—for each different method type you configure. Although you should use session agent constraints (and not rate constraints) for INVITEs, if you also set up rate constraints for INVITEs, then the smallest configured value takes precedence.

About Counters and Statistics

Each rate constraint you configure for a SIP method tracks its own counters. For example, if you configure a rate constraint for the PUBLISH method, the burst and sustain rates you set for it apply only to the PUBLISH method and not to any other methods for which you might set up

rate constraints. You can, however, set the burst rate window in the session constraints configuration that will apply to all methods configured as rate constraints.

The Oracle Communications Session Border Controller captures statistics for SIP methods throttled by rate constraints for SIP interfaces and session agents; it does not capture these statistics for the global SIP configuration.

SIP Request Method Throttling Configuration

This section shows you how to set up rate constraints for session constraints (which are then applied to SIP interfaces) and session agents.

To use this feature, you must enable the **extra-method-stats** parameter in the global SIP configuration.

To set the extra-method-stats parameter in the global SIP configuration:

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

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

2. Type **session-router** and press Enter.

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

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

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
If you are adding this feature to an existing configuration, you need to
select the configuration (using the ACLI select command) before making
your changes.
```

4. **extra-method-stats**—Set this parameter to **enabled**.
5. Save and activate your configuration.

Rate Constraints for SIP Interfaces

To apply rate constraints to SIP interfaces, you need to configure rate constraints in the session constraints configuration and then apply the session constraints to the SIP interface where you want them used.

Note that you need to set up the parent **session-constraint** configuration to save any rate constraints you configure.

To configure rate constraints:

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

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

2. Type **session-router** and press Enter.

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

3. Type **session-constraints** and press Enter.

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

If you are adding rate constraints to an existing configuration, then you will need to select the configuration you want to edit.

4. Type **rate-constraints** and press Enter.

```
ORACLE(session-constraints)# rate-constraints
ORACLE(rate-constraints)#
```

5. **method**—Enter the SIP method name for the method you want to throttle. Although the parameter accepts other values, your entries should come only from the from the following list for the feature to function properly:
   - NOTIFY
   - OPTIONS
   - MESSAGE
   - PUBLISH
   - REGISTER
6. **max-inbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
7. **max-outbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
8. **max-inbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
9. **max-outbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
10. Save your changes and apply this session constraint and its rate constraint(s) to SIP interfaces.

Applying Session and Rate Constraints to a SIP Interface

You need the name of the session constraints configuration to apply the restrictions you set up to a SIP interface.

To apply session and rate constraints to a SIP interface:

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

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

2. Type **session-router** and press Enter.

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

3. Type **sip-interface** and press Enter.

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

   If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **constraint-name**—Enter the name of the session constraint configuration where you have set up rate constraints to apply them to this SIP interface. This parameter has no default, and must be the valid name of a session constraint configuration.
5. Save and activate your configuration.

Configuring Rate Constraints for Session Agents

You can also use this feature for individual SIP session agents.

To configure rate constraints for a SIP session agent:

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

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

2. Type **session-router** and press Enter.

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

3. Type **session-agent** and press Enter.

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

   If you are adding rate constraints to an existing configuration, then you will need to select the configuration you want to edit.

4. Type **rate-constraints** and press Enter.

```
ORACLE(session-agent)# rate-constraints
ORACLE(rate-constraints)#
```

5. **method**—Enter the SIP method name for the method you want to throttle. Your entries should come only from the following list:

- NOTIFY
- OPTIONS
- MESSAGE
- PUBLISH
- REGISTER

6. **max-inbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
7. **max-outbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
8. **max-inbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
9. **max-outbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
10. Save and activate your configuration.

Suppressing Re-INVITEs for Call Hold/Resume Dialogs

You can configure the SBC to suppress Re-INVITEs for Call Hold/Resume dialogs and REPLACES dialogs to reduce excess signaling traffic. From the perspective of the SBC, a re-INVITE on one side of a session does not necessarily need to be forwarded to other side. When the SBC receives a Re-INVITE that triggers, for example, a call hold, it can suppress that message from being sent out the egress and handle the transaction locally, between itself and the endstation that sent the re-INVITE.

Within an existing session, SIP allows for re-INVITEs when a station requests a change to dialog parameters, session parameters, or both. To configure this feature, you enable the **suppress-hold-resume-reinvite** on the applicable **realm-config**. The applicable realm depends on your design. You enable **suppress-hold-resume-reinvite** on:

- The ingress realm to suppress hold/resume INVITEs that come from the egress
- The egress realm to suppress hold/resume INVITEs that come from the ingress

In addition, if you enable **suppress-hold-resume-reinvite** on both the ingress and egress realms, the system does not suppress hold/resume re-INVITEs.

When enabled, the SBC targets Re-INVITEs for this suppression feature when it receives a Re-INVITE within specific call flow scenarios:

- Call Hold re-INVITE—The SBC recognizes a hold re-INVITE when it receives an INVITE that is within the existing dialog and includes the SDP media attribute set to inactive or sendonly, or when it receives an INVITE with a connection IP set to 0.0.0.0.
  When it receives a call hold Re-INVITE the SBC:
  - Plays music on hold, assuming you have configured the tone to be played by the SBC during hold.

> **Note:**
>
> This feature does not require MOH.

  - Suppresses this INVITE
  - Responds locally to the station that sent the hold with a 200 OK

  The SBC does not suppress these re-INVITEs when it finds the SDP media attribute set to recvonly. Furthermore, the SBC continues to forward any subsequent re-INVITEs, disabling this re-INVITE suppression feature, after forwarding a recvonly INVITE until the call returns to sendrecv mode. Having returned to sendrecv, the SBC begins to enforce its re-INVITE suppression feature again.

- Call Resume re-INVITE—The SBC recognizes a resume re-INVITE when it receives an INVITE after a call hold re-INVITE that is within the existing dialog. This re-INVITE can be an Offerless re-INVITE, or an INVITE that includes both an SDP media attribute set to sendrecv and the connection IP set to the original IP.
  When it receives a resume Re-INVITE the SBC:
  - Stops MOH
  - Suppresses this INVITE
  - Responds locally to the station that sent the resume with a 200 OK
  - Resumes media support
- re-INVITE with codec change—This scenario can occur when an end-station simply changes codecs. In addition, each of the scenarios above may include a codec change whether or not the ensuing signaling results in transcoding.
- re-INVITE with Replaces header—The SBC suppresses a Re-INVITE that includes the REPLACES header. This scenario is a hold/resume scenario that includes moving the dialog to a different end station.
  When receiving an INVITE with a replaces header, the SBC:
  1. Suppresses this re-INVITE from forwarding out the egress
  2. Replies to the sender with a 200 OK
  3. Supports the new media
  4. Sends a BYE to the original station
- Offerless Re-INVITE—When you configure this feature, the SBC does not forward an offerless re-INVITEs out the egress interface.

Related Configuration

Related configuration considerations include:

- For deployments that need to support dialog replacement, this feature requires that you apply a **sip-profile** with the **replace-dialogs** parameter enabled to the **sip-interface**.
- This feature requires at least one **codec-policy** on either the ingress or the egress realm involved in the call.

You can use a 'dummy' **codec-policy** for this purpose, if desired. A 'dummy' **codec-policy** refers to a **codec-policy** that has no effect, such as the one in the codeblock below, which simply allows all codecs.

```
codec-policy
name Dummy_Policy1
allow-codecs *
```

- This feature takes precedence over the **suppress-reinvite** option available on a **sip-interface**.
- If you configure a **session-timer-profile** on an applicable **sip-interface**, the system generates Re-INVITEs or UPDATEs based on **force-reinvite** parameter.
- This feature does not work in conjunction with TrFO configurations.

Consider the following functional limitations when deploying this feature:

- This feature does not work within REFER scenarios.
- This feature does not perform INVITE suppression on re-INVITEs and replace INVITEs when they contain multiple m-lines.
- If a call is on hold with music on hold playing, an HA switchover causes the MOH to stop.

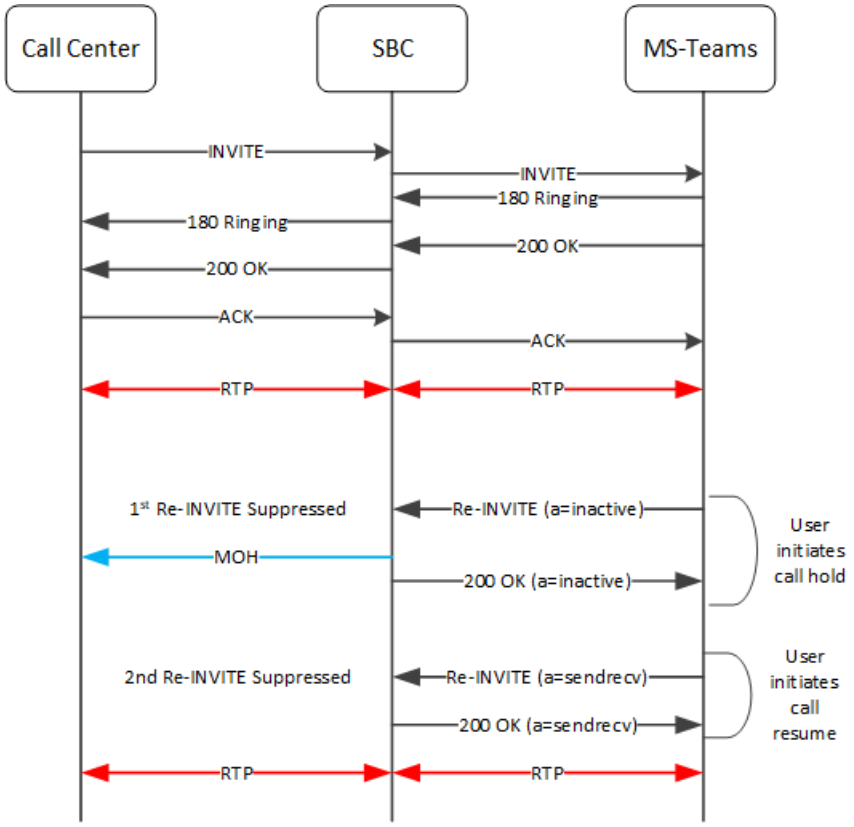
Call Flows for INVITE Suppression

This section provides categories of call flows within which the SBC suppresses INVITEs from transmission out the egress realm to reduce traffic in that realm. The call flows present an MS-Teams environment that initiates and receives calls to a call center and supports a set of applicable operations.

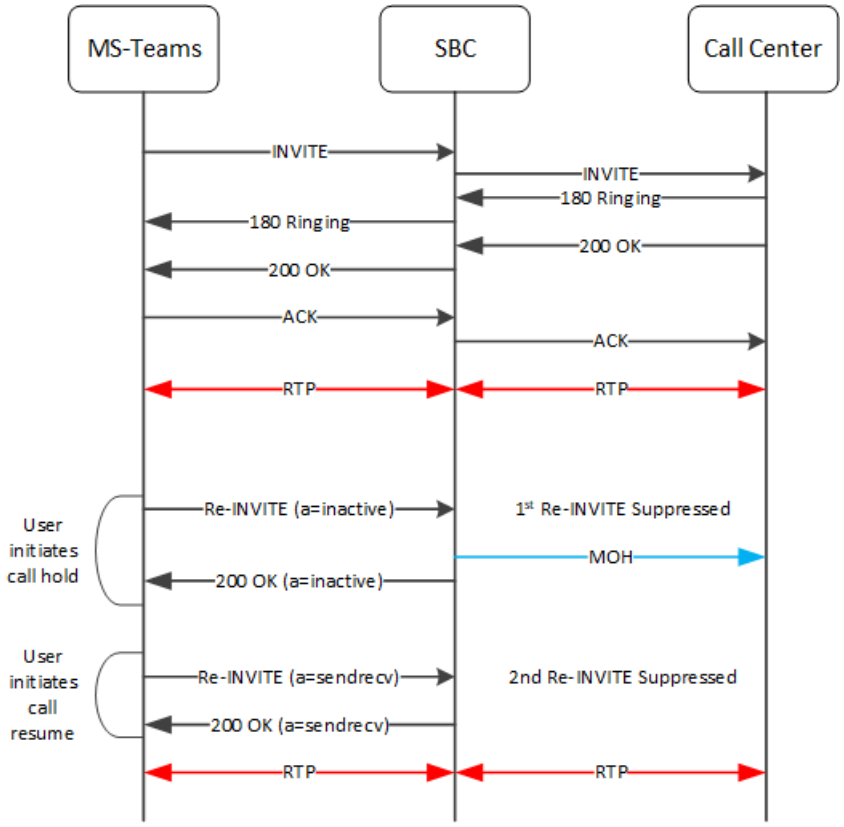
This section displays signaling detail on hold, transcoding, replaces and offerless INVITE operations within which the SBC suppresses INVITEs.

Hold and Resume Call Flows

In this first call flow, the SBC supports a call from the call center to the MS-Teams deployment within which the Teams station temporarily puts a call on hold. The SBC suppresses extraneous INVITEs that occur when the call goes on hold and resumes. In both cases, the SBC responds locally to the MS-Teams station to maintain transaction integrity. In addition, this flow shows that you have configured and properly triggered music on hold to the call center while waiting for the RTP to resume.



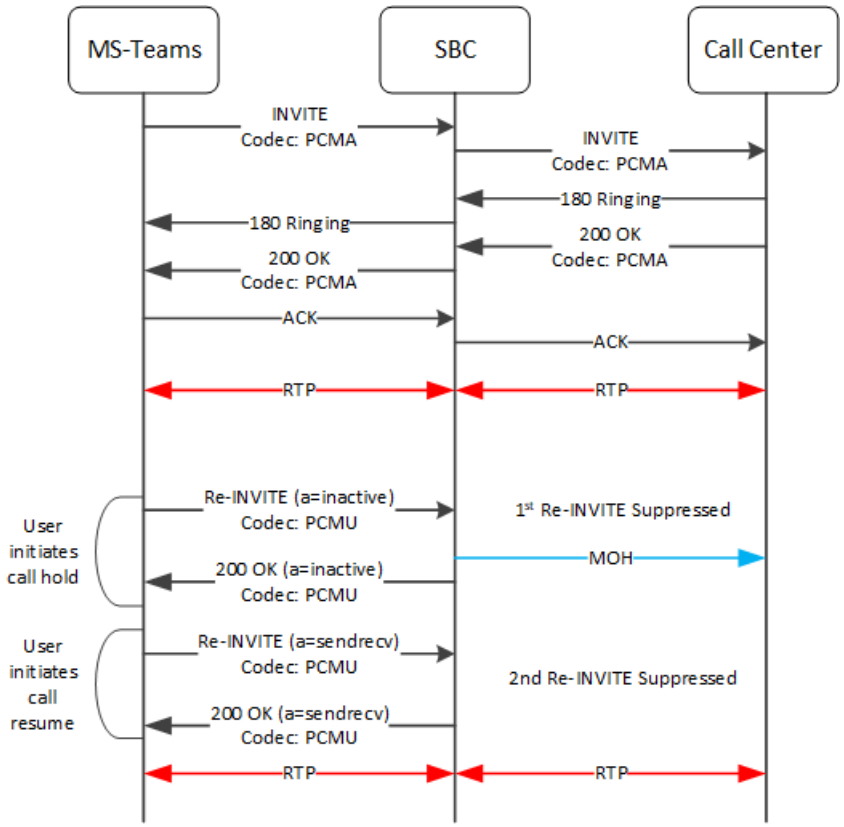
This call flow performs the same function as the first flow with the exception that the call starts on the MS-Teams side. The flow demonstrates the SBC supporting the suppression function on behalf of the caller instead to the callee.



Transcoding

This call flow depicts the SBC performing re-INVITE suppression when an MS-Teams station that has started a call initiates a codec change. The initial call had been proceeding using PCMA on both sides of the call. With the request of a codec change, the SBC changes codec support between itself and the MS-Teams station from PCMA to PCMU. The calling station includes a call hold signal while the codec change takes place, and the SBC issues music on hold during the change.

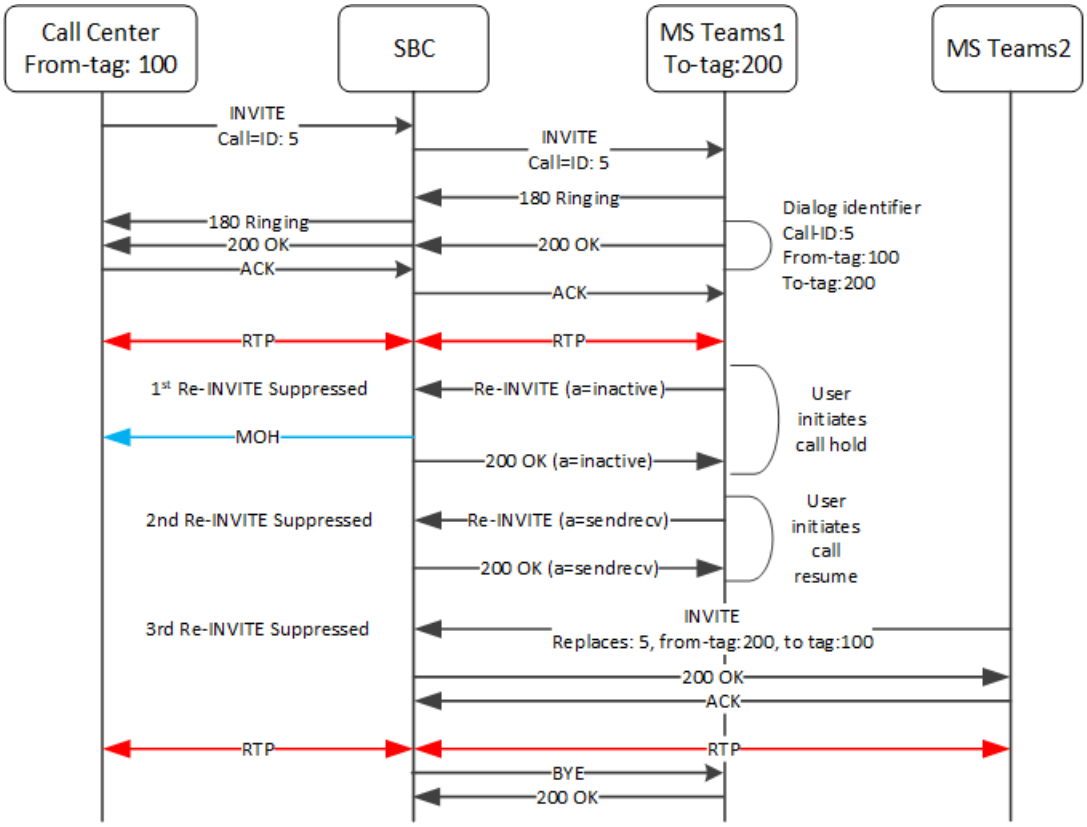
This codec change is not needed on the call center side, so the SBC suppresses the re-INVITE and issues music on hold during the codec change. When the SBC receives a resume re-INVITE, it stops the music on hold and transcodes from PCMU to PCMA transparently to the call center.



REPLACES Call Flow

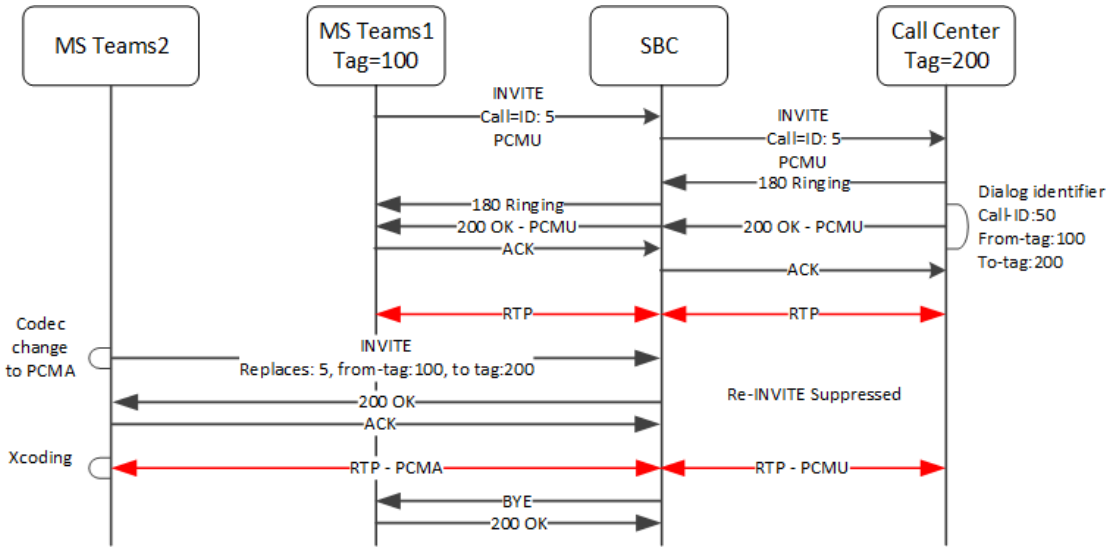
This first replaces flow depicts a call initiated from a call center to an MS Teams endstation. The SBC supports this call setup using its original dialog identifier. The scenario then includes the original MS Teams station replacing the dialog to transfer the call to a second MS Teams station. During the replaces process, the SBC suppresses the INVITEs from the call center, including the hold, resume and replaces INVITEs. The SBC accepts the new dialog to the seconds MS Teams station, and sends a BYE to the first.

After the replaces process, both legs of the SBC maintain different call-id's for the same call.



This next replaces flow depicts a call initiated from a call center to an MS Teams endstation, and includes a codec change resulting in the SBC performing transcoding. In this flow the call center initiates a similar call to the first MS Teams station using PCMU.

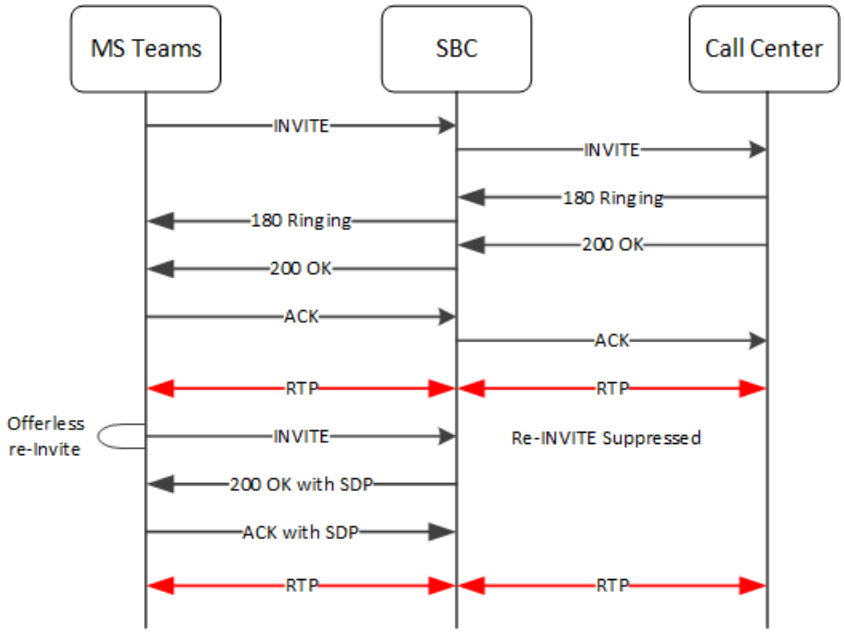
Within the context of the replaces process, however, the second MS Teams station includes a different change. The SBC suppresses this re-INVITE from the call center, accepts the media flow in PCMA. The result is a new media flow using transcoding.



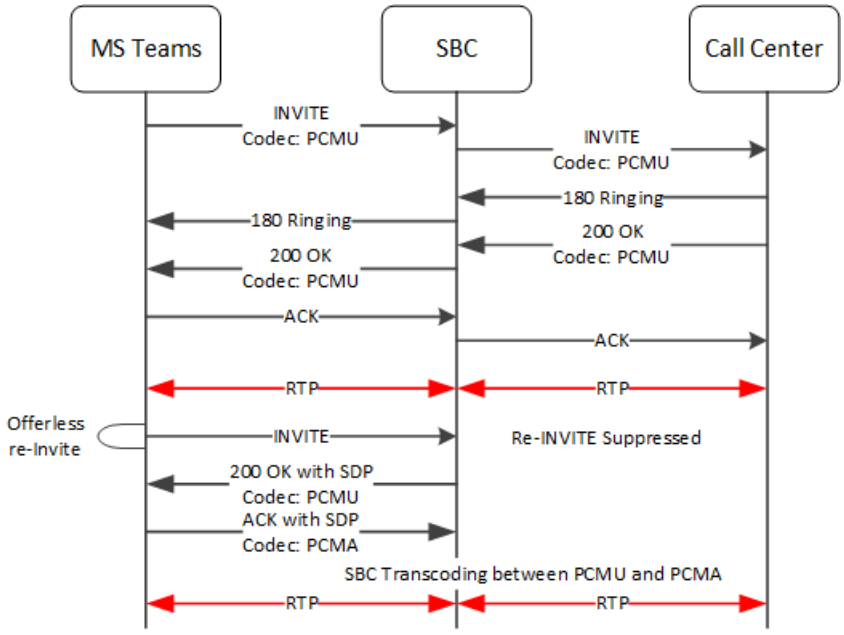
Offerless Re-INVITE Call Flows

In this call flow, an MS Teams station initiates a successful call to a call center. Subsequently, the MS Teams station issues a re-INVITE that does not include SDP. The SBC suppresses this

re-INVITE and responds to the MS Teams station with a 200 OK that includes SDP. MS Teams responds with an ACK and the RTP resumes.



This second offerless INVITE example is similar to the first, with the exception that the final ACK presents a different codec. The SBC responds to this situation by transcoding the PCMA RTP presented by the MS Teams station to PCMU, which was previously used by both legs of the call.



Interaction Between Session Timers and re-INVITE Suppression

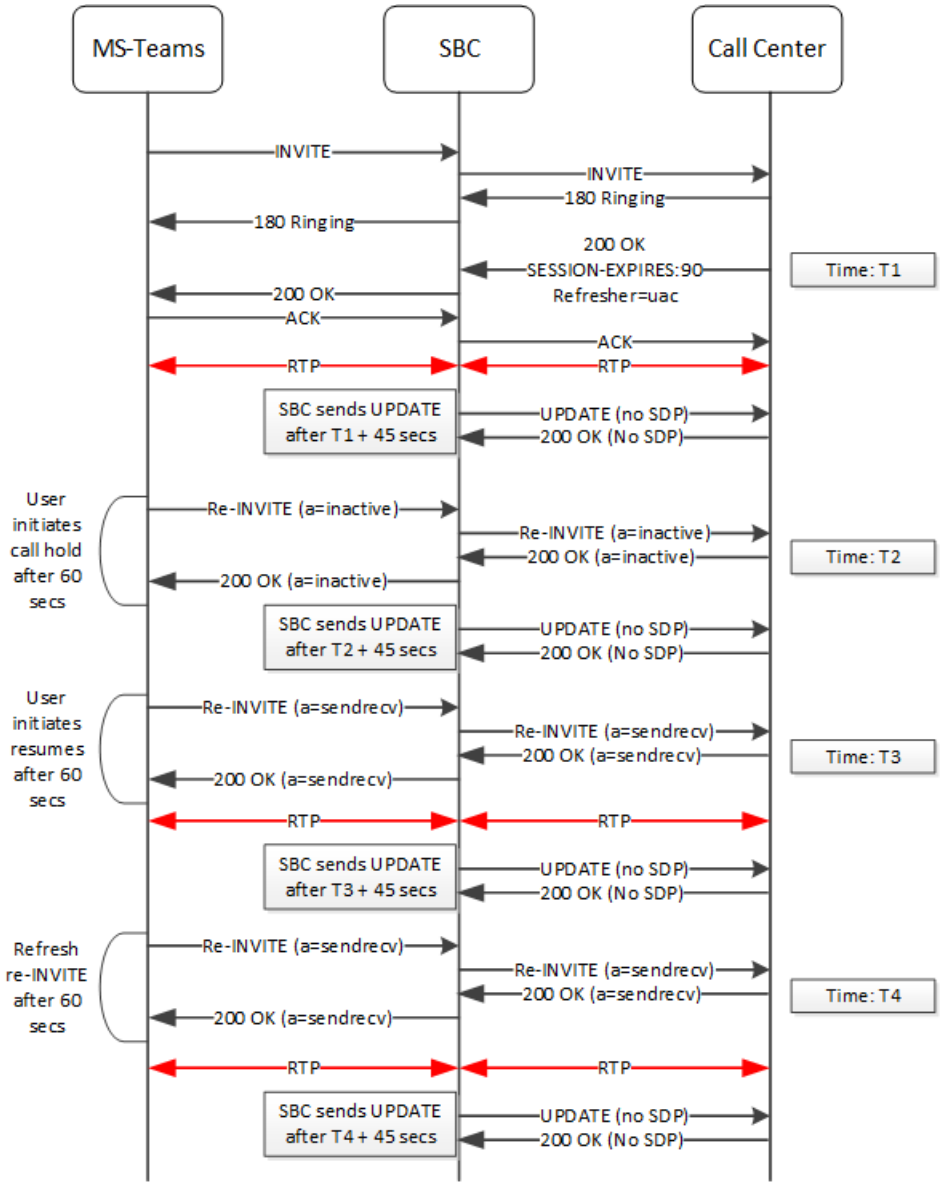
When configured, the SBC performs session timer and re-INVITE suppression operations simultaneously.

This re-INVITE suppression excludes refresh re-INVITEs. Within the context of standard SIP operation, neither an endstation nor the SBC may necessarily be aware of a session ending, for example, without a BYE. SIP operation may use session timer mechanisms within refresh

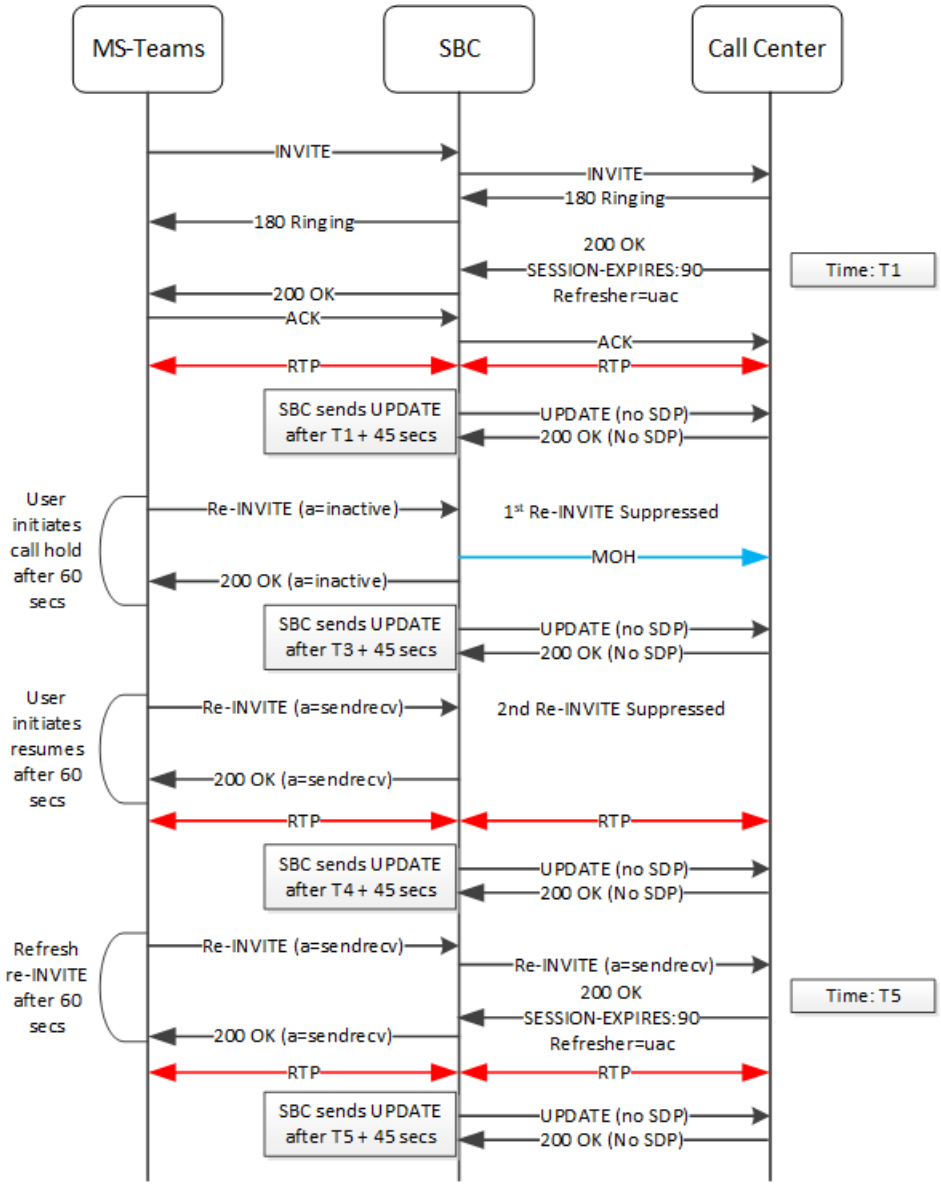
re-INVITEs and UPDATEs to convey expiration or minimal session timing information, which can confirm that a session is still active.

The SBC distinguishes between a re-INVITE that, for example, triggers a call hold and one that performs a refresh re-INVITE triggered by the presence of the Session-Expires parameter. The Session-Expires parameter conveys the lifetime of the session. The Min-SE parameter conveys the minimum allowed value for the session timer. The SBC forwards the latter to support session timing mechanisms unless you have also configured the **session-timer-profile** on the applicable **sip-interface**. This profile allows the SBC to end the session by itself.

The call flow below presents the SBC performing the functions of the **session-timer-profile** below. This profile has its **force-reinvite** parameter set to its default, enabled.



The call flow below presents the SBC performing the functions of the **session-timer-profile** and the the **suppress-hold-resume-reinvite** on the ingress **realm-config** below.



SIP Delayed Media Update

The Oracle Communications Session Border Controller supports SIP delayed media update. When enabled, this feature keeps the Oracle Communications Session Border Controller from updating its media flow information for flows established after an offer-answer exchange. The Oracle Communications Session Border Controller does not update the flow information until a new offer and answer arrive for a specific set of media flows.

The (subsequent) offer does not have to be for the same session; rather, it can appear as a new SIP INVITE that uses the same SDP.

Delayed Media Update Disabled

When this feature is disabled (which is the default behavior), the Oracle Communications Session Border Controller updates media flow entries in its CAM based on signaled SDP when it processes the SDP. If it processes an SDP offer, Oracle Communications Session Border

Controller allocates steering port resources; the Oracle Communications Session Border Controller updates any missing elements for the flow when the answer is returned.

In cases when a secondary offer arrives (either a reINVITE, an UPDATE, or the original INVITE is hairpinned back through the Oracle Communications Session Border Controller), the Oracle Communications Session Border Controller updates the following media flow information at the time of the offer

- Destination IP address
- Destination port
- Realm for the media flows
- Media release settings

This behavior affects specific applications that are better served by the Oracle Communications Session Border Controller waiting to update media flow information until it receives the answer to the second offer.

Delayed Media Update Enabled

When you enable the SIP delayed media update feature, the Oracle Communications Session Border Controller:

- Delays changing the active media flow CAM entry for a new offer if a previous offer and answer have been received for the same media flows; it encodes new SDP information in an outgoing offer, but does not change the CAM entry until the answer is received
- Delays changing the active media flow CAM entry even when the new offer is for a new session
- Supports media release when performing delayed media update changes
- Offers per-realm configuration

This section describes how the delayed media update feature works for hairpinned call flows and for an SDP offer arriving for installed flows.

- Hairpinned call flows—In this type of call flow, the application server (AS) sends an INVITE back to the Oracle Communications Session Border Controller and that INVITE needs to be forwarded to another user (user B). When it receives the offer in this INVITE and delayed media update is disabled, the Oracle Communications Session Border Controller determines that the call is hairpinned and deletes the CAM entry for the flow for user A, who has sent the initial INVITE. The Oracle Communications Session Border Controller deletes the CAM entry for the flow from the AS to user A.
  With delayed media update enabled, the CAM entry for the flow from the AS to user A is not deleted. Instead, the Oracle Communications Session Border Controller waits until it has an answer from user B, and then performs the necessary updates and deletions.
- SDP offer for installed media flows—With delayed media update enabled, if it has received an offer and answer and a new offer arrives for the same flow, the Oracle Communications Session Border Controller delays updating the CAM entries until an answer is received for the new offer.

SIP Delayed Media Update Configuration

You enable this feature on a per-realm basis by setting one parameter.

To enable SIP delayed media update:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

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

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

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **delay-media-update**—Enable keeping the Oracle Communications Session Border Controller from updating its media flow information for flows established after an offer/answer exchange. The default is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

Expedited Call Leg Release for Preempted Hairpin Calls

When hairpinned calls are ended because of signaling failures (such as a SIP mid-dialog signaling timeout, or an H.323 TCP keepalive failure) on one call leg, the Oracle Communications Session Border Controller deletes both legs' media flows simultaneously by default. In addition, when the first hairpinned call leg is torn down, the second call leg is gracefully released immediately by the Oracle Communications Session Border Controller creating and sending an appropriate signaling message (e.g., BYE for a SIP call or ReleaseComplete for an H.323 call) to the endpoint.

You can override this behavior by configuring the **dont-terminate-assoc-legs** option in the media manager. When configured, the orphaned call leg in the hairpin scenario will be torn down after the **initial guard timer expires.** The disconnect times of the two call legs, as recorded by the accounting appliation, will be significantly different, due to the initial guard time for the second call leg.

Accounting Considerations

To indicate cases like this, where the second leg of the hairpinned call was preempted, Oracle Communications Session Border Controller includes the following combination of release/termination causes in the CDR:

```
VSA 49: Acct-Terminate-Cause = NAS_REQUEST
VSA 62: Acme-Disconnect-Cause = 8
```

Supporting Released Flows that are Subsequently Hairpinned back to the SBC

The Oracle Communications Session Border Controller (SBC) allows you to quickly re-establish media flows for released calls that may be hairpinned back the SBC by other network elements.

When the SBC releases a media flow, it also deletes information related to that flow by default. For a call that the environment subsequently hairpins back to the SBC, there is a delay in determining how to forward it. You can configure the SBC to retain IP/port information for all released calls to cause it to quickly process and forward these calls. To do this, you configure the **media-manager** with the **hairpin-released-flows** option.

```
ORACLE(media-manager)# options +hairpin-released-flows
```

This configuration makes the SBC store the IP/port information of the released media port sent out in the SDP. Should a proxy or media server send the media connection with that IP/port back to the SBC, the SBC recognizes the leg and performs the proper linking between the flows.

Recall that you determine whether the SBC releases or manages media using the **mm-in-realm**, **mm-in-network**, **mm-same-ip** and **mm-in-system** settings in the realm configuration.

SIPconnect

The Oracle Communications Session Border Controller supports the SIPconnect model, wherein PBXs register themselves so that service providers do not need to know IP addresses or locations in advance for static configurations. This is particularly helpful when the PBX is behind a NAT.

In the PBX registration process, the PBX creates a binding between one of its phone numbers as the address of record (AoR) and Contact-URI in the REGISTER message. The registrar knows that the single AoR actually represents many addresses, and so it registers them implicitly. However, the registrar does not return the implicit AoR number in P-Associated-URIs.

The SIPconnect feature resolves the following issues that arise from using this model:

- SIP INVITEs sent to the PBX from the Registrar through the Oracle Communications Session Border Controller have the Request-URI of registered contact. Because it typically ignores the To-URI, the PBX needs the Request-URI username portion to be the specific extension number being called.
  With the SIP connect feature enabled, the Oracle Communications Session Border Controller overwrites the Request-URI username with the To-URI username.
- SIP INVITEs from the PBX have the From AoR and Contact-URI usernames of specific phones rather than of the registered AoR and Contact-URI. For the Oracle Communications Session Border Controller, this means that it cannot use the **allow-anonymous** parameter value of register; there would be no registered user matches, and the Oracle Communications Session Border Controller would reject them (with a 403 Forbidden).
  With the SIP connect feature enabled, the Oracle Communications Session Border Controller performs allow-anonymous checking based on the registered Via address, which is the same for all requests for the same PBX.

Modifications to Registration Caching Behavior

With the SIP connect feature enabled, Oracle Communications Session Border Controller registration caching works the same way that it does with the feature disabled, with the following exceptions:

The Oracle Communications Session Border Controller determines whether the destination realm has the sip-connect-pbx-reg option configured, and then:

- If it is configured, the Oracle Communications Session Border Controller replaces the user part of the Request-URI with the user part of the To header. When the INVITE contains a P-Called-Party-ID header, the Oracle Communications Session Border Controlleruses the user part of the P-Called-Party-ID header (instead of the To header).
- If it is not configured, the Oracle Communications Session Border Controller determines if the destination address is for a session agent and whether that session agent has sip-connect-pbx-reg option configured. When it is configured, the Oracle Communications Session Border Controller performs the same replacements described in the bullet directly above. When it is not configured, the Oracle Communications Session Border Controller does not make any replacements.

When it receives an INVITE request, the Oracle Communications Session Border Controller checks the incoming realm for the sip-connect-pbx-reg option.

- If it is configured, the Oracle Communications Session Border ControllerC uses the INVITE's source address (instead of the AoR and Contact-URI) to search the registration cache for a matched registration entry.
- If it is not configured, the Oracle Communications Session Border Controller determines if the INVITE's source address is for a session agent and whether that session agent has sip-connect-pbx-reg option configured.
  When it is configured, the Oracle Communications Session Border Controller replaces the user part of the Request-URI with the user part of the To header. When the INVITE contains a P-Called-Party-ID header, the Oracle Communications Session Border Controlleruses the user part of the P-Called-Party-ID header (instead of the To header).

  When it is not configured, the Oracle Communications Session Border Controller does not make any replacements.

Configuring SIP Connect Support

You configure this feature by adding the sip-connect-pbx-reg option to the realm configuration. In addition, though this feature requires that your configuration also be set up as outlined in this section. The first two items are required, and Oracle recommends that you also implement the suggested additional configuration.

Required Configuration

- Registration caching is enabled.
- For the realm from which registrations come, the options list must include sip-connect-pbx-reg; this is new configuration introduced to support this feature. The presence of this option instructs the Oracle Communications Session Border Controller to skip matching the Contact header in the INVITE request with the registered Contact of the registration entry. The Oracle Communications Session Border Controller finds a registration using only the INVITE's source address.

Alternatively, you can configure the sip-connect-pbx-reg option in the options list for a session agent. When the realm where an INVITE comes from does not have this option set, the Oracle Communications Session Border Controller determines whether or not the INVITE came from a session agent. You might choose to configure session agents with this option if you do not want it applied to an entire realm. If the PBX is behind a NAT device, the session agent's IP address for the PBX (if statically configured) must be the IP address of the NAT device. And if DNS is use, the session agent's hostname must resolve to the NAT device's IP address.

Suggested Additional Configuration

- In the SIP ports configuration (accessed through the SIP interface configuration), the **allow-anonymous** parameter must be set to registered. This setting allows the Oracle Communications Session Border Controller to accept SIP requests from session agents and registered endpoints only, but to accept REGISTER requests from any endpoint.
- For the SIP interface that accepts registrations, the **options** parameter must be set to reg-via-key. This setting allows the Oracle Communications Session Border Controller to use the source address of an INVITE as the key to find a registration entry in the registration cache. When the INVITE's Contact header matches the registered Contact in the registration entry, the Oracle Communications Session Border Controller accepts the INVITE request.

SIP Connect Configuration

To set the SIP connect option for a realm configuration:

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

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

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

   ```
   ORACLE(media-manager)# realm-config
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **sip-connect-pbx-reg** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(realm-config)# options +sip-connect-pbx-reg
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

   To set the SIP connect option for a SIP session agent configuration:

6. In Superuser mode, type **configure terminal** and press Enter.

```
Oracle Communications Session Border Controller# configure terminal
```

7. Type **session-router** and press Enter to access the signaling-related configurations.

```
Oracle Communications Session Border Controller(configure)# session-router
```

8. Type **session-agent** and press Enter.

```
ORACLE(session-router)# session-agent
```

   If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the session agent that you want to edit.

9. **options**—Set the options parameter by typing **options**, a Space, the option name **sip-connect-pbx-reg** with a plus sign in front of it, and then press Enter.

```
Oracle Communications Session Border Controller(session-agent)# options
+sip-connect-pbx-reg
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the session agent's configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

10. Save and activate your configuration.

SIP Registration Event Package Support

Certain endpoints subscribe to the Registration Event Package, RFC 3680, which defines how SIP user agents can request and obtain notifications about registration events. Previously, the Oracle Communications Session Border Controller (SBC) passed the Subscribe and Notify messages of this package transparently, without modifying the XML bodies of either. However, in many cases the XML body can contain IP addresses, contact URIs, and expires times that the SBC needs to modify for proper operation. This new feature enables the SBC to modify correctly the XML body for the Registration Event Package.

In addition to resolving this type of issue, enabling registration event package support on your system provides the functions described below:

- The SBC performs NAT on all contacts in the reginfo, regardless of their state.
- The SBC performs NAT on the address of record (AoR) attribute of the Registration element when it matches an existing cache entry. When either the Contact-URI or the AoR does not match a cache entry and the host part of the URI is an IP address, the SBC will NAT the host part using the applicable SIP NAT configuration
- Contacts are found in the XML URI element for the contact. But if there is no URI element, then the SBC uses the Contact element information for the contact.
- If the expires attribute in the Contact element is a value other than zero, the SBC uses (inserts) the expires values from the registration cache.
- This feature also introduces delayed deletion from the registry cache. When a 200 OK comes back in response to a REGISTER message and the 200 OK does not include all previously registered contacts, the missing contacts are deleted. If the global SIP

configuration option contact_cache_linger=XX (where XX is the number of seconds to wait before deleting), then the contacts to be deleted remain for the specified number of seconds before they in fact are deleted.

The SBC provides statistics counters for the number of Reg-Event subscriptions it manages via the ACLI, SNMP and HDR.

Updating Expiration Values

This feature also supports updating the expiration values for the registration cache when a Contact element has the expires attribute. For this support, the following apply:

- If the value of the expires attribute is greater than the expiration value for the access-side registration cache entry, the Oracle Communications Session Border Controller replaces the XML expires attribute value with the cached one from the access side.
- If the value of the XML expires attribute is less than the core-side expiration value for the core-side registration cache entry, the Oracle Communications Session Border Controller updates the core-side expiration value with the value from the expires attribute. Further, the Oracle Communications Session Border Controller adjusts the access-side expiration value of the registration cache in these ways:
  - If the value of the XML expires attribute is less than the current access-side expiration value for the registration cache entry, the Oracle Communications Session Border Controller sets the access-side expiration value to be equal to the value in the expires attribute.
  - Otherwise, the Oracle Communications Session Border Controller leaves the expires value for the access-side expiration value for the registration cache entry unchanged. If this happens, the Oracle Communications Session Border Controller replaces the value of the XML expires attribute with the adjusted access-side expiration value.
- If the expires attribute from a Contact element is 0 (meaning that the core is removing the registration), the Oracle Communications Session Border Controller removes that Contact-URI from its registration cache. And if the registration cache entry has no remaining Contact-URIs, the Oracle Communications Session Border Controller deletes the registration cache entry altogether.

Contact Cache Linger Configuration

You enable this feature as part of the global SIP configuration, using that configuration's **options** parameter. You can optionally configure the number of seconds you want to keep a contact in the registration cache before it is deleted. This is the option:

- **contact-cache-linger=XX**—Number of seconds to wait before a contact is deleted from the cache (where XX is the number of seconds)
  To enable SIP Registration overload protection on your Oracle Communications Session Border Controller:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**contact-cache-linger=XX**) where XX is the number of seconds to keep a contact in the cache before deleting it.

```
ORACLE(sip-config)# options +contact-cache-linger=5
```

If you type either of these options without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

SIP Event Package for Registrations

This feature enables the Oracle Communications Session Border Controller, acting as a Proxy Call Session Control Function (P-CSCF) to initiate subscription to the SIP Event Package for Registrations.

Applicable Standards

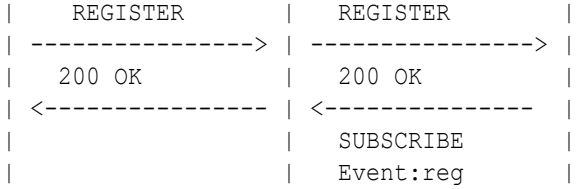
RFC 3265, *Session Initiation Protocol (SIP)-Specific Event Notification*, outlines a framework within which a SIP node, designated as the subscriber, can request automatic, asynchronous notification of certain events from a remote peer, designated as the notifier. RFC 3265 also defines an Event Package as a set of state information to be reported by a notifier to a subscriber, and mandates that such Event Packages be described in a specific RFC explicitly identifying the state information to be reported by the notifier and defining required syntax and semantics to support the subscription/notification exchange.

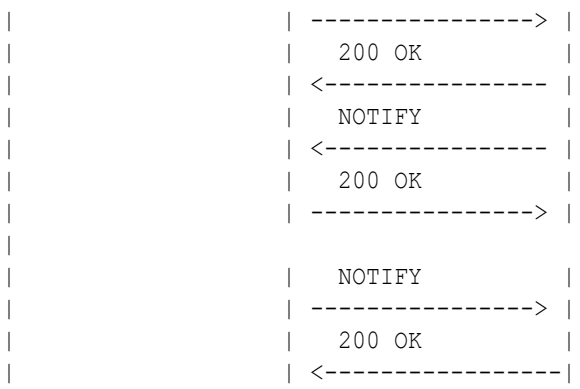
RFC 3680, *A Session Initiation Protocol (SIP) Event Package for Registrations*, fulfills this requirement by defining a method that enables SIP user agents to request a defined set of state information from a SIP Registrar.

Section 5.2.3 of the 3GPP (Third Generation Partnership Project) 24.229, *IP Multimedia Call Control Protocol Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)*; Stage 3, mandates that a P-CSCF subscribe to the SIP Event Package for Registrations as defined in RFC 3680.

Call Flow

An example call flow between a SIP endpoint, the Oracle Communications Session Border Controller, acting as a P-CSCF, and a SIP Registrar (S-CSCF) illustrates the Subscription/ Notification process.





The first two messages (the REGISTER request and the 200 REGISTER response) accomplish the successful registration of the SIP endpoint.

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP pc34.example.com;branch=z9hG4bKnaaff
From: sip:joe@example.com;tag=99a8s
To: sip:joe@example.com
Call-ID: 88askjda9@pc34.example.com
CSeq: 9976 REGISTER
Contact: sip:joe@pc34.example.com
```

Immediately after processing the initial 200 OK from the SIP Registrar, the Oracle Communications Session Border Controller sends a SUBSCRIBE Request to the S-CSCF.

```
SUBSCRIBE sip:joe@example.com SIP/2.0
Via: SIP/2.0/UDP app.example.com;branch=z9hG4bKnashds7
From: sip:sd.example.com;tag=123aa9
To: sip:joe@example.com
Call-ID: 9987@app.example.com
CSeq: 9887 SUBSCRIBE
Contact: sip:joe@pc34.example.com
P-Asserted-Identity: <sip:sd@example.com>
Event: reg
Max-Forwards: 70
Accept: application/reginfo+xml
```

The Request URI and To header contain the address-of-record (sip:joe@example.com) of the subscription subject. This value was previously contained in the From and To headers of the of the original REGISTER request. These fields are always identical in REGISTER requests, except in the case of third-party registration.

The From and P-Asserted-Identity headers contain the identity of the subscription requester.

The Contact header contains an IP address or FQDN at which the subscription subject can be reached. This field duplicates the value of the Contact header in the original REGISTER request. Multiple contacts can be registered for a single address-of-record.

The Event header contains the required value, reg, which identifies the requested Event Package subscription. reg specifies the SIP Event Package for Registrations.

The Accept header contains the required, default value, application/reginfo+xml, indicating syntactical support for Registration notifications and attached XML notification bodies.

Assuming the S-CSCF accepts the subscription, it responds with a 200 SUBSCRIBE response.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP app.example.com;branch=z9hG4bKnashds7
;received=192.0.2.1
From: sip:sd.example.com;tag=123aa9
To: sip:joe@example.com;tag=xyzygg
Call-ID: 9987@app.example.com
CSeq: 9987 SUBSCRIBE
Contact: sip: joe@pc34.example.com
Expires: 3600
```

The From header contains the identity of the subscription requester.

The To header contains the address-of-record of the subscription subject.

The Contact header contains an IP address or FQDN at which the subscription subject can be reached.

The Expires header contains the subscription duration in seconds. Upon receipt of a 2xx response to the SUBSCRIBE request, the Oracle Communications Session Border Controller stores the information for the established dialog and the expiration time. If continued subscription is required, the Oracle Communications Session Border Controller automatically refreshes the subscription to the SIP Event Package for Registrations, either 600 seconds before the expiration time if the initial subscription was for greater than 1200 seconds, or when half of the time has expired if the initial subscription was for 1200 seconds or less.

Following the 200 SUBSCRIBE response, the S-CSCF generates an initial notification, with Event Package state information contained in an XML body.

```
NOTIFY sip:app.example.com SIP/2.0
Via: SIP/2.0/UDP server19.example.com;branch=z9hG4bKnasaij
From: sip:app@example.com;tag=xyzygg
To: sip:sd.example.com;tag=123aa9
Call-ID: 9987@app.example.com
CSeq: 1289 NOTIFY
Contact: sip:server19.example.com
Event: reg
Max-Forwards: 70
Content-Type: application/reginfo+xml
Content-Length: ...

<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="1"
state="partial">
    <registration aor="sip:joe@example.com" id="a7" state="active">
        <contact id="76" state="init" event="registered"
        duration-registered="0">
            <uri>sip:joe@pc34.example.com</uri>
        </contact>
    </registration>
</reginfo>
```

Notification Bodies

Registration state changes are reported in XML attachments to NOTIFICATIONS generated by the S-CSCF. As shown above, the XML consists of one or more registration elements that report the state of a specific address-of-record. Attributes supported by the registration element are as follows:

aor contains the address-of-record

id identifies this specific registration

state reports the Registration state — init, active, or terminated

init — address-of-record not yet cached

active — address-of-record maintained in current cache

terminated — address-of-record removed from cache, not currently valid

registration elements, in turn, contain one or more child contact elements. Attributes supported by the contact element are as follows>

id identifies this specific contact

state reports the Contact state — active or terminated

event reports the event that generated the last state change — registered, created, refreshed, shortened, expired, deactivated, probation, unregistered, or rejected

duration-registered reports the length (in seconds) of the current registration

contact elements, contain a single child uri element that identifies the contact address of FQDN.

SIP Event Package for Registrations Configuration

Subscription to the SIP Event Package for Registrations is enabled at the SIP interface level.

1. Use the following command sequence to move to **sip-interface** Configuration Mode.

   ORACLE# **configure terminal**

   ORACLE(configure)# **session-router**

   ORACLE(session-router)# **sip-interface**

   ORACLE(sip-interface)#

2. Use the **subscribe-reg-event** parameter to subscribe to the SIP Event Package for Registrations.

   By default, subscription is disabled.

   ORACLE(sip-interface)# **subscribe-reg-event enabled**

   ORACLE(sip-interface)#

3. Use **done**, **exit**, and **verify-config** to complete enabling the Event Package subscription.

SIP Transport Selection

With this feature enabled, when the Oracle Communications Session Border Controller forwards a message larger than the value specified in the maximum UDP length parameter, it attempts to open on outgoing TCP connection to do so. This connection might fail for a number of reasons; for example, an endpoint might not support UDP, or it might be behind a firewall. The UDP fallback option addresses this condition. If it is configured in SIP interfaces associated with an outgoing message and a TCP session cannot be established, the Oracle Communications Session Border Controller falls back to UDP and transmits the message. When the option is not present, the Oracle Communications Session Border Controller's default behavior is to return the SIP status message 513 Message too Large.

SIP Transport Selection Configuration

You enable this feature per SIP interface by setting options that control the maximum UDP length and allow UDP fallback:

- max-udp-length=X (where X is the maximum length)—Sets the largest UDP packers that the Oracle Communications Session Border Controller will pass. Packets exceeding this length trigger the establishment of an outgoing TCP session to deliver the packet; this margin is defined in RFC 3261. The system default for the maximum UDP packet length is 1500.
  You can set the global SIP configuration's max-udp-length=X option for global use in your SIP configuration, or you can override it on a per-interface basis by configuring this option in a SIP interface configuration.
- udp-fallback—When a request needs to be sent out on the SIP interface for which you have configured this option, the Oracle Communications Session Border Controller first tries to send it over TCP. If the SIP endpoint does not support TCP, however, then the Oracle Communications Session Border Controller falls back to UDP and tries the request again.
  To enable SIP Transport Selection:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **max-udp-length=X** (where X is the maximum UDP length you want to set), and then press Enter.

   ```
   ORACLE(sip-interface)# options +max-udp-length=900
   ```

If you type options max-udp-length=X, you will overwrite any previously configured options. In order to append the new option to the sip-interface's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. **options**—Set the options parameter by typing **options**, a Space, the option name **udp-fallback**, and then press Enter.

```
ORACLE(sip-interface)# options +udp-fallback
```

If you type options udp-fallback, you will overwrite any previously configured options. In order to append the new option to the sip-interface's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

uaCSTA NAT Support

The Oracle Communications Session Border Controller offers User Agent Computer Supported Telecommunications Application (uaCSTA) support, which allows for the network address translation (NAT) of a key XML element in SIP INFO messages to use a phone's real contact URI.

Overview

Certain customers who use a uaCSTA for third party call control have encountered difficulties with the XML in their SIP messages used to support business applications. In these cases, the XML—specifically the <deviceID> XML tag—carries encoded IP addresses that need to be changed as they traverse the Oracle Communications Session Border Controller.

The SIP business application allows users to click-to-dial another party using e-mail application clients. The user's click triggers the application server to send a uaCSTA SIP INFO message through the system to the UA/phone. These SIP INFO messages contain XML with the user's Contact-URI. But the server is only aware of the Oracle Communications Session Border Controller's NAT'd Contact-URI and not the user's, so the XML in the SIP INFO is carrying incorrect information.

The XML element, then, needs to be NAT'd to the phone's real Contact-URI. This is especially important because of the broad use of SIP INFO messages, which instruct a phone to:

- Answer a call
- Hold a call
- Retrieve a call

All of these functions are available via a clickable interface on the e-mail application.

The Oracle Communications Session Border Controller performs the NAT to the <deviceID> XML tag only if it is configured to perform registration caching.

When the Oracle Communications Session Border Controller receives a SIP message from the core side and the request has:

- A Content-Type of application/csta+xml
- A Content-Length greater than 0

it parses the message's message body into an XML document. Should parsing fail, then the Oracle Communications Session Border Controller will forward the SIP INFO request without modification to the XML message body. Otherwise, the Oracle Communications Session

Border Controller searches for the <deviceID> subelement within the XML document. If it finds the <deviceID> subelement, the Oracle Communications Session Border Controller searches through its registration cache for a registered Contact that matches the value of the <deviceID>. If it finds a match, the Oracle Communications Session Border Controller replaces the value of the <deviceID> with that of the corresponding registered Contact. If the value of the <deviceID> is a Contact that the Oracle Communications Session Border Controller generates for a registered UA, the corresponding contact from the look-up would be the Contact of the registered UA.

These functions performed, the Oracle Communications Session Border Controller then reformats the SIP INFO request with the modified XML message body before sending it to the next hop. If there is no match found, then the Oracle Communications Session Border Controller forwards the SIP INFO request without modifying the XML message body.

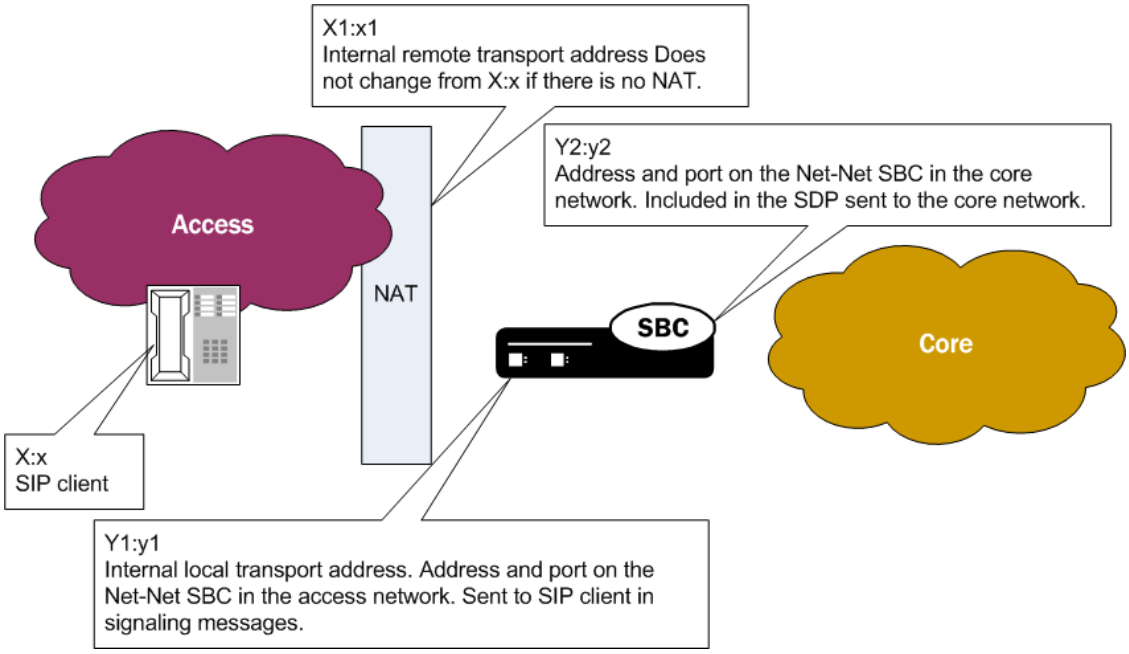
Other than ensuring your Oracle Communications Session Border Controller is configured to perform registration caching, you do not need take any further steps.

SIP Packet Cable Multi-Media

As a packet cable multi-media (PCMM) enhancement for SIP sessions key to next generation architectures, the Oracle Communications Session Border Controller can now include certain SDP attributes specifying media flow addresses in outgoing SIP messages. Previously, these address were hidden by the Oracle Communications Session Border Controller. Since SIP proxies and application servers in the core network, however, need to know these addresses to guarantee QoS for media flows in packet cable networks.

Certain options in the SIP interface configuration enable the Oracle Communications Session Border Controller to reveal address information on the core side.

When a SIP client in the access network sends and receives RTP media, the Oracle Communications Session Border Controller uses the SIP client's IP address and port (X:x) as its own internal remote transport address. The Oracle Communications Session Border Controller adds this information to outgoing SDP that it sends to the core side, and removes it from incoming SDP. If the SIP client sits behind a NAT, then the Oracle Communications Session Border Controller uses the IP address and port produced from the NAT (X1:x1) process for insertion and removal. The SIP client sends RTP to an IP address and port (Y1:y1) on the Oracle Communications Session Border Controller, referred to as the internal local transport address; this information is included in SDP (included in SIP messages) sent to the SIP client. Meanwhile, the Oracle Communications Session Border Controller also has an IP address and port (Y2:y2) in the core network. The far-end SIP UA sends RTP to this IP address and port, which are also included in SDP the Oracle Communications Session Border Controller sends to the core side.



To enforce QoS properly on the access side, the flow between the SIP client (or the SIP client's post-NAT IP address and port) and the internal local address must be revealed on the core side using SIP signaling messages.

Details

To enable this enhancement, you set three parameters in the SIP interface configuration:

- **sdp-internals**—Establishes that local and remote transport addresses need to be added.This option must be enabled on the access-side SIP interface, which is where the Oracle Communications Session Border Controller receives SDP.
- **sdp-local=<name>**—Sets a name for the internal local transport port address that the Oracle Communications Session Border Controller inserts into outgoing SDP. This option is configured on the core-side SIP interface. This address is removed from incoming SDP from the core side to prevent attributes from being sent back to the core in a hairpinned call.
- **sdp-remote=<name>**—Sets a name for the internal remote transport address that the Oracle Communications Session Border Controller inserts into outgoing SDP. This option is also configured on the core-side SIP interface. This address is also removed from incoming SDP from the core side to prevent attributes from being sent back to the core in a hairpinned call.

Further, the Oracle Communications Session Border Controller determines whether or not to insert the SDP attributes based on a call's ingress and egress signaling realms:

Address Information	Calling-Side SDP	Called-Side SDP
Internal local transport address	Added to SDP when: The ingress signaling realm's SIP interface has the sdp-internals option configured The egress signaling realm's SIP interface has a defined sdp-local option	Added to SDP when: The egress signaling realm's SIP interface has the sdp-internals option configured The ingress signaling realm's SIP interface has a defined sdp-local option
Internal remote transport address	Added to SDP when: The ingress signaling realm's SIP interface has the sdp-internals option configured The egress signaling realm's SIP interface has a defined sdp-remote option	Added to SDP when: The egress signaling realm's SIP interface has the sdp-internals option configured The ingress signaling realm's SIP interface has a defined sdp-remote option

Core-Side SDP Insertion Configuration

In a typical configuration intended to send SDP to the core side with the inserted attributes, the access SIP interfaces have the **sdp-internals** option enabled, and the core SIP interfaces have the **sdp-local** and **sdp-remote** values configured.

To set the access SIP interface for SDP insertion on the core side:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-interface)#
   ```

   If you are adding support for this feature to a pre-existing SIP configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **sdp-internals** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +sdp-internals
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

To set the local and remote transport addresses for a core SIP interface:

6. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

7. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

8. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-config
ORACLE(sip-interface)#
```

9. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**sdp-local=<name>**, where the name is attribute name for the SDP), and then press Enter. Follow the same steps to add the **sdp-remote** option.

```
ORACLE(sip-interface)# options +sdp-local=Local_Turn
ORACLE(sip-interface)# options +sdp-remote=PCMM_USERADD
```

If you type either of these options without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

10. Save and activate your configuration.

SIP Method-Transaction Statistic Enhancements

In prior releases, the Oracle Communications Session Border Controller tracks SIP session agents, SIP interfaces and SIP realms on a global level. Only counters that are related to session rates and constraints are displayed.

You can now enable your Oracle Communications Session Border Controller to track transaction messages for specific SIP session agents, SIP realms, and SIP interfaces.

The following SIP methods are tracked for Recent, Total, and Period Max values:

- INVITE | ACK | BYE | REGISTER | CANCEL | PRACK | OPTIONS | INFO | SUBSCRIBE | NOTIFY | REFER | UPDATE | MESSAGE | PUBLISH | other (unknown)

With this new tracking enhancement, the **show sipd** command has been updated with a new method argument which allows you to query statistics for a particular method for a given SIP agent, SIP interface, or SIP realm.

SIP Method Tracking Enhancements Configuration

To enable or disable the expanded SIP Method statistics tracking:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
```

4. **extra-method-stats**—Enable this parameter if you want to use the expanded SIP Method tracking feature. The default is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

National Security and Emergency Preparedness for SIP

The Oracle Communications Session Border Controller (SBC) supports Emergency Telecommunications Service (ETS), which gives priority treatment of National Security and Emergency Preparedness (NSEP) communications for IP network infrastructures. ETS can increase the likelihood that calls, sessions, and other communications will be successfully completed when they are initiated by government-authorized users over the public network infrastructure. Legacy circuit-switched services such as Government Emergency Telecommunications Service (GETS) and Wireless Priority Service (WPS) also fall under the ETS rubric, and are now also supported on the SBC.

To provide this support, you can enable the SBC to act on SIP calls that contain an ETS dial number (DN) and/or the SIP Resource-Priority header that carries ETS resource values.

The SBC identifies ETS calls by using the system's network management controls (NMC) functionality. With NMC and Resource-Priority header (RPH) support enabled on your system, the SBC detects ETS calls and provides the appropriate treatment for them.

The SBC supports this feature by treating ETS calls based on the r-value parameter in the Resource-Priority header. The r-value is a key piece of information because it defines the resource priority that the call originator requests. The r-value parameter provides namespaces and priorities that the SBC can manipulate in outgoing traffic.

The system also allows you to specify a percentage of total session capacity that you reserve for NSEP sessions. When you do this, you extend upon the system's prioritization of NSEP sessions by ensuring transport of emergency sessions even when the system is overloaded with standard traffic. The system provides checks on what you can reserve and reports on its use of this reservation behavior.

An RPH profile is applied to an NMC rule to specify r-values, a media policy to use, and what type of call treatment to apply. Although this also applies to an NMC rule, the RPH policy provides information about which r-values to insert and which to override.

Matching by NMC and by RPH

When a Oracle Communications Session Border Controller has been enabled to act on RPH, it checks incoming requests for RPH, tries to parse that RPH, and then rejects requests in the circumstances listed below. For all of these rejections, the Oracle Communications Session Border Controller logs the error at the TRACE level.

- Request with multiple instances of the same namespace in the RPH—The Oracle Communications Session Border Controller sends out a 400 Bad Request response with

the Invalid RPH - Namespace repeated header showing that there are multiple instances of the same namespace in the RPH.

- Request with invalid resource priority for a namespace—The Oracle Communications Session Border Controller sends out a 400 Bad Request response with the Invalid RPH - Invalid rvalue: x showing that there is an invalid resource value (where x is the invalid value).
- Request with WPS namespace, but without ETS namespace—The Oracle Communications Session Border Controller sends out a 400 Bad Request response with the Invalid RPH - No ETS value header showing that there is no ETS namespace.

If the Oracle Communications Session Border Controller successfully parses the RPH, it identifies the ETS call by checking the Request-URI of the incoming request against destination identifiers that you configure in the NMC rules. If there is a match between the request's ETS DN and the destination value identifier in the NMC rules, the Oracle Communications Session Border Controller tags the call; note that NMC rules need to be configured with the **rph-feature** parameter set to enabled to identify an ETS call properly. If there is no match to an NMC rule, then the Oracle Communications Session Border Controller performs matching based on RPH by comparing resource values (r-values) in the RPH with values you set in the RPH profile configuration.

For an ETS call that matches by ETS DN and NMC rule, the system checks the NMC rule to determine if it has an RPH profile (with r-values) assigned to it. If so, the Oracle Communications Session Border Controller continues by comparing the RPH profile's r-values against those in the request's RPH. In cases where the RPH does not contain a recognized value r-value, the Oracle Communications Session Border Controller:

- Processes the call as it normally would (as a non-ETS call) without changing the RPH if the resource-priority option tag is not present in the Required header (for an INVITE only and not any other requests or response from which RPH would be deleted)
- Rejects the Request when the Require header has the resource-priority header; or, inserts an Accept-Resource-Priority header (ARPH) in the response if the **insert-arp-header** parameter option is enabled

However, the call goes through the Oracle Communications Session Border Controller as an ETS call when it is matched by ETS DN and the applicable NMC does not have an RPH profile assigned. According to the settings in the NMC rule, the Oracle Communications Session Border Controller either diverts or rejects such a call. And when the call matches by RPH rather than ETS DN, the Oracle Communications Session Border Controller applies the configured RPH profile from the relevant NMC rule.

It can be the case that non-ETS calls have RPH in their requests. Here, the Oracle Communications Session Border Controller call treatment is performed according to the settings in the matching RPH profile when there is no matching NMC rule. When you configure treatment as "reject," then the Oracle Communications Session Border Controller rejects the call with a 417 Unknown-Resource Priority status code. When you set the treatment to either "accept" or priority, the Oracle Communications Session Border Controller allows the call to proceed as a non-ETS call or as a priority call.

The ETS r-value can appear in ACK, BYE, INFO, PRACK, REFER and UPDATE requests. In cases when it does and the session with which the request is associated is a non-ETS call, the Oracle Communications Session Border Controller removes the RPH from the request before forwarding it and logs a TRACE-level error. The Oracle Communications Session Border Controller also removes RPH from responses before forwarding them and logs a TRACE-level error when responses contain RPH headers with ETS values for non-ETS sessions.

Call Treatment

This section describes how ETS calls are treated as they traverse the Oracle Communications Session Border Controller.

Call Treatment	Description
Routing	ETS calls are routed the same way as any other calls are, except when the applicable NMC rule's treatment type is divert, and rule defines the next hop. This route takes precedence over other normal routes.
Local NMC	ETS calls are exempt from the local NMC, including: session agent constraints, bandwidth constraints (e.g., per-realm bandwidth), per-user CAC, and CPU constraints. However, the call is subject to the ETS congestions control threshold. Licensing session constraints apply.
ETS Call Congestion Control	ETS calls are subject to congestion control constraints that you configure specifically for this type of traffic. In the global SIP configuration, you set up one option that defines a load limit (greater than that set for normal calls).
ETS CAC	Although the Oracle Communications Session Border Controller uses the call rate control value in the applicable NMC rule, you can also enforce call rate on a per-user basis for ETS calls.

When the Oracle Communications Session Border Controller receives a SIP INVITE with an RPH matching an NMC with an ETS DN, but whose r-values do not match the NMC's rph-profile, the Oracle Communications Session Border Controller behaves as follows:

- If the INVITE does not have the resource-priority option tag and:
  - If the matching NMS is set to PRIORITY, the call will be treated as an NSEP call. If there is an rph-profile matching the r-value (not necessarily the one in the NMC), the Oracle Communications Session Border Controller uses the media-policy from that rph-profile for the call. The rph-policy from the NMC (if present) also applies to the call.
  - If the matching NMC is not set to PRIORITY, the Oracle Communications Session Border Controller will treat the call as a normal one.

If the INVITE contains the resource-priority option tag, the Oracle Communications Session Border Controller will reject the call with the 417 Unknown Resource-Priority message.

Generating Egress RPH

For each ETS call, the Oracle Communications Session Border Controller generates RPH for the outgoing request. It forms this RPH according to the information in the NMC rule. The outgoing request types are INVITE, ACL, BYE, CANCEL, INFO, PRACK, REFER, and UPDATE.

Request RPH Status	Generated Egress RPH
Incoming request without RPH (matched by ETS DN)	Outgoing RPH value becomes the r-value set in the insert-r-value parameter in the RPH policy applied to the NMC rule.
Incoming request without RPH (matched by ETS DN)	If the insert-r-value parameter is empty in the RPH policy applied to the NMC rule or there is no RPH policy applied to the NMC rule, then the egress RPH will also not have RPH.
Incoming request has RPH	Egress RPH is the same as the ingress if the NMC rule has an RPH policy applied but the override-r-value for the policy is empty or if there is not RPH policy applied to the NMC rule. If the override-r-value for the policy is set, then the egress RPH is set to that value.

For example, given an incoming request with the resource priority ets.0, dsn.flash and an RPH policy with an override value of wps.1,ets.1, the egress request would be sent with a resource-priority of wps.1,ets.1,dsn.flash.

The Oracle Communications Session Border Controller also includes RPH in the following series of responses, even when the downstream SIP entity does not respond with an RPH: 1xx, 2xx, 3xx, 4xx, 5xx, and 6xx. The 401 Unauthorized response is an exception.

Media Treatment

If the RPH profile set in an NMC names a media policy, then the Oracle Communications Session Border Controller implements it for the ETS call. This media policy overrides any media policy set in the realm configuration.

The possible Differentiated Services Code Point (DSCP) values for an ETS call are:

- Audio—Applied to the respective media for an ETS call
- Video—Applied to the respective media for an ETS call
- SIP—Applied to the ETS calls' SIP signaling messages, only for the egress call leg for the ETS session

DSCP Marking for NSEP Traffic

You can configure the SBC to mark NSEP calls with DSCP codes on a realm-specific basis. To do this, you assign a **media-policy** that you configure for the realm's NSEP traffic using the **nsep-media-policy** parameter within the egress **realm-config**. When the system identifies this traffic, it handles the traffic according to that **media-policy**, which can include marking the traffic with DSCP values. If you have configure an **nsep-media-policy** on a realm, the system marks egress SIP messages that belong to NSEP sessions with the TOS bits set by that **nsep-media-policy**. The system applies this **nsep-media-policy** to all responses except self-generated responses. The SBC applies precedence to this policy configuration, referring to other traffic policy configuration if this parameter is empty. Applicable media traffic for this configuration includes ETS and WPS. You can configure this policy using the ACLI, REST, and OCSDM.

The **nsep-media-policy** configuration provides you with the flexibility to mark NSEP calls going out different realms with different DSCP values. The system uses the same **network-management-control** and **rph-profile** configurations to identify NSEP traffic, requiring you to configure these objects normally. In addition. the **network-management-control** may or may not use an **rph-policy** in addition to the **rph-profile**. The use of an **rph-policy** is dependent on your needs for special r-value handling on this realm's NSEP traffic.

Having identified the traffic, however, the system next checks to see if there is **nsep-media-policy** configuration on the egress realm. If so, the system uses the **media-policy** configured under the **nsep-media-policy** parameter to determine how to handle the traffic.

When the **nsep-media-policy** is empty, the SBC refers to the **media-policy** of the **RPH-profile** that matches the RPH-header in the incoming SIP message to handle the traffic.

1. Refers to the media-policy associated with RPH-profile that matched the RPH header present in incoming SIP message to handle the NSEP traffic.
2. If there is no **RPH-profile** match, the system applies the realm's **media-policy** to all traffic matching that policy.
3. If no **media-policy** is associated with the **rph-profile**, then the system refers to the common traffic **media-policy** configured at egress to handle NSEP traffic.

The SBC performs this function using the following steps when it receives an NSEP INVITE with an rph-header and the egress realm has a configured **nsep-media-policy** parameter:

1. Check the NMC rule to determine if it has an RPH profile (with r-values) assigned to it.
2. If so, the system compares the r-values in the **RPH-profile** with those in the request's RPH header.
3. If there is a match, and the **nsep-media-policy** parameter has an assigned **media-policy** on the egress **realm-config**, it performs realm-specific DSCP marking.
4. The system handles all traffic according to the **media-policy**, including marking all signaling traffic within the server and client dialogs, as well as the RTP packets for the session with the configured DSCP marking.

> **Note:**
>
> The SBC does not perform DSCP marking on locally generated responses, including 100 trying and 4xx related to the initial invite, using an **nsep-media-policy** because it has not yet determined it is handling an NSEP call.

Configuration

This feature requires the following configurations:

- Enable the **rph-feature** in the **sip-config**.
- Enable the **net-management-control** in the ingress **realm-config**.
- Configure an **rph-profile** to identify and handle NSEP calls based on RPH header match..
- Configure **net-management-control** rules for treating the NSEP calls properly.
- Enable the **rph-feature** in those NMCs.
- Associate the applicable configured **rph-profile** with each NMC.
- Configure a **media-policy** to mark NSEP calls that match that particular **rph-profile** with DSCP.
- Associate the configured **media-policy** to the egress realm by assigning it to the **nsep-media-policy** parameter in the applicable **realm-config**.

> **Note:**
>
> Although the applicable **rph-profile** may have an assigned **media-policy**, the SBC ignores that policy when it finds an **nsep-media-policy** configured on the applicable realm.

See Configuring Packet Marking by Media Type and ensuing tasks for instructions on creating a **media-policy** for media and signaling traffic.

Configure Realm-Specific DSCP Marking for NSEP Traffic

To Configure Realm-Specific DSCP Marking for NSEP Traffic, you enable the **rph-feature** in the **sip-config** and enable the **net-management-control** in the ingress **realm-config** just as you would to enable handling for all NSEP prioritization.

Procedures for this feature include you configuring an RPH Profile for your applicable NSEP Traffic. See Setting up and Applying an RPH Profile for instruction on how to configure your **rph-profile**, including identifying and prioritizing this traffic. The RPH profile contains information about how the system should act on the namespace(s) present in a Resource-Priority header. You can associate a **media-policy** to your **rph-profile**, but the system ignores that setting when performing this realm-specific function, using the **media-policy** you configure under the **nsep-media-policy** parameter instead.

In addition, you:

- Configure a Media Policy for your applicable NSEP Traffic.
- Enable NSEP and Apply the RPH Profile to the NMC.

> **Note:**
>
> You may apply an RPH Policy to the NMC if needed for your implementation.

- Associate the media policy to the egress realm under the **nsep-media-policy** configuration parameter.

Configure a Media Policy for NSEP Traffic

To establish realm-specific DSCP marking for NSEP traffic, you create a **media-policy** that you configure under a realm's **nsep-media-policy** parameter. This policy uses the same controls within **network-management-control** and **rph-profile** configurations, which identify, prioritize and route this NSEP traffic, using a **media-policy** that is specific to this NSEP traffic.
To set up a media policy configuration to mark audio-voice or video packets:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **media-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# media-policy
   ORACLE(media-policy)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

   From this point, you can configure media policy parameters. To view all configuration parameters for media profiles, enter a **?** at the system prompt.

4. **name**—Create a reference name for this policy and press Enter.

   ```
   ORACLE(media-policy)# name myPolicy
   ```

5. Type **tos-settings** and press Enter to configure your TOS settings sub-element. The system prompt changes appropriately.

```
ORACLE(media-policy)# tos-settings
ORACLE(tos-settings)#
```

6. **media-type**—Enter the media type that you want to use for this group of TOS settings. You can enter any of the IANA-defined media types for this value: audio, example, image, message, model, multipart, text, and video. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-type message
```

7. **media-sub-type**—Enter the media sub-type you want to use for the media type. This value can be any of the sub-types that IANA defines for a specific media type. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-sub-type sip
```

8. **media-attributes**—Enter the media attribute that will match in the SDP. This parameter is a list, so you can enter more than one value. The values are case-sensitive and can be up to 255 characters in length. This parameter has no default.

   If you enter more than one media attribute value in the list, then you must enclose your entry in quotation marks ().

```
ORACLE(tos-settings)# media-attributes sendonly sendrecv
```

9. **tos-value**—Enter the TOS value you want applied for matching traffic. This value is a decimal or hexidecimal value. The valid range is:

   - 0x00 to 0xFF.

```
ORACLE(tos-settings)# tos-value 0xF0
```

10. Save and activate your configuration.

Enable NSEP and Apply the RPH Profile to the NMC

In addition to setting the RPH profile for an NMC rule, you also need to enable this feature for the NMC rule.

See "Network Management Controls" in the service provider Configuration Guide for explanation and instruction on how to configure your **net-management-control** for its additional functions, including routing. For this feature, you must have already defined the name of the **rph-profile** that you are using for this traffic.
To enable NSEP for an NMC rule:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **net-management-control** and press Enter.

```
ORACLE(session-router)# net-management-control
ORACLE(net-management-control)#
```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **rph-feature**—Enable this parameter if you want to turn the NSEP feature on for this NMC rule. The default is **disabled**. This feature requires this parameter be enabled.

```
ORACLE(net-management-control)#rph-feature enable
```

5. **rph-profile**—Enter the name of the **rph-profile** you have configured for handling incoming NSEP traffic.

```
ORACLE(net-management-control)#rph-profile myRphProfile
```

6. Navigate to the **sip-config**.

```
ORACLE(configure)# session-router
ORACLE(session-router)#sip-config
ORACLE(sip-config)#
```

7. **rph-feature**—Enable this parameter on the applicable to turn the NSEP feature on for this **realm-config**. The default is **disabled**. This feature requires this parameter be enabled.

```
ORACLE(sip-config)#rph-feature enable
```

8. Save and activate your configuration.

Specify a Media Policy for a Realm's NSEP Traffic

To apply a media policy for NSEP traffic egressing this realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# media-manager
```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm-config
ORACLE(realm)#
```

4. **nsep-media-policy**—Enter the unique name of the **media-policy** that you want to apply to your NSEP traffic.

```
ORACLE(realm)#nsep-media-policy MyNsepMediaPolicy
```

5. Type **done** to complete the configuration.

6. Save and activate your configuration.

Reserving Session Capacity for NSEP

You can reserve session capacity for NSEP traffic, which includes WPS traffic, to reserve sessions from the system's total session capacity pool. The SBC uses this reserved session pool for NESP calls after the general session capacity is utilized. You reserve these sessions by configuring a percentage RPH calls using the **reserved-nsep-session-capacity** parameter.

The SBC identifies NSEP calls using the system's network management controls (NMC) functionality. With NMC and Resource-Priority header (RPH) support enabled on your system, the SBC detects NSEP calls and provides the appropriate treatment for them. When a SIP INVITE for a call arrives at the SBC on an ingress realm where network management controls have been enabled, the SBC checks the NMC rules for a match to the call. If there is none, the call proceeds normally; if there is a match it handles the call according to the rule.

When the SBC identifies an NSEP call, it normally allocates resources for the call from the general session capacity pool. This pool is shared among all kinds of traffic. When this pool gets exhausted, the SBC would begin rejecting calls, including high priority NSEP calls.

To avoid dropping NSEP calls because the system is exhausting session capacity, you can configure the **reserved-nsep-session-capacity** parameter in the **system-config** element. Your configured value specifies the percentage of the total licensed session capacity that the SBC reserves for NSEP sessions. The SBC calculates the number of NSEP reserved sessions from this percentage. By default, this setting is 0%, meaning there are no resources reserved solely for NSEP sessions. When configured to a value over 0%, the SBC reserves session capacity resources for NSEP traffic.

After this configuration, the SBC derives its general session pool by simply subtracting the number of NSEP reserved sessions from the total session capacity. The SBC starts to use this NSEP reserved session pool when it exhausts the general session pool. The SBC continuously monitors this general session pool. When sessions become available from this pool again, the SBC moves ongoing NSEP calls from the NSEP-reserved pool to the general pool. This reduces the number of calls being used by the NSEP pool, further ensuring session availability if the general pool becomes exhausted again.

The SBC rejects your **reserved-nsep-session-capacity** configuration during the save process if you specify a value that exceeds the percentage capacity of sessions currently available. You can also check this error using **verify-config**.

When you configure **reserved-nsep-session-capacity** to a non-zero value, the system also monitors current NSEP traffic and issues alarms and SNMP traps, using critical, major and minor thresholds, to notify you when you are running low on your reserved NSEP session capacity. The default thresholds are 70%, 80% and 90%, equating to minor, major and critical levels of busy NSEP sessions. You can customize these thresholds using the **reserved-nsep-sessions** sub-element type in the **system-config**, **alarm threshold**.

Below is an example of a Reserved NSEP session alarm.

```
ID Task Severity First Occurred Last Occurred
327684 3068 5 2021-06-18 03:01:25 2021-06-18 03:01:25
Count Description
1 Reserved NSEP session usage (80%) is exceeded threshold of 70%.
```

The system issues these alarms simultaneously with the apSysResrvdNsepSessionCapacity trap within the apSysMgmtGroupTrap group to notify you when session pool usage exceeds

these thresholds over SNMP. The system issues the apSysMgmtGroupClearTrap when the reserved session usage falls below this threshold, as documented in the *MIB Guide*.

Related Configuration

The following configurations, which enable and define network management behavior, are also required for this feature:

- Enable the **rph-feature** in the **sip-config**.
- Enable the **net-management-control** in the ingress **realm-config**.
- Enable the **rph-feature** in your target **net-management-control**.
- Configure an **rph-profile** to handle NSEP calls containing RPH headers in the **session-router**.
- Apply your **rph-profile** to your **net-management-control**.

Reporting NSEP Reserved Capacity Statistics

The SBC provides reporting on NSEP reserved capacity using the **show sessions** command on the ACLI. Reported data includes the number of reserved NSEP sessions and the number of inbound NSEP sessions.

```
SBC# show sessions
03:21:47-165 Capacity=2000
General Session Capacity=1500
Session Stats                              -- Period -- -------- Lifetime
--------
                                  Active     High   Total      Total
PerMax   High
Total Sessions                       0         0      0          0
0       0
SIP Sessions                         0         0      0          0
0       0
H.323 Sessions                       0         0      0          0
0       0
IWF Stats                                  -- Period -- -------- Lifetime
--------
                                  Active     High   Total      Total
PerMax   High
H.323 to SIP Calls                   0         0      0          0
0      0
SIP to H.323 Calls                   0         0      0          0
0      0
SIP Audio/Video Stats                      -- Period -- -------- Lifetime
--------

                                  Active     High   Total      Total
PerMax   High
Audio Calls                          0         0      0          0
0      0
Video Calls                          0         0      0          0
0      0
Messaging Sessions                   0         0      0          0
0      0
Reserved NSEP Session Capacity=500
Session Stats                              ---- Lifetime---
```

```
                                    Current    Total PerMax
NSEP Inbound Sessions                 0          0     0
```

The system also reports on lifetime values of these statistics using SNMP. You can get this information using SNMP queries to the apSysMgmtMIBNSEPStatsObjects MIB, as documented in the *MIB Guide*.

Reject Non-Emergency Traffic using Emergency DSCP

You can configure the SBC to reject traffic that uses emergency DSCP codes to designate itself as emergency traffic. This function applies to calls from both registered and unregistered endpoints and for both UDP and TCP traffic.

When configured, the SBC checks initial INVITE packets to see if it is using any of the DSCP codes you configured in an **emergency-dscp-profile** to signal an emergency call. When there is a match, the SBC validates the use of that code by checking whether the INVITE also includes an ETS dial number (DN) and/or the SIP Resource-Priority header (RPH) as a SIP emergency designation. If not, the SBC drops the INVITE, logs the attempt to improperly use that code, and, if configured, replies to the sender with an error message. You can specify the contents of this message. If you do not configure an error code and text, the SBC sends the default SIP error code and message “403 unauthorized attempt to use reserved resources”.

To enable this functionality, you configure an **emergency-dscp-profile** under the **session-router** branch. These multi-instance profiles include codes you specify for matching and, optionally, error codes/text to send back towards the sender. Parameters include:

- **name**—Establishes a label for individual emergency-dscp-profiles. You use this label to apply these profiles to session-agents, sip-interfaces or the sip-config. This field supports a maximum of 128 characters.
- **tos-values**—A space-separated string of decimal or hex numbers, that the SBC searches for within a packet. If it finds a value configured here, and the packet does not have any SIP information identifying the call as an emergency call, the system rejects the call.
- **error-code**—This optional field can include an integer that specifies the error code you send back to any endpoint that sends a non-emergency INVITE that includes a DSCP value you configured in this profile.
- **error-message**—This optional field can include a string that specifies the error text that you send back to any endpoint that sends a non-emergency INVITE that includes a DSCP value you configured in this profile.

You apply these profiles to a **session-agent**, a **sip-interface** or the **sip-config**, by configuring the parameter in those elements, which has the same name as the profile name (**emergency-dscp-profile**), with the name of the applicable profile. The system using that order as precedence for which profile, if any, applies.

As a prerequisite, you must have also configured the SBC for NSEP traffic identification and have enabled emergency treatment using NMC rule configuration.

The SBC uses this feature on initial INVITEs only. If a call fails this check based on the initial INVITE, there would be no further traffic for the call. For any call that proceeds beyond the initial INVITE, the SBC assumes the call is progressing properly and does not reject messages based on DCSP.

This feature imposes no changes to the system's handling of properly designated emergency calls.

Reporting

When the SBC rejects a non-emergency call based on this feature, it:

- Increments the “Drop Unauth NSEP DSCP” counter in the **show sipd error** command.
- Increments your configured error codes, unless they are well known codes, under the “4XX Client Error” statistics in the **show sipd invite** output.
- Includes debug/trace/error/info level logging in all required log files (log.sipd).

Reporting on NSEP Traffic Statistics

The SBC provides you with NSEP traffic statistics from the ACLI and SNMP. You can access system wide NSEP traffic reports when you configure the system for applicable network management controls (NMC). In addition, you can configure the system to provide realm-specific reporting on a per-realm basis by configuring the **nsep-stats-profile** on the **session-router** and enabling **nsep-stats** on the applicable realms.

The system-wide NSEP statistics provide global incoming/outgoing success/reject sessions. You can filter your results to a specific or all r-values. Per RFC-4412, the SBC treats r-value names as case insensitive strings. The SBC presents these statistics using current window statistics and lifetime statistics:

- In the current window, which uses a duration timeframe of 30 minutes, the SBC displays:
  - Currently active sessions
  - The highest number of sessions in the current window
  - Total sessions in the current timeframe
- The SBC categorizes lifetime statistics using:
  - Total sessions
  - The highest number recorded among all windows reported
  - The highest number of sessions in the lifetime window

The SBC reports NSEP statistics simultaneously through SNMP. To support this the SBC uses a set of nested MIB index and value objects that correspond to the ACLI statistics and display.

When you use the ACLI **show nsep-stats** command without further arguments, the system displays counters on inbound and outbound sessions. You extend this command with the **all** argument or a specific r-value (namespace and r-priority combination).

The system presents realm-based NSEP statistics using the same timeframe/data scheme presented for system-wide statistics and allows you to filter based on dialed number in addition to r-values:

- Dialed Number Statistics—You configure a set of dialed numbers with a country-code in your **nsep-stats-profile**. The country code is common for all dialed number, similar to the STD code. Whenever the SBC processes NSEP traffic, it matches a combination of country-code and dialed number with the req-uri of received/sent messages and increments the corresponding counters with the match results.
- r-values—The system records, increments and displays realm-based r-values in the same manner as it does for system-wide statistics.

Use these show commands with the following arguments to display realm-based NSEP statistics:

- show nsep-stats realms <realm-name>
- show nsep-stats realms <realm-name> <rvalue-name>
- show nsep-stats realms <realm-name> dialed-numbers

You can reset any or all NSEP statistics counters. You reset these statistics to zero using the **reset** command with same arguments above.

Presentation examples of this reporting is available for the ACLI in the *Maintenance and Troubleshooting Guide* and for SNMP in the *MIB Guide*.

Configuring Realm-based NSEP Reporting

For the system to produce realm-level statistics on NSEP traffic, you configure the **nsep-stats-profile** on the **session-router** and enable **nsep-stats** on the applicable realms.

The **nsep-stats** parameter allows you to specify the realms on which you collect NSEP statistics individually. The system does not collect these NSEP statistics for any realm that has this parameter disabled.

Applicable **nsep-stats-profile** parameters include:

- **state**—Enabled/Disabled
- **rvalues**—Add the r-values on which you want to collect NSEP statistics within realms
- **dialed-numbers**—Optionally add the dialed number(s) on which you want to collect NSEP statistics for realms.
- **feature-code**—Add the country STD Code, appended with each configured dialed number. This must be configured if you have configured a **dialed-numbers**.

If you do not configure the **feature-code** parameter in conjunction with the **dialed-number**, the system throws a verify error when you use the **done** command on your **nsep-stats-profile** element.

```
ERROR: feature code must be configured when dialed number is configured
```

RPH Configuration

This section shows you how to configure RPH profiles and policies that enable the Oracle Communications Session Border Controller to act on SIP calls that have an ETS DN and/or an RPH carrying ETS resources values. There are also settings for the global SIP configuration and for the NMC rule configuration that support this feature.

In addition, note that:

- You must set a media policy for the RPH profile to use. Check your system configuration and note the name of the media policy that best suits your needs.
- Valid values for the parameters that take r-values are wps.x and ets.x, where x is 0 through 4.

Remember to save and activate your configuration after you have completed the processes detailed in this section.

Setting Up and Applying RPH Policy

The RPH policy is a configuration on the Oracle Communications Session Border Controller that you apply to NMC rules. It designates the following for ETS/WPS namespaces:

- An override resource value—Resource value used to override the incoming RPH's resource value
- An insert resource value—Resource value inserted when the Oracle Communications Session Border Controller does not recognize the RPH, the incoming request has no RPH, or the call is H.323 and matches an NMC rule based on the ETS DN

Note that RPH policies do not apply for DSN, DRSN, Q.735, or any other type of namespace; these remain untouched in outgoing requests.

To configure an RPH policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **rph-policy** and press Enter. From here, you can configure the individual parameters for the RPH policy.

   ```
   ORACLE(session-router)# rph-policy
   ORACLE(rph-policy)#
   ```

4. **name**—Enter the name that uniquely identifies this RPH policy. This is the value you use to apply the policy in the NMC rules configuration. There is no default for this parameter, and you are required to set it.
5. **override-r-value**—Enter the value that the system uses to override r-values in the original RPH.

   ```
   ORACLE(rph-policy)# override-r-value ets.1
   ```

6. **insert-r-value**—Enter the value that the Oracle Communications Session Border Controller inserts into the RPH.

   ```
   ORACLE(rph-policy)# insert-r-value wps.1
   ```

7. **rph-policy**—Enter the name of the RPH policy that you want to apply for this NMC rule. This parameter is empty by default; if you do not set an RPH policy, none will be applied.

Setting Up and Applying RPH Profile

The RPH profile contains information about how the system should act on the namespace(s) present in a Resource-Priority header (if any). The list of resource values in this configuration calls out the resource values (or r-values) recognizable to the Oracle Communications Session Border Controller; the ETS and WPS namespaces are supported.

You also set a media policy for the RPH profile to use; it defines the Differentiated Services Code Point (DSCP) that the Oracle Communications Session Border Controller uses for media or signaling packets belonging to the egress call leg for the ETS session.

The call treatment parameter tells the Oracle Communications Session Border Controller what to do with a non-ETS call that has RPH in its request; the call can be allowed, rejected, or treated as a priority call.

To configure an RPH profile:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **rph-profile** and press Enter. From here, you can configure the individual parameters for the RPH policy.

```
ORACLE(session-router)# rph-profile
ACMEORACLEPACKET(rph-profile)#
```

4. **name**—Enter the name that uniquely identifies this RPH profile. This is the value you use to apply the profile in the NMC rules configuration. There is no default for this parameter, and you are required to set it.
5. **r-values**—Enter one or more r-values that the Oracle Communications Session Border Controller is to recognize for matching purposes. When you enter more than one value in the list, you type the name of the parameter followed by a Space, open quotation mark, the values for the list separated by spaces, a closed quotation mark. Then press Enter.

   You must enter them in the order reflected below (a WPS and then an ETS value). A WPS call always has to have an ETS namespace.

```
ORACLE(rph-profile)# r-values "wps.0 ets.2"
```

6. **media-policy**—Enter the name of a media policy configuration that you want applied for this RPH profile. The Oracle Communications Session Border Controller implements this media policy for the ETS call, and this media policy overrides any media policy set in the realm configuration.
7. **call-treatment**—Enter the call treatment method for a non-ETS call that contains RPH matching it to this profile. The default is accept. The valid values are:
   - **accept**—The call proceeds as it normally would
   - **reject**—The Oracle Communications Session Border Controller rejects the call with the 417 Unknown-Resource Priority status code
   - **priority**—The Oracle Communications Session Border Controller treats the call as a priority call

Enabling NSEP for an NMC Rule

In addition to the RPH policy and RPH profile you can set for an NMC rule, you also need to set the state of this feature for the NMC rule.

To enable NSEP for an NMC rule:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **net-management-control** and press Enter.

```
ORACLE(session-router)# net-management-control
ORACLE(net-management-control)#
```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **rph-feature**—Enable this parameter if you want to turn the NSEP feature on for this NMC rule. The default is **disabled**. The valid values are:
   - enabled | disabled

Global SIP Configuration Settings Enabling NSEP

For the global SIP configuration, you can turn the NSEP feature on, and you can also set parameters that support call admission and congestion control.

In addition, you can enable the insertion of the ARPH header in a response when the resource-priority tag is present in the Require header and the Oracle Communications Session Border Controller rejects the request with a 417 Unknown Resource-Priority response. The ARPH value is the list of r-values you set in the RPH profile.

To enable NSEP for the global SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **rph-feature**—Enable this parameter if you want to turn the NSEP feature on for the global SIP configuration. The default is **disabled**. The valid values are:
   - enabled | disabled

Global SIP Configuration Settings Enabling CAC and Congestion Control

To set call admission and congestion control parameters for NSEP:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **nsep-user-sessions-rate**—Enter the maximum INVITEs per second to admit for ETS calls on a per-user basis. To enable NSEP call admission control (CAC), you must change the parameter value from 0; if you leave this parameter set to 0, then it is the same as disabling CAC for ETS calls. The default is **50**. The valid range is:

   - Minimum—0
   - Maximum—999999999

5. **options**—To enable congestion control for ETS calls, you configure an option that sets the CPU threshold. If this threshold is exceeded, the Oracle Communications Session Border Controllerrejects new ETS calls with the 503 Service Unavailable response. The value you set here should be larger than the load limit value for normal calls; ETS calls are allowed even when the load limit threshold for normal calls is exceeded.

   The threshold value can be between 0 and 100. Using a value of 0 or 100 for this parameter disables ETS call congestion control.

   Set the options parameter by typing **options**, a Space, the option name **nsep-load-limit** with a plus sign in front of it, then the equal sign and the ETS call threshold you want to set. Then press Enter.

   ```
   ACMEPACKET(sip-config)# options +nsep-load-limit=50
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

Global SIP Configuration Settings Enabling ARPH Insertion

To enable ARPH insertion in responses:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—To enable ARPH insertion in responses type **options**, a Space, the option name **insert-arp-header** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-config)# options +insert-arp-header
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

Setting Up NSEP for Session Agents

In earlier releases, the Oracle Communications Session Border Controller supports NSEP-related CAC for users and for NMC. You can now configure a sessions-per-second rate for session agents. Set in the global SIP configuration, this rate applies to all SIP session agents. When session exceed the limit, the Oracle Communications Session Border Controller rejects them with a 503 Service Unavailable message.

To configure NSEP limits for SIP session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **nsep-sa-sessions-rate**—Enter maximum acceptable number of SIP INVITES (NSEP sessions) per second to allow for SIP session agents. This parameter defaults to 0, meaning there is no limit.
5. Save and activate your configuration.

Configure NSEP Resource Reservation

To specify the session capacity to be reserved for NSEP calls, you configure the **reserved-nsep-session-capacity** parameter in the **system-config** element. Your configured value specifies the percentage of the total licensed session capacity that the SBC reserves for NSEP traffic sessions. The SBC calculates the number of NSEP reserved sessions from this percentage. The default is 0%.

When you configure **reserved-nsep-session-capacity** to a non-zero value, the system uses default thresholds to trigger minor, major and critical alarms and SNMP traps when session utilization exceeds these thresholds. You can customize these thresholds by configuring the **reserved-nsep-sessions alarm threshold** type in the **system-config**.

```
ORACLE(alarm-threshold)# type reserved-nsep-sessions
```

To reserve 15% of total session capacity for NSEP sessions, and customize its minor alarm to trigger at 50% of NSEP sessions:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **system-config** and press Enter.

   ```
   ORACLE(configure)# system-config
   ORACLE(system-config)#
   ```

4. Type **reserved-nsep-session-capacity**, followed by a percentage value of session resources you want to reserve for NSEP sessions. The example below uses 15%.

   ```
   ORACLE(system-config)# reserved-nsep-session-capacity 15
   ```

5. To customize NSEP session utilization alarm and trap thresholds, type **alarm threshold** and press Enter.

   ```
   ORACLE(system-config)# alarm threshold
   ORACLE(alarm threshold)#
   ```

6. Type **type**, followed by the value **reserved-nsep-sessions**, then press Enter.

   ```
   ORACLE(alarm threshold)# type reserved-nsep-sessions
   ORACLE(alarm-threshold)# severity minor
   ORACLE(alarm-threshold)# value 50
   ```

7. Type **done** to complete the **alarm threshold** configuration.
8. Type **done** to complete the **system-config** configuration.
9. Save and activate your configuration.

E-CSCF Emergency Setting Precedence for NMC

When the Oracle Communications Session Border Controller acts as an E-CSCF, it can route emergency or priority calls (i.e., 112, 911, 999 calls) to the corresponding ECS/PSAP based on the calling party's information. However, for registered users, this ability mixes with the Oracle Communications Session Border Controller's NMC function so that the Service Routes takes precedence over the NMC. So rather than routing the emergency call to the ECS/PSAP, the call ends up at the S-CSCF of the Service Route.

For non-registered users where there no Service Route exists, this is not an issue.

When you configure the **treatment** parameter within the **network-management-control** configuration to **apply-local-policy**, you allow the NMC to take precedence over cached Service Route when a session matches an NMC rule. Instead, it locates a matching local policy.

Without the NMC configured to take precedence, the Oracle Communications Session Border Controller does not function optimally as an E-CSCF for registered users.

Consider the following scenario in which a UE registers as a P-CSCF to the S-CSCF via the Oracle Communications Session Border Controller. In this case, the registrar returns a Service Route header in a 200 OK response to the REGISTER message to the Oracle Communications Session Border Controller. Or, if an implicit service route is enabled, the Oracle Communications Session Border Controller generates a Service Route that it saves in the register cache before forwarding the 200 OK on to the UA. Then when a new INVITE comes from the UA, the Oracle Communications Session Border Controller checks its register cache and uses the Service Route as the next hop—without taking local policy into consideration.

Configuring he **treatment** parameter to **apply-local-policy** requires local policy's consideration in deciding the next hop. If you configure this value and the Oracle Communications Session Border Controller receives a new INVITE, it will decide whether the session is priority, registered, or with Service Route. Then it will determine if the call is E-CSCF. If so and the first matching local policy has E-CSCF enabled, the priority local policy is applied over the Service Route. The Oracle Communications Session Border Controller stays with the Service Route if not.

E-CSCF Emergency Configuration

This section shows you how to configure a network management control rule to take precedence over a Service Route.

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **net-management-control** and press Enter.

```
ORACLE(session-router)# net-management-control
ORACLE(net-management-control)#
```

4. **treatment—Follow your entry with this value:**

   - apply-local-policy

```
ORACLE(net-management-control)# treatment apply-local-policy
```

   This value allows a network management control rule to take precedence over a service route for an emergency or priority call.

5. Type **done** and continue.

SIP TCP Connection Reuse

You can configure your Oracle Communications Session Border Controller to reuse TCP connections created by SIP peering devices for outgoing SIP in-dialog and out-of-dialog request transactions.

The SIP draft draft-ietf-sip-connect-reuse-07.txt describes a way for SIP UAs to reuse connections created by a remote endpoint for outgoing requests for TLS. The Oracle Communications Session Border Controller does not support the model connection reuse signalled by a parameter; rather, it is provisioned on a per-session-agent basis.

You enable SIP TCP connection reuse on a per-session-agent basis. The Oracle Communications Session Border Controller checks incoming TCP connection request to determine if they are from session agent that has this feature turned on. When it is, the Oracle Communications Session Border Controller adds the connection's source address to its list of alias connections. This is a list of connections that the Oracle Communications Session Border Controller can use for outgoing requests rather than creating its own connection (as it does when this feature is not enabled). So if a preferred connection fails, the Oracle Communications Session Border Controller can refer to this list and use the alias connection.

The presence of an alias parameter in the Via header is just one mechanism that will call the Oracle Communications Session Border Controller to use the inbound TCP/TLS connection for outbound requests. The Oracle Communications Session Border Controller will automatically add an alias for the inbound connections in the following circumstances:

- The other end of the connection is behind a NAT. When the Oracle Communications Session Border Controller sees that the Via sent-by does not match the source address of the connection, it will automatically reuse the connection to deliver requests to the UA.
- The Contact address of a REGISTER request received on a TCP connection matches the source address and port. This is because the contact adress is the ephemeral port the UA used to form the connection to the Oracle Communications Session Border Controller and, therefore, will not be listening on that port for inbound connections.
- The presence of reuse-connections in the options field of the sip-interface will cause the Oracle Communications Session Border Controller to reuse all inbound TCP connections for sending rquests to the connected UA.

SIP TCP Connection Reuse Configuration

This section describes how to enable SIP TCP connection reuse for a session agent. Currently there are two options for the new **reuse-connections** parameter: none (which turns the feature off) and tcp (which enables the feature for TCP connections). You also set the re-connection interval.

To enable SIP TCP connection reuse for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the session agent that you want to edit.

4. **reuse-connections**—Enable or disable SIP TCP connection reuse. The default is **none**. This value disables the feature. The valid values are:
   - tcp | sctp | tls |none
5. **tcp-reconn-interval**—Enter the amount of time in seconds before retrying a TCP connection. The default for this parameter is **0**. The valid range is:
   - Minimum—0, 2
   - Maximum—300
6. Save and activate your configuration.

SIP TCP Keepalive

The Oracle Communications Session Border Controller supports a special TCP keepalive mechanism for SIP. By enabling this feature either for a session agent or for a SIP interface, you allow the Oracle Communications Session Border Controller to use standard keepalive probes to determine whether or not connectivity with a remote peer has been lost.

This feature adds to the Oracle Communications Session Border Controller's pre-existing TCP keepalive functionality that you can enable in the network parameters configuration. Using existing functionality, you can customize keepalive timing by:

- Specifying the number of unacknowledged packets the Oracle Communications Session Border Controller sends to the remote peer before it terminates the TCP connection.
- Specifying the number of seconds of idle time before TCP keepalive messages are sent to the remote peer.

You can now set three modes for TCP keepalive for session agents and SIP interfaces:

- none—(Default) Keepalives are not enabled for use with the session agent/SIP interface; when you select this setting for a session agent, it will use the setting for this feature from the SIP interface.
- enabled—Keepalives are enabled for the session agent/SIP interface.
- disabled—Keepalives are disabled for the session agent/SIP interface.

Note that the setting for this feature for a session agent takes precedence over that for a SIP interface. In addition, the session agent offers you a way to set the re-connection interval.

SIP TCP Keepalive Configuration for Session Agents

To enable SIP TCP keepalive for session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the session agent that you want to edit.

4. **tcp-keepalive**—Enable or disable standard keepalive probes to determine whether or not connectivity with a remote peer is lost. The default value is **none**. The valid values are:

   - none | enabled | disabled

   ```
   ACMEPACKET(session-agent)# tcp-keepalive enabled
   ```

5. Save and activate your configuration.

SIP TCP Keepalive Configuration for SIP Interfaces

To enable SIP TCP keepalive for SIP interfaces:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

If you are adding support for this feature to a pre-existing SIP interface, then you must select (using the ACLI **select** command) the SIP interface that you want to edit.

4. **tcp-keepalive**—Enable or disable SIP TCP keepalive. The default value is **none**. The valid values are:
   - none | enabled | disabled

```
ORACLE(session-agent)# tcp-keepalive enabled
```

5. Save and activate your configuration.

SIP Enforcement Profile and Allowed Methods

For this feature, you use a configuration called an enforcement profile that allows you to configure sets of SIP methods that you want applied to: the global SIP configuration, a SIP interface, a realm, or a SIP session agent. The enforcement profile is a named list of allowed methods that you configure and then reference from the configuration where you want those methods applied.

SIP Enforcement Profile Configuration

To use the enforcement profile, you need configure it with a name and the list of SIP methods you want to designate as allowed. Then you need to configure the global SIP configuration, a SIP interface, a realm, or SIP session agent to use the set.

Setting Up and Enforcement Profile

To set up an enforcement profile:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **enforcement-profile** and press Enter.

```
ORACLE(session-router)# enforcement-profile
ORACLE(enforcement-profile)#
```

4. **name**—Enter the name for the enforcement profile. This parameter has no default, but you must note it so that you can apply this set of allowed SIP headers in: the global SIP configuration, a SIP interface, a realm, or SIP session agent.

```
ORACLE(enforcement-profile)# name EnfProfile1
```

5. **allowed-methods**—Enter a list of SIP methods that you want to allow for this set. The default value is **none**. Valid values are:
   - INVITE | REGISTER | PRACK | OPTIONS | INFO | SUBSCRIBE | NOTIFY | REFER | UPDATE | MESSAGE | PUBLISH

     To enter multiple methods for the list, type the parameter name followed by a space, then the names of all methods you want to include each separated by a only a comma and in capital letters.

```
ORACLE(enforcement-profile)# allowed-methods INVITE,REGISTER,PRACK
```

6. Save and activate your configuration.

Applying an Enforcement Profile

You can apply an enforcement profile to: the global SIP configuration, a SIP interface, a realm, or SIP session agent. This section shows you how to do all four. Remember that if you are adding this functionality to a pre-existing configuration, you need to select the configuration you want to edit.

To apply an enforcement profile to the global SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to the global SIP configuration.
5. Save and activate your configuration.

   To apply an enforcement profile to a SIP interface:
6. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

7. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

8. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

9. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this SIP interface.
10. Save and activate your configuration.

    To apply an enforcement profile to a SIP session agent:

11. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

12. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

13. Type **session-agent** and press Enter.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

14. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this session agent.
15. Save and activate your configuration.

    To apply an enforcement profile to a realm:

16. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

17. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

18. Type **realm-config** and press Enter.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

19. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this realm.
20. Save and activate your configuration.

Local Policy Session Agent Matching for SIP

When you enable the local policy session agent matching option in your global SIP configuration, you change the way local policies match session agents. Normally, the Oracle Communications Session Border Controller looks up and stores matched session agents configured as next hops so it does not need to perform the lookup while processing requests. In this type of matching, the Oracle Communications Session Border Controller does take the realm set in the local policy attributes into consideration. When the Oracle Communications Session Border Controller performs its regular matching method and you have enabled overlapping IP addresses for session agents, the Oracle Communications Session Border Controller might match session agents to different realms than the ones you intended when creating your configuration.

Local policy session agent matching provides a way to match session agents differently, taking realms and nested realms into consideration during the matching process. This difference is key to deployments with multiple peering partners that use the overlapping IP address feature, and have multiple local policies routing to the same IP address in different realms where some target next hops require session constraints but others do not. In the cases where no session constraints are required, session agents are not needed. But session agents still match the local policy, applying their constraints, because they match the next hop IP address.

In addition to modifying this behavior, this feature also affects the use of realms and nested realms. It triggers the use not only of realms, but of all the realms nested however deeply—thereby improving matching efficiency.

You can set the local policy session agent matching option with values that define how the Oracle Communications Session Border Controller performs session agent matching:

- **any**—The Oracle Communications Session Border Controller looks up and stores matched session agents configured as next hops so it does not need to perform the lookup while processing requests, without regard to realms.
  This behavior is the default when the SIP configuration does not have the local policy session agent matching option set.
- **realm**—The Oracle Communications Session Border Controllerselects session agents in the realm that the local policy attribute indicates; this provides an exact match, rather than not taking the realm into consideration during session agent selection.
  For example, the session agent is a match if the session agent **realm-id** and the local policy attribute **realm** parameters are an exact match.
- **sub-realm**—Session agents in the same realm or the same realm lineage—where session agents and realms are related to one another via realm parent-child relationships no matter the depth of realm nesting configured
  For example, the session agent is a match if the local policy attribute **realm** is a sub-realm of the realm specified in the session agent **realm-id** parameter.
- **interface**—Session agents in the same realm or same realm lineage via the realm set in the local policy attribute, and whose realm uses the same signaling interface as the realm set in the local policy attribute
  For example, the session agent is a match if the session agent **realm-id** is a sub-realm of the local policy attribute **realm**, and both referenced realms use the same SIP signaling interface.
- **network**—Session agents whose realm is in the realm lineage for the same realm set in the local policy attributes, and whose realm is associated with the same network interface as the realm set in the local policy attributes

For example, the session agent is a match if the session agent **realm-id** is a sub-realm of the local policy attribute **realm**, and realm reference by both use the same network interface.

If it cannot find a match, the Oracle Communications Session Border Controller will use the IP address as the next hop. Further, requests matching local policy attributes will not be associated with session agents, and so their constraints will not be applied.

The Oracle Communications Session Border Controller stores session agent information that it looks up when performing local policing session agent matching. To perform the lookup, it uses the session agent hostname as a key. When the hostname is an FQDN and there is a configured IP address in the **ip-address** parameter, the Oracle Communications Session Border Controller uses the ip-address value as a secondary key. Given this implementation, the following are true when selecting session agents:

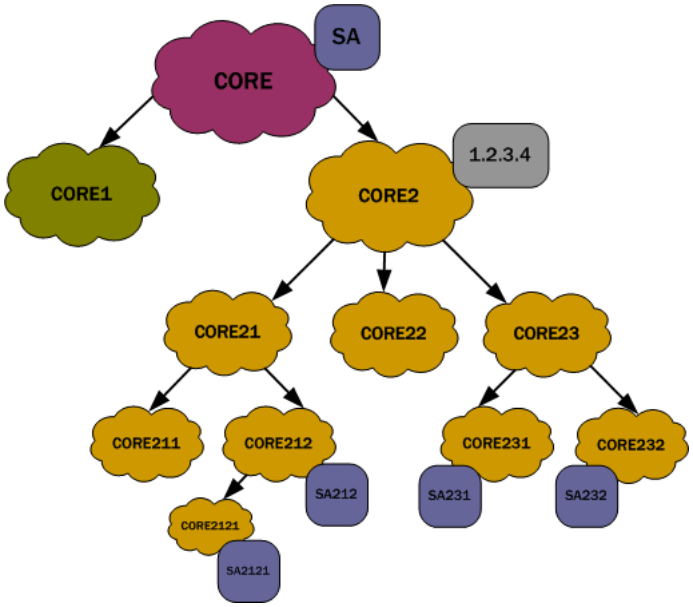
- If multiple session agents share the same IP address, the one with an IP address in the hostname parameter takes precedence.
- If all session agents with the same IP address have an FQDN as their hostname, the one whose name is alphabetically lower will take precedence, where alphabetically lower means earlier in the alphabet (closer to A than to Z).
- For non-global session agents (whose realms are configured but not wildcarded) with an IP address, the Oracle Communications Session Border Controller uses a key that is a combination of the IP address and the realm in the form <address>:<realm>.
- For a session agent whose realm has a parent realm, the Oracle Communications Session Border Controller uses a combination of the IP address, realm, and realm-path (or lineage for the realm) in the form <address>:<realm-path>. For example, the realm path for a realm core3 with a parent core2, which in turn has a parent core would be core:core2:core3.

When it looks up a session agent with a realm, the Oracle Communications Session Border Controller first searches for an exact match for the IP address and realm combination. If this fails, it performs a second search if the desired realm has parents or children. The Oracle Communications Session Border Controller locates an entry in its repository of session agent information that is greater than or equal to the IP address with the base realm, which is the ancestor of the desired realm without a parent. Having gathered this set of candidates, the Oracle Communications Session Border Controller narrows down the search for a match by comparing sub-realms and determines there is a match if either:

- The desired realm path is a sub-string of the entry's realm path, or
- The entry's realm path is a substring of the desired realm path (i.e., the desired realm is a sub-realm of the entry's realm)

Then the Oracle Communications Session Border Controller orders the candidates by depth of the entry's realm-path, or number of levels from the base realm relative to the depth of the desired realm. By searching the ordered set until the entry's realm depth equals the desired realm's depth, the Oracle Communications Session Border Controller determines a parent candidate, all subsequent entries being sub-realms of the desired realm. The Oracle Communications Session Border Controller only considers entries at the first level deeper than the desired realm. If at this point there is only one entry, the Oracle Communications Session Border Controller deems it a match. Otherwise, it selects the parent candidate as the matching entry. In the event the search does not yield a matching realm, the Oracle Communications Session Border Controller uses the global session agent for the IP address, if there is one.

The following diagram shows the realm tree, where the clouds are realms and squares are session agents, representing a group of session agents sharing the IP address 1.2.3.4. The Oracle Communications Session Border Controller searches for the session agents lower in the tree along the session agent realm-path and the desired realm.



For the diagram above, the following shows how the hostname would look for this group of session agents.

Key	Session Agent (hostname[realm])
1.2.3.4 (This session agent owns the primary key for the IP address because its hostname is the IP address.)	1.2.3.4[CORE2]
1.2.3.4:CORE (IP+realm key entry)	SA[CORE]
1.2.3.4:CORE (IP+realm key entry)	1.2.3.4[CORE2]
1.2.3.4:CORE212 (IP+realm key entry)	SA212[CORE212]
1.2.3.4:CORE2121 (IP+realm key entry)	SA2121[CORE2121]
1.2.3.4:CORE231 (IP+realm key entry)	SA231[CORE231]
1.2.3.4:CORE232 (IP+realm key entry)	SA232[CORE232]
1.2.3.4:CORE: (IP+realm-path key entry)	SA[CORE]
1.2.3.4:CORE:CORE2: (IP+realm-path key entry)	1.2.3.4[CORE2]
1.2.3.4:CORE2:CORE21:CORE212 (IP+realm-path key entry)	SA212[CORE212]
1.2.3.4:CORE2:CORE21:CORE212:CORE2121 (IP+realm-path key entry)	SA2121[CORE2121]
1.2.3.4:CORE2:CORE23:CORE231 (IP+realm-path key entry)	SA231[CORE231]
1.2.3.4:CORE2:CORE23:CORE232 (IP+realm-path key entry)	SA232[CORE232]

For each realm in the table above, the search results for each realm would look like this:

IP Address	Realm	Session Agent (hostname[realm])
1.2.3.4	CORE	SA[CORE]
1.2.3.4	CORE2	1.2.3.4[CORE2]
1.2.3.4	CORE21	SA212[CORE212[
1.2.3.4	CORE211	1.2.3.4[CORE2]
1.2.3.4	CORE212	SA212[CORE212]
1.2.3.4	CORE2121	SA2121[CORE2121]
1.2.3.4	CORE22	1.2.3.4[CORE2]
1.2.3.4	CORE23	1.2.3.4[CORE2]
1.2.3.4	CORE231	SA231[CORE231]
1.2.3.4	CORE232	SA232[CORE232]

Local Policy Session Agent Matching Configuration

When you enable local policy session agent matching, remember that you can choose from five different ways to use the feature: **all**, **realm**, **sub-realm**, **interface**, and **network**.

This example shows you how to use the **realm** selection.

To enable local policy session agent matching using the realm method:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **options**—Set the options parameter by typing options, a Space, the option name **lp-sa-match=X (where X is the local policy session agent matching method you want to use)** with a plus sign in front of it. Then press Enter.

   Remember that if you do not specify a method, the system uses the **all** method.

```
ORACLE(sip-config)# options +lp-sa-match=realm
```

   If you type **options** and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

- Unordered—Meaning that the endpoint can deliver data within regard for their stream sequence number

You set this preference in the network parameters configuration.

About Wildcarding

The Oracle Communications Session Border Controller supports wildcarding the event type in the **subscribe-event** configuration. To wildcard the value, you enter an asterisk (*) for the **event-type** parameter instead of typing in the name of an actual event type.

When you wildcard this value, the Oracle Communications Session Border Controller applies the subscription limitations you set across all event types. Or, if you have entered multiple subscribe-event configurations, the Oracle Communications Session Border Controller applies the wildcard limits across the event types for which you have not set limits.

Consider the following example of a configured enforcement profile with a wildcarded **subscribe-event** configuration:

```
enforcement-profile
        name                            rulefour
        allowed-methods
        sdp-address-check               disabled
        subscribe-event
                 event-type                     *
                 max-subscriptions              1
        subscribe-event
                 event-type                     xyz
                 max-subscriptions              0
        last-modified-by                admin@console
        last-modified-date              2008-11-11 12:49:27
```

In this example, the enforcement profile allows all subscriptions that are event type xyz for a user. But it allows only one maximum for every other subscription event type.

Monitoring

You can display the number of subscription dialogs per SUBSCRIBE event type using the ACLI **show registration sipd subscriptions-by-user** command. You can display this information per event type, or you can show data for all event types by wildcarding the event type argument.

Enforcement Profile Configuration with subscribe-event

This section shows you how to configure an enforcement profile with a **subscribe-event** configuration. Remember that you can set up multiple **subscribe-event** configurations to correspond with the event types you want to control. It also shows you how to apply these limitations to a realm.

Setting Up Subscribe Dialog Limits

Setting up subscribe dialog limits means setting up an enforcement profile. For the sole purpose of setting up the subscription event limits, you only need to configure the name parameters and then as many **subscribe-event** configurations as you require. The enforcement profile has other uses, such as SIP SDP address correlation, so only configure the parameters you need.

To configure subscribe dialog limits:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **enforcement-profile** and press Enter.

```
ORACLE(session-router)# enforcement-profile
ORACLE(enforcement profile)#
```

4. **name**—Enter a name for this enforcement profile. You will use this name later when you apply the enforcement profile to a realm; it is the value you enter into the **enforcement-profile** parameter in the realm configuration.
5. Still in the enforcement profile configuration, type **subscribe-event** and press Enter.

```
ORACLE(enforcement profile)# subscribe-event
ORACLE(subscribe-event)#
```

6. **event-type**—Enter the SIP subscription event type for which you want to set up limits. You can also wildcard this value (meaning that this limit is applied to all event types except the others specifically configured in this enforcement profile). To use the wildcard, enter an asterisk (*****) for the parameter value.

   By default, this parameter is blank.

   > **Note:**
   >
   > The value you enter must be configured as an exact match of the event type expected in the SIP messages (except for the wildcard). Further, the value conforms to the event type BNF specified in RFC 3265.

7. **max-subscriptions**—Enter the maximum number of subscriptions allowed to a user for the SIP subscription event type you entered in the **event-type** parameter. Leaving this parameter set to 0 (default) means that there is no limit. You can set this parameter to a maximum value of 65535.
8. If you are entering multiple **subscribe-event** configurations, then you save them each by using the ACLI **done** command and then repeat Steps 6 and 7 to configure a new one. If you do not save each, then you will simply overwrite the first configuration repeatedly.

```
ORACLE(subscribe-event)# done
```

9. When you finish setting up **subscribe-event** configurations and have saved them, exit to return to the enforcement profile configuration.

```
ORACLE(subscribe-event)# exit
```

10. You also need to save the enforcement profile configuration.

```
ORACLE(enforcement profile)# done
```

Applying an Enforcement Profile to a Realm

For the Oracle Communications Session Border Controller to use the limits you have set up, you need to apply them to a realm.

To apply an enforcement profile to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this realm. This value corresponds to the **name** parameter in the enforcement profile configuration. This parameter has no default value.
5. Save and activate your configuration.

STUN Server

The Oracle Communications Session Border Controller supports RFC 3489, which defines Simple Traversal User Datagram Protocol (UDP) through Network Address Translators (NATs). Known as STUN, this lightweight protocol that allows applications to:

- Discover the presence and types of both NATs and firewalls between themselves and the public Internet
- Determine the public IP addresses allocated to them by the NAT

SIP endpoints use the STUN protocol to find out the public IP addresses and ports for SIP signaling and RTP media transport. Then they can use the address and port information to create multimedia sessions with other endpoints on the public network.

You can define STUN servers functionality on a per-realm basis, allowing you set up multiple STUN servers.

About STUN Messaging

STUN messages uses six messages, three of which are used for Binding and three of which are uses for the Shared Secret. While it supports all three Binding messages (request, response, and error), the Oracle Communications Session Border Controller does not support

the Shared Secret Request or the message integrity mechanism that relies on the shared secret. When acting as a STUN server, the Oracle Communications Session Border Controller responds to STUN binding requests in accordance with RFC 3489 and the rfc3489bis draft.

STUN messages can contain the following attributes:

Message Type	Attribute Description
MAPPED-ADDRESS	Appears in the Binding Response; contains the source IP address and port from which the Binding Request was sent to the STUN server.
XOR-MAPPED-ADDRESS	Appears in the Binding Response; contains the MAPPED-ADDRESS information encoded in a way the prevents intelligent NAT devices from modifying it as the response goes through the NAT.
SOURCE-ADDRESS	Appears in the Binding Response; contains the IP address and port from which the STUN server sent its response.
CHANGED-ADDRESS	Appears in the Binding Response; contains an alternate STUN server IP address and port, different from the primary STUN server port. The STUN client might use this attribute to perform the NAT tests described in RFC 3489.
CHANGE-REQUEST	Appears in the Binding Request; instructs the STUN server to send its response from a different IP address and/or port. The STUN client might use this attribute to perform the NAT tests described in RFC 3489.
RESPONSE-ADDRESS	Appears in the Binding Request; defines an IP address and port to which the STUN server should send its responses. Appears in the Binding Request;
REFLECTED-FROM	Appears in the Binding Response; reflects the IP address and port from which a Binding Request came. Only included when the Binding Request has used the RESPONSE-ADDRESS attribute.
UNKNOWN-ATTRIBUTES	Appears in the Binding Error; reflects the mandatory attributes in a Binding Request message that the server does not support.
ERROR-CODE	Appears in the Binding Error; indicates an error was detected in the Binding Request, and contains an error code and reason phrase.

To perform NAT discovery, the endpoint (STUN client) sends a Binding Request to the STUN server port (IP address and port) with which it is configured. The STUN server then returns either a;

- Binding Response—Allows the transaction to proceed
- Binding Error—Halts the transaction, and prompts the client to take the action appropriate to the response given in the ERROR-CODE attribute

When the transaction proceeds and the STUN server sends the Binding Response, that response contains the MAPPED-ADDRESS attribute, which contains the IP address and port from which the server received the request. The STUN client then uses the MAPPED-ADDRESS when sending signaling messages.

For example, a SIP endpoint sends Binding Requests from its SIP port to determine the public address it should place in SIP headers, like the Via and Contact, of the SIP requests it sends. When this SIP endpoint prepares to make or answer a call, it sends Binding Requests from its RTP port to find out the public address it should place in SDP included in an INVITE request or response.

STUN Server Functions on the Oracle Communications Session Border Controller

When the Oracle Communications Session Border Controller receives a STUN message, it first determines its message type. Only STUN Binding Requests are processed, and all other message types are dropped without response.

Then the Oracle Communications Session Border Controller examines the Binding Request's STUN attributes. It returns error responses if it finds any unsupported mandatory attributes. This takes the form of a Binding Error Response, containing the ERROR-CODE attribute with reason 420 (Unknown Attribute) and an UNKNOWN-ATTRIBUTES attribute with a list of the unsupported attributes. If the Oracle Communications Session Border Controller receives a Binding Request with attributes that do not belong in STUN Binding Requests, it returns the Binding Error Response with the ERROR-CODE attribute with reason 400 (Bad Request).

Next the Oracle Communications Session Border Controller determines whether to follow RFC 3489 procedures or rfc3489bis procedures. If the Transaction ID contains the STUN cookie, then the Oracle Communications Session Border Controller follows rfc3489bis procedures; if not, the it follows RFC 3489 procedures. Because it defines the procedures for testing the NAT to see what type of NAT it is, RFC 3489 procedures are most complex. Issues with reliability of those results have caused testing procedures and attributes to be deprecated in fc3489bis.

RFC 3489 Procedures

The Oracle Communications Session Border Controller (the STUN server) constructs the Binding Response and populates it with these attributes:

- MAPPED-ADDRESS and (optionally) XOR-MAPPED-ADDRESS—Containing the source IP address and port from which the server saw the request come
- SOURCE-ADDRESS—Containing the IP address and port from which the server will send the Binding Response
- CHANGED-ADDRESS—Containing the STUN server port that has a different address and different port from the ones on which the server request was received

If the Binding Request contains a RESPONSE-ADDRESS attribute, the server adds the REFLECTED-FROM attribute with the IP address and port from which the server saw the request come. Then the server sends the Binding Response to the IP address and port in the RESPONSE-ADDRESS attribute. If the RESPONSE-ADDRESS attribute's IP address and port are invalid, the server sends a Binding Error Response with an ERROR-CODE attribute reason 400 (Bad Request) to the client.

If the Binding Request contains a CHANGE-REQUEST attribute, the server sends Binding Response from the IP address and port matching the information in the CHANGE-REQUEST. The following variations can occur:

- If the IP address and port flags are set, the server selects the server port with a different IP address and different port.
- If only the IP address flag is set, the server selects the server port with a different IP address but with the same port.
- If only the port flag is set, the server selects the server port with the same IP address but with a different port.

The selected server port appears in the Binding Responses's SOURCE-ADDRESS attribute. When there is no CHANGE-REQUEST attribute, the server uses the server port on which the Binding Request was received.

Finally, the server encodes the outgoing message and sends it to the client at either:

- The destination IP address and port in the REPONSE-ADDRESS attribute, if it was present in the Binding Request.
- The MAPPED-ADDRESS.

rfc3489bis Procedures

If the Binding Request contains the appropriate cookie in its Transaction ID, the server constructs a Binding Response populated with the XOR-MAPPED-ADDRESS attribute. That attribute will contain the source IP address and port from which the server saw the request come. Then the server encodes and sends the message to the client from the IP address and port on which the request was received. The message is sent to the IP address and port from which the request came.

Monitoring

- STUN Server Statistics—You can display statistics for the STUN server using the ACLI **show mbcd stun** command when the STUN server has been enabled. However, if the STUN server has not been enabled since the last system reboot, the command does not appear and no statistics will be displayed.
- STUN Protocol Tracing—You can enable STUN protocol tracing two ways: by configuration or on demand.
  - By configuration—The Oracle Communications Session Border Controller's STUN protocol trace file is called stun.log, which is classified as a call trace. This means that when the system configuration's call-trace parameter is set to enabled, you will obtain STUN protocol information for the system. As with other call protocol traces, tracing data is controlled by the log-filter in the system configuration.
  - On demand—Using the ACLI **notify mbcd log** or **notify mbcd debug** commands, you enable protocol tracing for STUN. Using **notify mbcd debug** sets the STUN log level to TRACE. You can turn off tracing using the **notify mbcd onlog** or **notify mbcd nodebug** commands. Using **notify mbcd nodebug** returns the STUN log level back to its configured setting.

STUN Server Configuration

You configured STUN servers on a per-realm basis, one server per realm. To support that various NAT tests it describes, RFC 3489 requires that two different IP addresses and two different UDP port numbers be used for each server. So your STUN server will listen on a total of four STUN server ports. Although newer work does away with this requirement, the Oracle Communications Session Border ControllerC supports it for the purpose of backwards compatibility.

For each realm configuration with an enabled STUN server, untrusted ACL entries will be added to forward all packets received on the four STUN Server Port.

To enable STUN server support for a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. **stun-enable**—Set this parameter to **enabled** to turn STUN server support for this realm on. This parameter defaults to **disabled**, meaning STUN server support is off.
5. **stun-server-ip**—Enter the IP address for the primary STUN server port. The default for this parameter is 0.0.0.0.
6. **stun-server-port**—Enter the port to use with the **stun-server-ip** for primary STUN server port. The default is 3478.
7. **stun-changed-ip**—Enter the IP address for the CHANGED-ADDRESS attribute in Binding Requests received on the primary STUN server port. This IP address must be different from than the one defined for the **stun-server-ip** parameter. The default for this parameter is 0.0.0.0.
8. **stun-changed-port**—Enter the port combination to define the CHANGED-ADDRESS attribute in Binding Requests received on the primary STUN server port. The default for this parameter is 3479.
9. Save and activate your configuration.

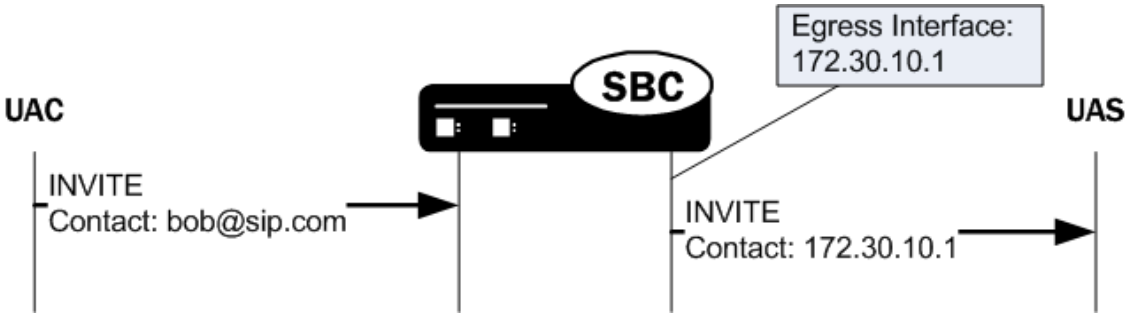
SIP GRUU

SIP Globally Routable User Agent (UA) URIs (GRUU) are designed to reliably route a SIP message to a specific device or end user. This contrasts with a SIP AoR which can refer to multiple UAs for a single user, thus contributing to routing confusion. The Oracle Communications Session Border Controller can perform different behaviors when it finds SIP GRUUs in Contact headers.

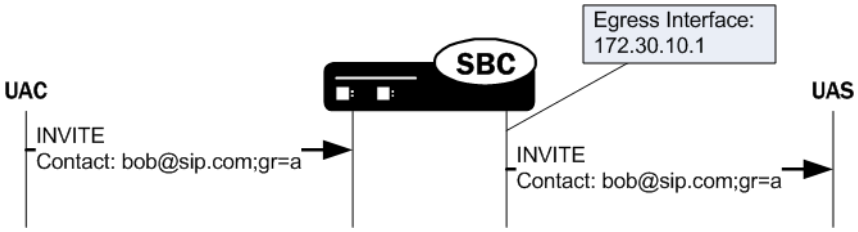
User agents supporting GRUU include a GRUU-identifying parameter in the Contact header of dialog forming and target refresh requests. The Oracle Communications Session Border Controller scans for the GRUU parameter in the Contact header either when the message is received on a realm or interface configured for registered endpoints or when the pass-gruu-contact parameter is enabled. Configure an interface for registered endpoints by enabling nat-traversal or registration caching.

Contact Header URI Replacement

When no GRUU is encountered in the contact header, and when a SIP message is forwarded to the egress realm, the contact header's URI is replaced with the Oracle Communications Session Border Controller's egress interface. For example:

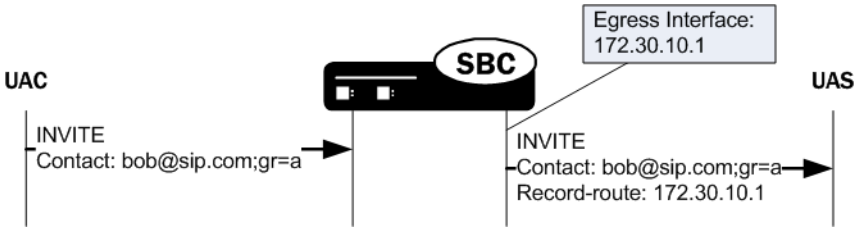


When the Oracle Communications Session Border Controller forwards a request where the original Contact header contains a GRUU, the contact header's URI is forwarded unchanged on the egress side of the call. For example:



Record-Route Addition

When the request is forwarded to a realm where the endpoint's registrar does not exist, the Oracle Communications Session Border Controller adds a Record-Route header containing the egress SIP interface address. This causes subsequent replies or requests addressed to the GRUU to be routed through the Oracle Communications Session Border Controller first.



When the request is forwarded to the realm where the registrar exists, adding the Record-Route header is unnecessary and does not occur. This is because subsequent requests are directed to the registrar which will ultimately forward them to the Oracle Communications Session Border Controller using the registered Contact in the Request-URI.

GRUU URI Parameter Name

The Oracle Communications Session Border Controller scans for a gr URI parameter in the contact header to identify it as a GRUU as defined in the ietf draft[2]. The Oracle Communications Session Border Controller can be configured to scan for a gruu URI parameter in the contact header too. This alternate behavior is enabled with the scan-for-ms-gruu option and is used to interact with the Microsoft Office Communications Server unified communications product. When scan-for-ms-gruu is enabled, the Oracle Communications Session Border Controller scans first for the gruu URI parameter. If not found, it then scans for gr URI parameter.

SIP GRUU Configuration

This section shows you how to configure the GRUU support for non-registered contacts. Enabling GRUU functionality to parse for gr URI parameter rather than the IETF standard gruu parameter is also provided.

To configure SIP GRUU functionality:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding this feature to an existing configuration, you need to select the configuration (using the ACLI **select** command) before making your changes.

4. **pass-gruu-contact**—Set this parameter to enabled to parse for gr URI parameter in the contact header in non-registered endpoints' messages and then pass the messages thought the system.
5. **options**—Set the options parameter by typing **options**, a Space, the option name **scan-for-ms-gruu.** This option forces the Oracle Communications Session Border Controller to first scan for the gruu URI parameter, then the gr URI parameter.
6. Save and activate your configuration.

SIP SIP-I Interworking

The Oracle Communications Session Border Controller (SBC) supports the Interworking function (IWF) between SIP and SIP-I networks by including ISUP message information in the body of SIP messages, and formatting SIP message information for SIP-I, from which subsequent devices use for ISUP. The SBC has broad support for ITU Specification Q.1912.5, which defines SIP-I. This specification defines ISUP message encapsulation and the mapping between SIP headers and ISUP parameters. This ITU specification is considered a superset of the IETF's SIP-T specification. The SBC also complies with IETF specifications on the use of MIME within SIP messages.

Within the context of SIP SIP-I IWF, the Oracle Communications Session Border Controller (SBC) operates between the SIP network and a Media Gateway Controller Function (MGCF) generating and interpreting SIPI as an interim step for originating and/or terminating services between SIP and ISUP. The configurable categories of interworking information include:

- Signaling Message and Parameter IWF
- Diversion Information IWF
- ISUP Version IWF

You configure **sip-profiles** and **sip-isup-profiles** and apply them to **session-agent**, **sip-interface** or **realm** to enable interworking. These configurations determine the base actions taken on the target traffic. The SBC supports three methods refining the interworking. You can them use the following configuration methods to refine the interworking by specifying information, such as header parameter detail, for ingress and/or egress traffic:

- Interworking with Header Manipulation Rules (HMR)
- Interworking with Session Translation
- Native SIP SIP-I Interworking Support

You configure the SBC to use one or a combination of these methods. Note the processing order in which the SBC executes these functions, explained in this document, as each subsequent function operates on the results of the prior function.

By default, you configure HMR for all SIP to SIP-I interworking, with the exception of IAM interworking. This is a legacy means of configuring this interworking, and is widely deployed.

Session translation configuration is similar to HMR, but is for less complex message manipulation. You configure session translation elements to perform specific text changes on a limited number of SIP and/or SIP-I objects using offsets to find your target objects, and add, change or delete the text you specify. You can also use session translation for called and calling number manipulation for SIP to SIP signaling.

Native SIP SIP-I interworking features automatically perform interworking on messages in addition the IAM. For native SIP SIP-I interworking, you simply enable it and the SBC performs the changes described herein. Advantages of Native IWF configuration include:

- Avoids the complexity of HMR configuration, reducing implementation time and resources
- Provides session statefulness within the SBC for these interworked sessions
- Overcomes the HMR limitation of HEX to decimal arithmetic, simplifying reason code mapping
- Allows better system performance

Regardless of the method you choose, at a high level, for SIPI to SIP interworking, the SBC constructs the SIP headers based on the SIPI information, then removes the entire SIPI MIME block. For the SIP to SIPI direction, the SBC interworks IAM parameters as described in this document and inserts the block.

> **Note:**
>
> The SBC does not comply with every aspect of ITU Specifications Q.1912.5, which defines SIP-I.

SIP-SIPI IWF Operational Overview

The SBC uses the ITU Q.1912.5 specification for guidance on this IWF. The following sections describe key SBC behavior that is not defined by any external specification.

Function Processing Order

There are three configurable objects that you can use to perform SIP-SIPI IWF. When designing your IWF, ensure you configure each object using the understanding of which functions the SBC has already performed. Doing this ensures that your configuration performs each step on the data based on which functions are already completed.

For ingress, the SBC acts on these objects in the following order:

1. HMR rules
2. **in-translationid** rule(s)
3. **sip-isup-profile**

For egress, the SBC acts on these objects in the following order:

1. **sip-isup-profile**
2. **out-translationid** rule(s)
3. HMR rules

The **sip-isup-profile** is applicable only on the SIP-I side of the call.

IAM Message Interworking

The SBC performs IAM message insertion and deletion, using the version specified in that applicable **sip-profile**, without requiring HMR when your configuration invokes a valid **sip-isup-profile**. This includes diversion information. The SBC also interworks the IAM parameters documented herein. Extended IAM support, including number portability interworking support requires that you configure the native inter-working method.

Additional Message Interworking

When enabled, Native interworking mode interworks additional ISUP messages:

- ACM
- CPG
- ANM
- CON
- REL
- RLC

Applicable parameter interworking is documented below.

BYE and REL

The SBC inserts an REL as an application/isup “content-type” with the header parameters based on the applicable **sip-isup-profile**, and formats the message according to the ISUP version configured in the **sip-isup-profile**.

In addition, if a Reason header, as described in IETF RFC 6432, is included in a 4xx, 5xx, 6xx response, the SBC maps the Cause Value of the Reason header to the ISUP Cause Value field in the REL message.

200OK and RLC

The SBC inserts an RLC as an application/isup content-type, with the header parameters based on the applicable **sip-isup-profile**. It does not include any ISUP parameters in this message.

CON Message Workaround

It is possible for an SIP-I node to receive a CONNECT (CON) without receiving any Address Complete Message (ACM) or Call Progress (CPG) message, such as when the call is answered automatically. In these cases, the SIP user does not receive a 18x provisional

response, and does not hear ringtone before the callee answers. To support these cases, the SBC inserts a CON message into the outgoing 200 OK response.

High Availability

By design, the SBC replicates session level data to the standby after the 200 OK message. Because the SBC processes ISUP signaling prior to the 200 OK, there is no HA replication of these messages. Specifically, the ACM and CPG get processed with provisional (18x) responses and the ANM or CON ISUP get processed with the 200 OK response. If an HA pair fails over between the IAM and ANM messages, the SBC cannot maintain state for these sessions.

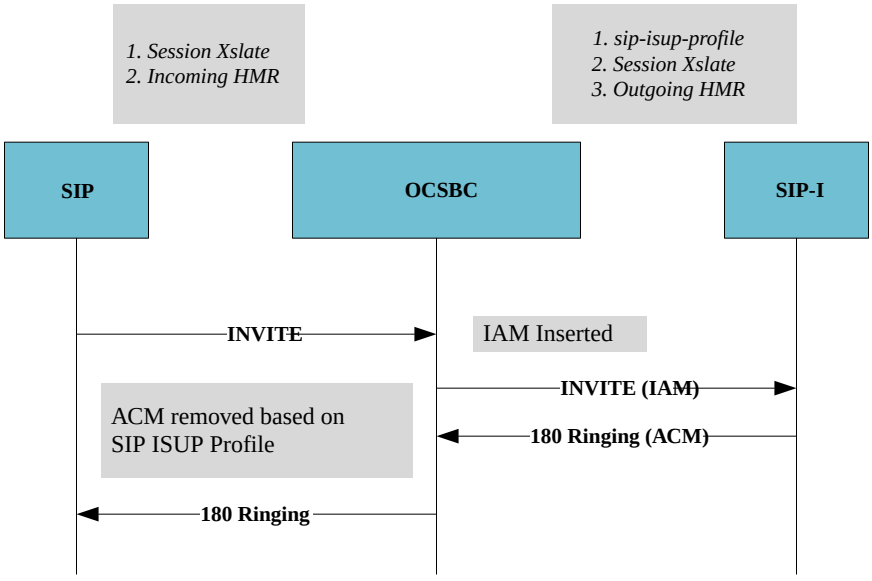
State Machine

The SBC maintains state for SIP to SIP-I sessions only when configured for native ISUP message interworking; not for the HMR method.

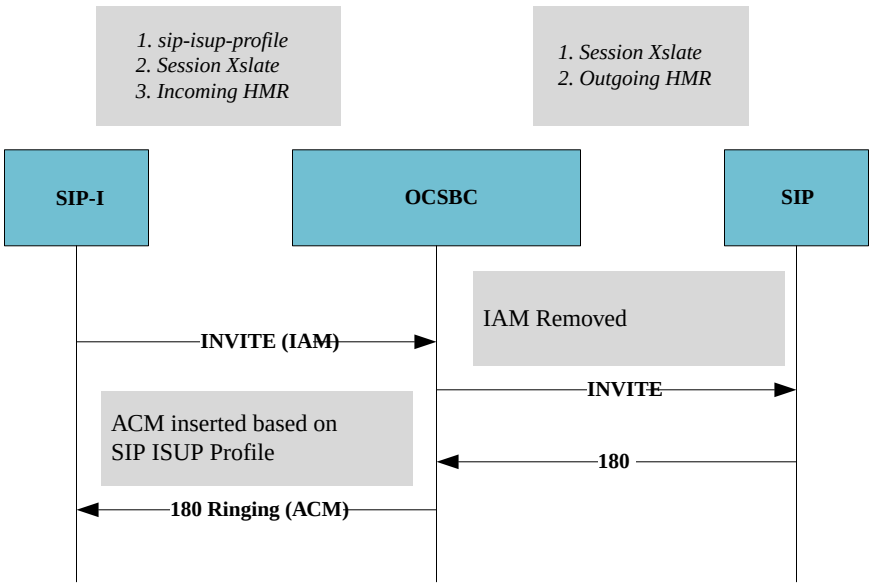
SIP-SIPI Interworking Call Flows

When configured for SIP-SIP-I interworking, the SBC inserts and removes ISUP signaling information into SIP signaling messages as MIME bodies. You can also configure manipulation and/or translation rules to refine your interworking. The diagrams below present the order in which the SBC processes the relevant configuration objects, resulting in incremental changes. You must configure interworking manipulation ensuring that the changes caused by the latter processing are based on the results of preceding processing.

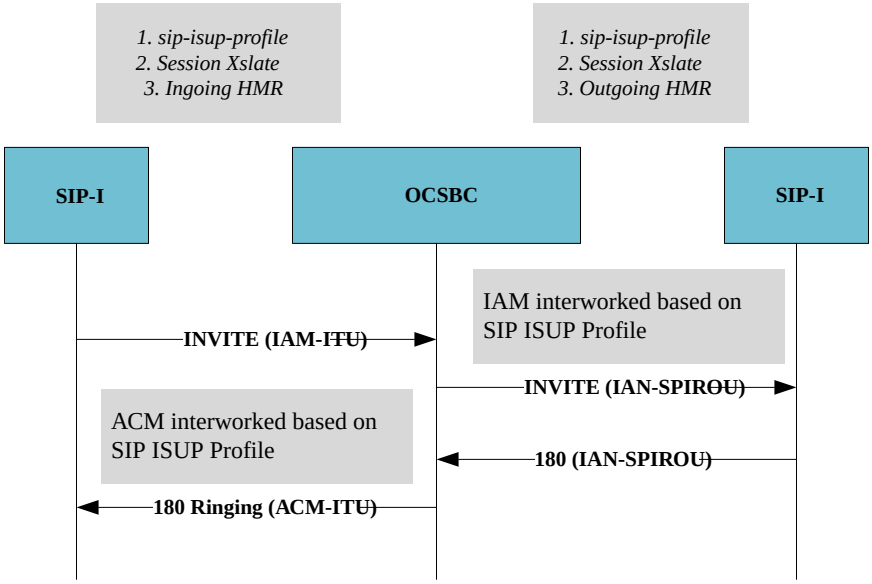
The diagram below shows a call initiated by a SIP endstation. The SBC interworks the call to SIP-I using your configuration and passes the call to the media gateway for SIPI to ISUP interworking. The SBC inserts the IAM as an ISUP MIME body part in the INVITE. The SBC also removes ACM presented by the SIP-I network from the 180 Ringing message.



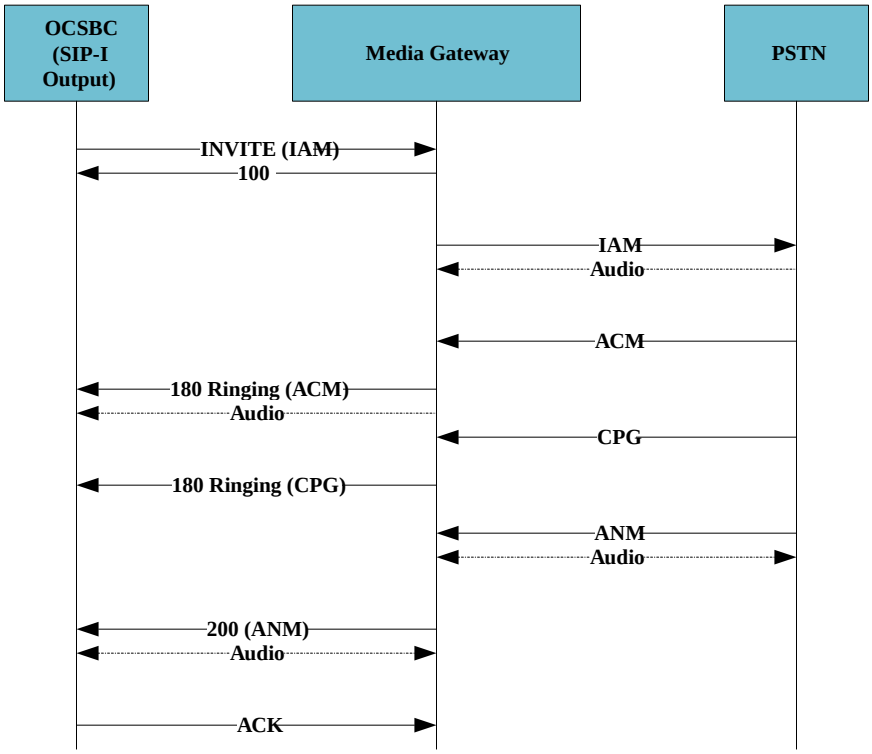
Similarly, the diagram below presents the opposite IWF behavior wherein the SBC detects an INVITE coming from SIP-I, removing the IAM and inserting the ACM, as appropriate.



Similarly, the diagram below presents the ISUP version (format) behavior wherein the SBC detects an INVITE coming from SIP-I, interworking the ITU-IAM to SPIROU-ACM, as appropriate.



To complete the picture, note the SBC providing SIP-I output to the media gateway, which interworks between SIP-I and the PSTN.



Refer to ITU-T Recommendation Q.1912.5 for SIP to ISUP message mapping tables to verify expected responses and their interworking.

Base SIP-SIPI IWF Functionality

By default, the SBC does not perform SIP-SIPI interworking. You configure one or more **sip-isup-profile** and **sip-profile** sub-elements within the **session-router** element, and then specify the profiles to use on a **session-agent**, **sip-interface** or **realm**. The SBC follows that order for the configuration's operation precedence.

- **sip-isup-profile**—This object specifies multiple SIP-SIPI IWF controls for the object to which it is applied. At a minimum, this parameter specifies the ISUP version for parameter interworking. Additional IWF configuration includes ISUP version interworking and disabling 183 message interworking.
- **sip-profile**—This object includes the **redirection** parameter that defines the basic SIP-SIPI insertion and removal behavior for the object to which it is applied. (You also use this object to configure functions not related to this IWF.)

The SBC performs the following insertion and removal actions based on the ingress and egress **sip-profile**, **redirection** settings:

- When you set the ingress **sip-profile**, **redirection** to **none** and the egress to:
  - **none**—Perform no action
  - **isup**—Add ISUP content and, if INVITE, generate the IAM ignoring diversion headers
  - **diversion**—Perform no action
- When you set the ingress **sip-profile**, **redirection** set to **isup** and the egress to:
  - **none**—Remove ISUP content
  - **isup**—Convert ISUP version if needed and enforce sip-isup-profile specific attributes

  - **diversion**—Remove ISUP content and, if INVITE, generate diversion header from the IAM
- When you set the ingress **sip-profile**, **redirection** set to **diversion** and the egress to:
  - **none**—Perform no action
  - **isup**—Add ISUP content and, if INVITE, generate IAM from the diversion headers.
  - **diversion**—Perform no action

You further specify your interworking behavior using other controls, including HMR, Session Translation and Native Interworking.

SIP-SIPI IWF using HMR

By default, interworking SIP-SIPI messaging requires that you create HMRs that recognize the message types and build messages appropriate for egress. You create these HMRs for each expected message type, with the exception of IAM messages. The Oracle Communications Session Border Controller (SBC) performs IAM interworking when it recognizes that a **sip-isup-profile** is applied. The *Header Manipulation Rules Resource Guide* includes full instructions on how to build all HMRs. This guide's MIME Support chapter includes a section on *HMR for SIP-ISUP* that directly addresses how to create HMRs for this interworking.

The SBC's HMR functionality can operate on ISDN user party (ISUP) binary bodies. Using the same logic and mechanisms that are applied to SIP header elements, HMR for SIP-ISUP manipulates ISUP parameter fields and ISUP message parts. You can create MIME rules that function in much the same way the SIP header rules do; whereas SIP header rules can change the specific headers of a SIP message, MIME rules can manipulate targeted body parts of a SIP message.

SIP-SIPI IWF Using Session Translation

You can use a feature called Number Translation function to perform simple changes to calling or called numbers for SIP-SIPI IWF calls. Number Translation uses **session-translation** elements to specify these translations. You use **session-translation** elements for SIP-SIPI IWF similarly to number translation to manipulate SIP-SIPI interworking information. Refer to "Number Translation" to for more on how Number Translation works and for additional instruction on **session-translation** configuration.

The **session-translation** element presents the list of SIP and ISUP objects that you can manipulate with session translation:

- **rules-calling**—Manipulates the SIP from header
- **rules-called**—Manipulates the SIP to header
- **rules-asserted-id**—Manipulates the SIP P-Asserted-Id header
- **rules-redirect**—Manipulates the SIP Diversion headers
- **rules-isup-cdpn**—Manipulates the ISUP Called Party Number parameter
- **rules-isup-cgpn**—Manipulates the ISUP Calling Party Number parameter
- **rules-isup-gn**—Manipulates the ISUP Generic Number parameter
- **rules-isup-rdn**—Manipulates the ISUP Redirecting Number parameters
- **rules-isup-ocn**—Manipulates the ISUP Original Called Number parameters
- **rules-history-info**—Manipulates the SIP History-Info header

The SBC only applies session-translation and translation-rule processing to a limited number of out-of-dialog requests, including:

- INVITE
- CANCEL
- REFER
- OPTIONS
- SUBSCRIBE

The SBC does not manipulate any header within in-dialog requests using session-translation or translation-rules.

As with number translation, the configuration steps for this process include:

1. Define **translation-rules**.
2. Apply **translation-rules** to the appropriate **session-translation** parameters.
3. Apply the applicable **session-translation** object on session-agents or realms. Apply these translations based whether you are targeting ingress or egress calls (**in-translationid** / **out-translationid**) and based on whether the call leg is SIP or SIPI.

Refer to "SIP-ISUP IWF Operational Detail" for processing sequences to ensure you are performing **session-translation** based on the expected state of each message object.

ISUP Session Translation and NOA Interworking

The SBC builds the ISUP-based number before translation, referring to the ISUP nature of address. Procedures performed at the SIPI egress include:

- Regardless of ISUP session translation configuration:
  - If the received URI userpart starts with +, the SBC removes it and sets the NOA to International.
- When you have configured any ISUP session translation:
  - If the received URI userpart starts with 0 (zero), the SBC removes it and sets the NOA to National.
  - If the received URI userpart does not start with either + or 0, the SBC does not make any change to the NOA. This case would most likely be a SIPI to SIPI interworking wherein the incoming NOA is neither national or international.

Nature of address	Example number	Number before translation
International	1234	+1234 (starts with +)
National	1234	01234 (starts with zero)
Other	1234	1234 (others)

After the translation, the ISUP parameters are filled back with the right nature of address.

Number after translation	New nature of address	Number in ISUP
+1234 (starts with +)	International	1234
01234 (starts with zero)	National	1234
1234 (others)	Do not change (keep previous one)	1234

Review the following conversion example.

Source ISUP called party number	Translation-rule #1 (convert to NATIONAL) called from rules-isup-cdpn	Result
• Nature of address INTERNATIONAL • Number 26226244555656	• Action replace • Delete-string +262 • Add-string 0	• Nature of address NATIONAL • Number 26244555656

Additional Controls on CDR Population

By default, the SBC does not populate CDRs with call information that ingress **translation-rules** and ISUP precedence values may change. This can present problems in environments where you use **translation-rules** to perform number normalization and the CDR fields are expected to be normalized. Also by default, the SBC populates CDRs prior to SIP-SIPI interworking. You can configure the SBC to include this non-default information in CDRs using the **sip-config** option **late-cdr-population**.

SBC can manipulate the following SIP header through session-translation:

- User part of To Header
- User part of Request-URI
- User part of From Header
- User part of Diversion Headers
- User part of P-Asserted-Id headers
- User part of History-Info headers

When configured, the SBC populates the following attributes with the values derived after ingress session-translation manipulation in CDRs.

- Calling-Station-Id
- Called-Station-Id
- Acme-P-Asserted-ID
- Acme-SIP-Diversion
- Acme-History-Info
- Acme-Primary-Routing-Number

> **Note:**
>
> If you have configured the accounting-config's **generate-start** parameter to INVITE, the SBC does not include ISUP precedence changes in the START CDR. Set **generate-start** to **OK** to include this information.

Set the options parameter by typing **options**, a space, the option name **late-cdr-population** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-config)# options + late-cdr-population
```

Header Population in CDRs

By default, the SBC does not populate CDRs with all diversion header values. You can set the **multiple-diversion-header-for-accounting** option in the **sip-config** to update the Diversion CDR field with all the Diversion headers present in SIP message. This option can be used in conjunction with the **late-cdr-population** option.

```
ORACLE(sip-config)# options + multiple-diversion-header-for-accounting
```

Header Integrity in CDRs

By default, the SBC may truncate diversion and history-info headers during insertion into CDRs. This is because the default CDR field character length is 246 characters. You can set the **cdr-attr-size-limit** option in the **sip-config** to ensure that no diversion and history-info CDR field are truncated.

When enabled, this option behaves differently for diversion and history-info headers:

- For diversion, the SBC removes the last headers
- For history-info, the SBC removes the first headers

This option can be used in conjunction with the **multiple-diversion-header-for-accounting** option.

```
ORACLE(sip-config)# options + cdr-attr-size-limit
```

If you type an option without the plus sign, you overwrite any previously configured options. Prepend new options with a plus sign, as shown in the previous example, to add new options to an options list.

SIP-SIPI IWF using Native Processing

The Oracle Communications Session Border Controller (SBC) can natively interwork ISUP parameters to and from SIP, as defined by 3GPP TS 129.163 *Interworking between IP Multimedia (IM) Core Network (CN) Subsystem and Circuit Switched (CS) Networks*, *Control Plane Interworking*. Support for the capabilities of SIP and SDP that are interworked with BICC or ISUP are defined in 3GPP TS 24.229. The information the SBC interworks is a subset of all possible interworking, but results in successful sessions for the vast majority of interworked calls. Configure deployments that need additional interworking with HMR.

You configure the SBC to natively interwork specific information in specific messages by setting the **sipi-behavior** option within the **sip-config** to **iam-anm**. This configuration also allows the SBC to achieve session stateful awareness for the applicable sessions.

The SBC supports native interworking for the following ISUP messages:

- Initial Address Message (IAM)—Interworks with SIP INVITE messages

> **Note:**
>
> The SBC performs IAM insertion regardless of the **sipi-behavior** setting. IAM removal, as well as all other ISUP message interworking requires the **iam-anm** setting.

- Address Complete Message (ACM)—Interworks with SIP 180 and 183 messages
- Call Progress Message (CPG)—Interworks with SIP 180 and 183 messages, as well as Re-INVITE and UPDATE messages, to support Hold/Resume flows
- Answer Message (ANM)—Interworks with SIP 200 OK messages
- Connect (CON), which is an alternate to ANM when no ACM is received—Interworks with the 200 OK of an INVITE
- Release (REL)—Interworks to SIP BYE and, if applicable, error responses
- Release Complete (RLC)—Interworks to the 200 OK of the BYE

IAM Interworking Support

When configured for IWF, the SBC supports SIP-SIPI IWF in both directions. For SIPI to SIP interworking, the SBC constructs the SIP headers based on the SIPI information, then removes the entire IAM block. For the SIP to SIPI direction, the SBC interworks IAM parameters as described here and inserts the block.

Assuming you have configured a **sip-isup-profile** and a **sip-profile** on the applicable interfaces, the SBC performs this interworking regardless of the **sipi-behavior** configuration. It is recommended that you do not configure IAM interworking using HMR or session translation.

The SBC supports the insertion of the IAM message based on the configured **redirection** parameter in the applicable **sip-profile** and the **version** field in the applicable **sip-isup-profile**. The SBC inserts the body after any application/sdp body parts, if that body is already in the **INVITE**.

When configured for native IWF, the SBC interworks the following IAM parameters:

- Called Party Number
- Nature of connection Indicator
- Forward Call Indicators
- Calling Party Category
- Transmission Medium Requirement
- Calling Party Number Parameter
- Generic Number

Called Party Number

For **Called Party Number**, mapping includes:

- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
  - **Special Number**, if there is a phone-context parameter in the request-uri user part, and that number does not start with a **+** or a **0** (encoded to 1110011).
- Internal Network Number Indicator, with a sole value of 1, indicates routing to an internal network number is not allowed.
- Numbering Plan, which is always **ISDN**.

Nature of connection Indicator

For Nature of connection Indicator,

- **No Satellite Circuit**—If the SIP messaging includes the Satellite Indicator
- **No continuity Check**—If the SIP messaging includes the Continuity Check indication
- **Outgoing Echo Device Included**—If the SIP messaging includes the Echo control device indicator

Forward Call Indicators

For Forward Call Indicators, the SBC inserts the following values into the IAM, based on the SIP indicators listed:

- **No end-to-end method available**—If the SIP messaging includes the End-to-end method indicator
- **Interworking encountered** —If the SIP messaging includes the Interworking indicator
- **No end-to-end information available**—If the SIP messaging includes the End-to-end information indicator
- **ISDN user part/BICC not used all the way**—If the SIP messaging includes the ISDN user part/BICC indicator
- **ISDN user part/BICC not required all the way**—If the SIP messaging includes the ISDN user part/BICC preference indicator
- **Non-ISDN Access**—If the SIP messaging includes the ISDN access indicator
- **No Indication**—If the SIP messaging includes the SCCP method indicator
- **Number not translated**—If the SIP messaging includes the Ported number translation indicator
- **No QoR routing attempt in progress**—If the SIP messaging includes the query on release attempt indicator

Calling Party Category

The SBC always inserts the value **Ordinary calling subscriber** for **Calling Party Category**, encoded as 000010101, regardless of the SIP signaling.

Transmission Medium Requirement

For Transmission Medium Requirement (TMR), the SBC inserts the TMR, coding it based on the content of the incoming SDP:

- **64 kbit/s unrestricted**—If the **Clearmode** codec is present in incoming SDP (encoded to 00000010).
- **3.1 kHz audio—**For all other cases (encoded to 00000011).

Calling Party Number

The IAM **Calling Party Number** equates to a combination of the ISUP **Address of Signal** and Address Presentation Restricted Indicator (**APRI**). The SBC maps this data based on the SIP **PAI**, **FROM URI**, and **Privacy Value**. as shown in the following table.

P-Asserted-Identity	From URI	Privacy Value	Address Of Signal	APRI
Not Present	Non E.164	Any	Omit	Omit
Not Present	Including E.164	Absent or with value “none”	Empty	Unavailable
Not Present	Including E.164	Value different than “none”	Empty	Unavailable
Present	Any	Absent or with value “none”	P-Asserted-Identity userpart	Presentation allowed
Present	Any	Value different than “none”	P-Asserted-Identity userpart	Presentation restricted

The SIP **P-Asserted-Identity** and **Privacy** headers may or may not be present in the INVITE. In addition the SIP **From** header is always be present, but it may contain an anonymous URI, often formatted as **anonymous@anonymous.invalid**.

In addition, the SBC always fills in other parameters as follows:

- Nature of address:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- Numbering Plan—Always **ISDN**.
- Screening Info—Always **Network Provided**.
- Number Incomplete Indicator— Always set as **Complete**.

Generic Number

The **Generic Number** parameter is mapped the similarly to the **Calling Party Number**. The table below shows this mapping.

P-Asserted-Identity	From URI	Privacy Value	Address Of Signal	APRI
Not Present	Including E.164	Any	Omit	Omit
Any	Non E.164	Any	Omit	Omit
Present	Including E.164	Absent or with value “none”	From userpart	Presentation allowed
Present	Including E.164	Value different than “none”	From userpart	Presentation restricted

The SIP **P-Asserted-Identity** and **Privacy** headers may or may not be present in the INVITE. In addition the SIP **From** header is always be present, but it may contain an anonymous URI, often formatted as **anonymous@anonymous.invalid**.

In addition, the SBC always maps the Generic Number, Number Qualifier Indicators to Additional Calling Party number parameters as follows:

- Nature of address:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- Numbering Plan—Always **ISDN**.

- Screening Info—Always user provided and not verified.
- Number Incomplete Indicator— Always set as **Complete**.

Number Portability Interworking Support for IAM

When the optional Number Portability Routing Number (NPRN) is present and supported, the SBC uses these procedures to code the ISUP Called Party Number.

For backward compatibility, and to handle deployments where this behavior is not expected (ITU- T1912.5 does not include this requirement), you set the SBC to use this behavior by configuring the **portability-method** parameter to **concatenate** within the **sip-isup-profile**. This attribute allows the SBC to map the different behaviors related to number portability in TS29.163.

In addition, the **sip-isup-profile** element includes the **country-code** parameter. When you have configured the **portability-method** parameter to **concatenate** and the SBC receives messages that include NPRN in the request URI of an incoming INVITE, the SBC formats the ISUP called number based on the text you configure in the **called-party number** parameter. The **called-party number** parameter accepts any string, but is typically configured with a country code. The format (coding) rules are presented below.

Called Party Number

If an NPRN is present as **rn** param in the **request-uri** of the incoming **INVITE**, and an NP Database Dip Indicator (NPDI) parameter is present, the SBC codes the Called Party Number as follows:

- Address Signal—The SBC inserts the following values and format into the IAM, based on the conditions described:
  - If the SIP called number starts with the same country-code as is in the egress **sip-isup-profile**::country-code:
    The format presented is **+CC** + **NPRN** + **Called party number** (NOA = International)
  - If the SIP called number starts with a different country-code as is in the egress **sip-isup-profile**::country-code, behave the same as when the portability method is configured as **none** (transparent), and:
    The format presented is **+CC2** + **Called party number** (NOA = International)
  - If the SIP called number does not start with **+**:
    The format presented is **NPRN** + **Called party number** (NOA = National)
- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
  - **Special Number**, if there is a phone-context parameter in the request-uri user part, and that number does not start with a **+** or a **0** (encoded to 1110011).
- Internal Network Number Indicator, with a sole value of 1, indicates routing to an internal network number is not allowed.
- Numbering Plan, which is always **ISDN**.

Examples of ISUP called party number results, assuming you set the **sip-isup-profile**, **country-code** to +33, include:

- If the Request-Uri is sip:+33454556677 and the **rn** is 10432;npdi@company.com
  Result: 3310432454556677 (NOA=International)

- If the Request-Uri is sip:+44454556677 and the **rn** is 10432;npdi@company.com
  Result: 44454556677 (NOA=International)
- If the Request-Uri is sip:0454556677 and the **rn** is 10432;npdi@company.com
  Result: 10432454556677 (NOA=National)
- If the Request-Uri is sip:0454556677 and the **rn** is 10432@company.com
  Result: 454556677 (NOA=National)
- If the Request-Uri is sip:+33454556677 and the **rn** is npdi@company.com
  Result: 33454556677 (NOA=International)

Number Portability Forward Information

The SBC codes this parameter as follows:

- If the NPDI is present, the SBC behaves as follows, in conjunction with the following conditions:
  - NPRN present—The SBC performs a number portability query for the called number, and inserts the ported called subscriber.
  - NPRN absent—The SBC performs a number portability query for the called number, and inserts the non-ported called subscriber.
- If the NPDI is absent—The SBC does not perform a number portability query.

BYE and REL Interworking Support

The SBC inserts an **REL** as an application/isup “content-type” with the header parameters based on the applicable **sip-isup-profile**, and formats the message according to the ISUP version configured in the **sip-isup-profile**.

If a Reason header, described in IETF RFC 6432, is included in the **BYE** request, the SBC maps the Cause Value of the Reason header to the ISUP Cause Value field in the **REL** message. If not included , the SBC inserts the following values:

- Location—Network beyond interworking point
- Cause Value—16 (Normal Call Clearing)

For BYEs received from the SIP-I side, the SBC inserts a Reason Header into the SIP body. This also requires that you have enabled the **add-reason-header** within the **sip-config**.

200OK (of BYE) and RLC Interworking Support

The SBC inserts an **RLC** as an application/isup content-type, with the header parameters based on the applicable **sip-isup-profile**. It does not include any ISUP parameters in this message. The SBC also formats the message according to the ISUP version configured in the **sip-isup-profile**.

Interworking 183 Message Content

The SBC supports interworking of 183 message content in both directions, creating the 183 on the SIP side and creating the ACM or CPG messages on the ISUP side. Via configuration, you can disable interworking of 183 messages or set the SIP to SIP-I interworking to be based on the presence of and the value of any PEM header present in a 183 Session Progress message.

Some standards do not require you to map 183 messages into ACM/CPG messages. When disabled, the **iwf-for-183** parameter in the applicable **sip-isup-profile** supports these environments. The default value is enabled. This parameter operates for native interworking, requiring that you set the **sipi-behavior** option in the **sip-config** to **iam-anm**.

Enabling vs. Disabling 183 Interworking

When enabled, the SBC:

- Inserts ACM to first 183 or 180
- Inserts a CPG into any subsequent 18x
- Inserts the REL into the SIP BYE
- Inserts the RLC into the 200 OK of the BYE, and
- Inserts the ANM into the 200 OK

When disabled, the SBC:

- Does not insert any ISUP body into 183s
- Inserts the ACM into the 180 instead of any 183
- Inserts a CPG into all further 180s, and
- Inserts the ANM into the 200 OK

If the session setup sequence uses only 183s, the SBC inserts a CON into 200 OK instead of an ANM.

Controlling 183 Messages using PEM

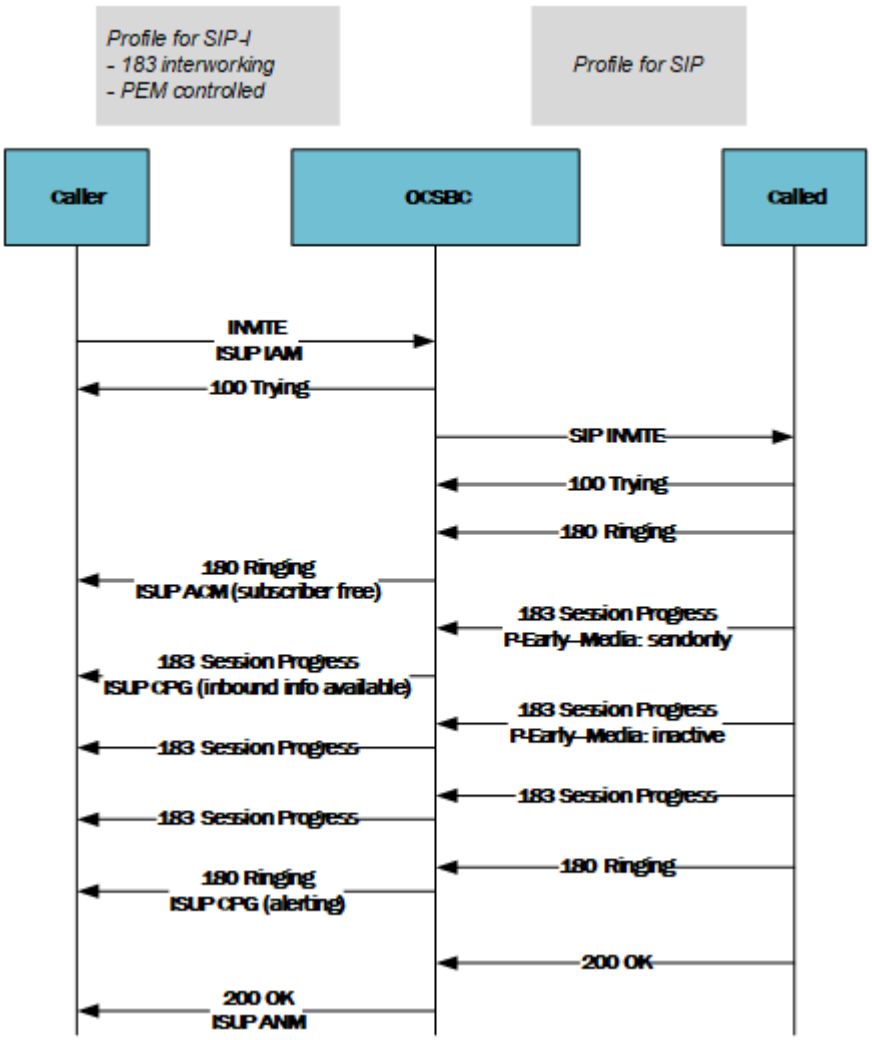
The SBC supports configuration that causes it to issue 183 Session Progress messages with or without an ISUP ACM toward the SIP-I side of an interworked call.

When the value of **iwf-for-183** parameter in **sip-isup-profile** configuration is set to **pem-controlled**, the SBC interworks 183 Session Progress responses received from IMS node as follows:

- If the received 183 Session Progress message contains P-Early-Media (PEM) header with value set to sendonly or sendrecv, the SBC behaves as if **iwf-for-183** is **enabled**.
- If the received 183 Session Progress message does not contain a PEM, the SBC behavior as if the **iwf-for-183** parameter is **disabled**.
- If the received 183 Session Progress message contains a P-Early-Media header with a value other than sendonly or sendrecv, the SBC behavior is the same as if the **iwf-for-183** parameter is **disabled**.

The call flow depicts the SBC configured to **pem-controlled**, receiving three 183 Session Progress messages, and forwarding messaging to the SIP-I side based on the contents of each 183:

1. Contains a PEM header with a value of sendonly, resulting in a 183 Session progress message that includes an ISUP CPG with inbound info available.
2. Contains a PEM header with a value that of inactive, resulting in a 183 Session progress message with no CPG.
3. Does not contain a PEM header, resulting in a 183 Session progress message with no CPG.



Interworking ACM/CPG Parameters for 180 and 183 Messages

When configured for native interworking, the SBC provides a fixed set of responses when presented with SIP to SIP-I and SIP-I to SIP-I calls that include ISUP progress messages, including user-configured ACM and CPG parameters. (The SBC only interworks an inbound-annoucement into a PEM header if it is present in the **extract-isup-params** list.) The SBC inserts interworked information into 180 and 183 messages.

For both SIP to SIP-I and SIP-I to SIP-I calls, the SBC checks for the presence of an in-band announcement in ISUP progress messages. It also checks to see if the `extract-isup-params` configuration contains the inband-announcement value. If it finds both these to be true, it interworks PEM headers while generating SIP 180 and 183 messages as follows:

- ACM message—The SBC confirms the presence of an in-band announcement if it finds an optional BCI parameter with the In-band information indicator bit set to 1.
- CPG message—The SBC confirms the presence of an in-band announcement if it finds an Event Information parameter with its Event indicator set to "In-band information" or "an appropriate pattern is now available" (0000011).

If an in-band announcement is present in the ACM/CPG message encapsulated in the received SIP 180 and 183 messages, the SBC may modify it as follows:

- If there is no PEM header, the SBC adds one with a value of sendonly.

- If there is a PEM header that does not have a value of sendonly, the SBC changes the value to sendonly.

Interworking Call Flows with History-Info Information

For interwork in which there are devices that support SIPI/SIP-T, the SBC supports SIP History-Info interworking. This feature enables such devices to function properly in instances that require SIP-T/SIP-I style ISUP message encapsulation in ISUP requests, and to receive any call forwarding information in the message according to ISUP standards.

For interwork in which there are device that support SIPI/SIP-T, the SBC support History-Info interworking

In the scenarios of call forwarding, when incoming INVITE, 180 or 200 OK messages contain "History-Info" header, the SBC encodes IAM, ACM (or CPG) and ANM (or CON) parameters described in the sections below.

Important configuration reminders with respect to the applicable sip-profiles include:

- If the ingress is set to ISUP and the EGRESS is set to SIP, the SBC removes ISUP content while forwarding request and add ISUP content using history-info headers while sending back response
- If the ingress is set to SIP and the EGRESS is set to ISUP, the SBC adds ISUP content using history-info headers while forwarding request and remove ISUP content while sending back response

Important operational behaviors include:

- The SBC only interworks History-Info headers in SIP or SIPS format, per RFC 4458. An example of this format, using SIPS, is below.

```
History-Info: <sips:UserA@ims.example.com;user=phone;cause=302?
Privacy=history>;index=1.1;mp=1.1
```

- If a History-Info header contains comma separated values, the SBC considers the first value as top most and the last one is the bottom most.

Interworking IAM with Redirection Parameters

When in receipt of History-Info information within a SIP to SIP-I INVITE call, the SBC interworking processes the following redirection parameter groups:

- Redirecting Number
- Redirection Information
- Original Called Number

Redirecting Number

Redirecting Number subfield interworking includes:

- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- Numbering Plan Indicator—ISDN (001)

- APRI—This interworking is dependent on the value of either the SIP privacy header or the priv-value presented in the History-info:
  - If the value is either 'History'. 'Session' or 'Header', map to **presentation restricted** (01).
  - If both values are absent or have values other than those above, map to **presentation allowed** (00).
- Address Signal—The SBC inserts the URI of the History-Info header whose hi-index matches the mp-parameter value of the last History-info header containing the "cause" URI parameter and the "mp" parameter.
  Furthermore, the SBC uses the hi-entry just before the last hi-entry with a cause parameter if:
  - The "mp" header field parameter is missing in the last hi-entry containing a "cause" URI parameter.
  - There is no HI entry with its hi-index matching the "mp" parameter of the last hi-entry and a "cause" parameter.

Redirection Information

Redirection Information subfield interworking includes:

- Redirecting Indicator—The SBC interworks this field based on context and value:
  - If the Privacy header or priv-value component in the History-Info header whose hi-index matches the "mp" parameter value of the last History-info header containing the "cause" URI parameter and the "mp" parameter, the SBC uses that value. Furthermore, the SBC uses the hi-entry immediately before the last hi-entry with a cause parameter if no HI entry with an hi-index matching the "mp" parameter of the last hi-entry with a "cause" parameter is present.
  - If the SBC identifies this value as 'History', 'Session' or 'Header', it interworks the value to **Call diverted, all redirection info presentation restricted** (100).
  - If both the Privacy Header and the priv-value components are absent or both present with value other than one mentioned above, the SBC interworks the value to **Call diverted** (011).
- Original Redirection Reason—Map to the value **Unknown/Not available** (0000)
- Redirecting Reason—The SBC maps this value based on the last HI header entry containing a cause parameter:
  - Cause Parameter 404—Map to the value **Unknown** (0000)
  - Cause Parameter 302—Maps to the value **Deflection immediate response** (0101)
  - Cause Parameter 486—Maps to the value **User Busy** (0001)
  - Cause Parameter 408—Maps to the value **No Reply** (0010)
  - Cause Parameter 480—Maps to the value **Deflection immediate response** (0101)
  - Cause Parameter 503—Maps to the value **Mobile Subscriber not reachable** (0110)
  - Cause Parameter 487—Map to the value **Deflection during alerting** (0100)
- Redirection Counter—The SBC inserts the sum of all History-Info headers in the SIP message that contain the Redirecting Reason cause parameters. This value never exceeds 5.

Original Called Number

Applicable Original Called Number subfields include:

- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- Numbering Plan Indicator—The SBC always inserts the ISDN (Telephony) numbering plan; (Recommendation E.164)
- APRI:
  - If the SIP **Privacy** value is absent or set to a value not listed below, map the **APRI** to **presentation allowed**.
  - If the SIP **Privacy** value is set to **History**, **Session**, or **Header** map to **presentation restricted**.

  If there are multiple history-info headers, the SBC selects the one with a hi-index that matches the "mp" parameter value of the first header that contains a "cause" URI parameter and an "mp" parameter. If the "mp" parameter is missing from the first history-info header that contains a "cause" URI parameter, the SBC uses the history-info header entry that precedes it.

  Furthermore, per the standard, the priv-value component in the history-info header can only have the values "history" or "none".
- Address Signal—The SBC inserts the URI of the first header with a hi-index that matches the "mp" parameter value of the first header that contains a "cause" URI parameter and an "mp" parameter.
  If the "mp" header field parameter is missing in the first hi-entry containing a "cause" URI parameter, the SBC uses the hi-entry just before it.

Interworking ACM or CPG with History-Info Headers

During a SIPI to SIP call, if the 180 RINGING response from SIP side contains a history-info header and the redirection mode on the egress side is set to "history-info", the SBC populates the following parameters from history-info header into the ACM/CPG message:

- Redirection Number
- Redirection Number Restriction
- Call Diversion Information
- Generic Notification Indicator

Redirection Number

Redirection Number subfield interworking includes:

- Nature of address indicator—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).

- Internal Network Number Indicator—The SBC always inserts the value **routing to internal network number not allowed** (1).
- Numbering Plan—The SBC always inserts the value **ISDN** (001.)
- Address Signal—The SBC inserts the URI of the last History-Info header containing a "cause" URI parameter.

Redirection Number Restriction

Redirection Number Restriction subfield interworking includes:

- APRI—This interworking is dependent on the value of either the SIP privacy header or the priv-value presented in the last History-info header with a cause parameter:
  - In the value is either 'History'. 'Session' or 'Header', map to **presentation restricted** (01).
  - If both values are absent or have values other than those above, map to **presentation allowed** (00).

Per standard, the priv-value component in the history-info header can only have values "history" or "none".

Call Diversion Information

Call Diversion Information subfield interworking includes:

- Notification subscription options—This interworking is dependent on the value of either the SIP privacy header or the priv-value presented in the History-info:
  - If there is a value in the Privacy header or priv-value component in the History-Info header, map to that value.
  - If the Privacy header received has the value "History", "Session" or "Header" or the SBC receives a priv-value of "History" in the headers component of both the diverted-to URI and the diverting URI(s), map to **presentation not allowed** (001).
  - If only the History-Info header with a priv-value of **history**, map to **presentation allowed without redirection number** (011).
  - In all other cases, map to **presentation allowed with redirection number** (010).
- Redirecting Reason—The SBC maps this value based on the cause parameter of the last HI entry that contains a cause parameter.
  - Cause Parameter 404—Maps to the value **Unknown** (0000)
  - Cause Parameter 302—Maps to the value **Unconditional** (0011)
  - Cause Parameter 486—Maps to the value **User Busy** (0001)
  - Cause Parameter 408—Maps to the value **No Reply** (0010)
  - Cause Parameter 480—Maps to the value **Deflection immediate response** (0101)
  - Cause Parameter 503—Maps to the value **Mobile Subscriber not reachable** (0110)
  - Cause Parameter 487—Maps to the value **Deflection during alerting** (0100)

Additional context information includes:

- The diverted-to URI corresponds to the hi-targeted-to-uri of the last hi-entry containing a "cause" URI parameter.
- The diverting URI corresponds to the hi-targeted-to-uri of the hi-entry containing a hi-index value that match the "mp" header field parameter value of the diverted-to URI. If the

diverted-to URI does not contain the "mp" header field parameter, the diverting URI corresponds to the hi-targeted-to-uri of the hi-entry before the last hi-entry containing a "cause" URI parameter.

- If there is no diverting URI (for example, an HI entry with hi-index matching "mp" param of last hi-entry with "cause" param is present) the hi-entry to use is the entry just before the last hi-entry with a cause parameter.
- If there is only a diverted-to URI and no other URI, the SBC only refer to the diverted-to URI while populating the "Notification subscription options" parameter.

Generic Notification Indicator

In addition, the SBC sets the Generic Notification Indicator in the ACM or CPG to **Call is Diverting** (1111011).

Interworking ANM or CON with History-Info Headers

This interworking applies to SIP-I to SIP calls. If the 200OK message from the SIP side contains one or more history-info headers and you have set the redirection mode on the egress side to **history-info**, the SBC populates these parameters into the ANM/CON message.

When configured for native IWF and in receipt of History-Info information, the SBC interworks the following parameter groups:

- Redirection Number
- Redirection Number Restriction

Redirection Number

The SBC only adds this parameter for ANM messages. Redirection Number subfield interworking includes:

- Nature of address indicator—The SBC inserts the following values, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- Internal Network Number Indicator—The SBC always inserts the value **routing to internal network number not allowed** (1).
- Numbering Plan Indicator—The SBC always inserts the value **ISDN** (001).
- Address Signal—The SBC inserts URI of the last History-info header with a cause parameter.

Redirection Number Restriction

The APRI is the only parameter interworked for this category:

- If the SIP **Privacy** or the priv-value component in the History-info entry containing a cause URI parameter is set to **History**, **Session**, or **Header** map the **APRI** to **presentation restricted** (01).
- If both the Privacy header or priv-value component in the last History-info header containing a "cause" URI parameter are absent or have any other value, map to **presentation allowed** (00).

Per the standard, the priv-value component in the history-info header can only have a value of either "history" or "none".

Interworking Call Flows with Diversion Information

For networks in which there are devices that do not support SIP-T or SIP-I (and support native SIP alone), the (SBC) supports SIP Diversion interworking. This feature enables such devices to function properly in instances that require SIP-T/SIP-I style ISUP message encapsulation in ISUP requests, and to receive any call forwarding information in the message according to ISUP standards.

The SBC interworks a SIP INVITE request to SIP-T one by inserting an ISUP IAM body based on the INVITE; this includes redirection information based on the Diversion header. This feature can also perform the reverse translation. That is, it can interwork a SIP INVITE that does have the ISUP IAM body to a non-ISUP INVITE. In this case, the SBC generates the necessary Diversion headers based on the IAM's Redirection information.

Criteria that generates this interworking includes:

- You have set the **sipi-behavior** option to **iam-anm** (Native interworking).
- There is a **sip-profile** applied to the call that has the **redirection** parameter set to **isup**.
- There is an accompanying **sip-isup-profile** applied to the call that specifies an ISUP version.
- SIP or ISUP signaling includes call forwarding information.

The SBC also performs additional processing for **IAM**, **ACM**, **CPG**, **ANM** and **CON** ISUP messages, interworking important information from SIP messages to ISUP. The SBC does not perform this interworking in the reverse direction, from SIP-I to SIP.

Interworking IAM with Diversion Information

When configured properly and in receipt of diversion information, the SBC interworks the following parameter groups in addition to those documented previously:

- Redirection Number
- Redirecting Information
- Original Called Number

For any overlapping interworking, the SBC uses the values in this diversion information list.

Redirection Number

Parameters interworked for this category include:

- Address Signal—The SBC inserts URI of the selected Diversion header. If there are multiple headers, it uses the first (top).
- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- **APRI**:
  - If the SIP **Privacy** value is absent or set to **none**, map the **APRI** to **presentation allowed**.

  - In all other cases, map to **presentation restricted**.

Redirecting Information

Parameters interworked for this category include:

- Redirecting Indicator—The SBC uses the following mapping:
  - Privacy header absent or set to **none**—The SBC inserts the value **Call diverted**.
  - Privacy header set to any other value—The SBC inserts the value **Call diverted, all redirection info presentation restricted**.
- Original Redirection Reason—The SBC always maps this value to **Unknown**.
- Redirection Counter—The SBC inserts the sum of the counters of all Diversion headers in the SIP message.

Original Called Number

For **Original Called Number**, mapping includes:

- Address Signal—The SBC inserts the URI of the selected History-Info or Diversion header. If there are multiple headers, it uses the last (bottom).
- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
  - Numbering Plan Indicator—The SBC always inserts the ISDN (Telephony) numbering plan; (Recommendation E.164)
  - **APRI**:
    * If the SIP **Privacy** value is absent or set to **none**, map the **APRI** to **presentation allowed**.
    * In all other cases, map to **presentation restricted**.

Interworking ACM or CPG with Diversion Information

When configured for native IWF and in receipt of diversion information, the SBC interworks the following parameter groups in addition to those documented previously:

- Redirection Number
- Redirection Number Restriction
- Generic Notification Indicator

For any overlapping interworking, the SBC uses the values in this diversion information list.

Redirection Number

Parameters interworked for this category include:

- Address Signal—The SBC inserts URI of the selected Diversion header. If there are multiple headers, it uses the first (top).
- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).

  - **International**, if there is a **+** in the number (encoded to 0000100).
- Internal Network Number Indicator—The SBC always inserts the value **1 routing to internal network number not allowed**.
- Numbering Plan—The SBC always inserts the value **ISDN**.

Redirection Number Restriction

Parameters interworked for this category include:

- APRI—The SBC maps the **Privacy** SIP header to the APRI.
  - If the SIP **Privacy** value is absent or set to **none**, map the **APRI** to **presentation allowed**.
  - In all other cases, map to **presentation restricted**.

Generic Notification Indicator

In addition, the SBC sets the Generic Notification Indicator in the ACM or CPG to **Call is Diverting**.

Interworking ANM or CON with Diversion Information

When configured for native IWF and in receipt of diversion information, the SBC interworks the following parameter groups in addition to those documented previously:

- Redirection Number
- Redirection Number Restriction

For any overlapping interworking, the SBC uses the values in this diversion information list.

Redirection Number

The SBC only interworks this category if it receives a Diversion header with Reason. Parameters interworked for this category include:

- Address Signal—The SBC inserts URI of the selected Diversion header. If there are multiple headers, it uses the first (top).
- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- Internal Network Number Indicator—The SBC always inserts the value **1 routing to internal network number not allowed**.
- Numbering Plan Indicator—The SBC always inserts the value **ISDN**.

Redirection Number Restriction

The APRI is the only parameter interworked for this category:

- APRI—The SBC maps the **Privacy** SIP header to the APRI.
  - If the SIP **Privacy** value is absent or set to **none**, map the **APRI** to **presentation allowed**.
  - In all other cases, map to **presentation restricted**.

Interworking User-Configured IAM Parameters into SIP INVITE Headers

You can configure the SBC to extract a specific set of ISUP parameters from the IAM message during SIP-I to SIP call processing, and interwork them into SIP headers while generating the SIP INVITE message. This list of IAM parameters can be configured in **extract-isup-params** parameter of **sip-isup-profile**.

By default, the SBC does not interwork any IAM parameters. You configure a list of parameters you want to interwork.

The parameters you can extract and interwork include:

- calling-party-number
- generic-number
- location-number
- user-to-user

You refine the list of parameters by setting the **extract-isup-params** parameter to add additional parameters, and by setting the **remove-isup-params** parameter to remove a single parameter from the list.

Keep in mind that the SBC is receiving SIP-I or SIP-T messages that contain encapsulated ISUP information. The SBC is, therefore receiving an INVITE and identifying ISUP parameters within the encapsulated information in each message. The specific operations performed by the SBC for this extraction/interworking are presented below. All operations assume you have configured extraction for the applicable parameter.

Extraction detail is compliant with T-REC-Q.1912.5.

calling-party-number

If the SBC finds a calling-party-number parameter, it interworks that information into a P-Asserted-Id SIP header, as follows:

- If no P-Asserted-Id header is present in received SIP INVITE, the SBC creates a new P-Asserted-Id with the user part extracted from the calling-party-number and domain name from the FROM URI. The SBC uses the same URI format (SIP, SIPS, TEL and so forth) as is used in the From URI in the SIP INVITE.
- If there is P-Asserted-Id header, the SBC replaces it with the calling-party-number parameter. If there are multiple P-Asserted-Ids, the SBC removes all instances except the first.
- If the "nature of address indicator" (NOA) in the calling-party-number indicates "international number" (0000100), the SBC adds a plus sign (+) as a prefix in the URI while generating/modifying the P-Asserted-Id.
- Interworks any APRI by modifying the Privacy header as follows:
  - If APRI is "Presentation restricted", adds the priv-value "id" to the Privacy header, if not already present.
  - If APRI is "Presentation Allowed" and the priv-value in the Privacy header is already "id" or "header", remove it. If the priv-value is "user', do not remove it.

generic-number

If the SBC finds the generic-number parameter with the number qualifier indicator set to "additional calling party number", (00000110), it interworks that information into the From URI

by replacing the user part of the "From" header URI with the generic-number. The SBC also prefixes the From header with a plus sign (+) if the nature of address (NOA) is set to international.

If there are multiple generic-numbers, the SBC selects the first one with its "number qualifier indicator" set to "additional calling party number".

The SBC also interworks any APRI by modifying the Privacy header as follows:

- If APRI is "Presentation restricted", adds the priv-value "user" to the Privacy header, if not already present.
- If APRI is "Presentation Allowed" and the priv-value in the Privacy header is already "user", remove it.

location-number

If the SBC finds a location-number parameter, it interworks it into a P-Access-Network-Information header by removing any exising P-Access-Network-Information and inserting a new one, as follows:

1. Set Access-type to "GSTN".
2. Set Operator-specific-GI to the text string between quotes with the sequence of digits found in octet 3 to N (except the filler) starting with the first digit.
3. Add "network-provided" if the screening indicator in location number is set to "network provided" (11).

user-to-user

If the IAM includes the user-to-user information parameter in the encapsulated IAM message, the SBC interworks this information into a User-to-User SIP header in SIP INVITE as follows:

1. Extracts the User-To-User-Information parameter (code 32) into the User-To-User SIP header, starting from the protocol discriminator.
2. Sets the encoding field to "hex" (encoding=hex)
3. Sets the content field to "isdn-uui" (content=isdn-uui)
4. Sets the purpose field to "isdn-uui" (purpose=isdn-uui)

Furthermore:

- The letters used for the hex digits are always capital letter.
- The SBC removes any existing user-to-user header from the SIP INVITE before adding a new one.

Interworking IAM Information into SIP INVITE History-info Headers

The SBC interworks the Redirecting Number, Called Party Number and Original Called Number from the ISUP IAM message into history-info headers. While interworking these parameters into a History-Info header, the SBC creates a SIP URI containing the "cause" URI parameter, an "hi-index" and an "mp" header field parameter.

Additional operational detail on this interworking includes the SBC:

- Setting the value of “Unknown User Identity” to “sip:unknown@unknown.invalid”, as defined in 3GPP TS 23.003.
- Limiting the redirection counter value from 1 to 5. The SBC does not perform interworking if this value is greater than 5, per specification.

- Using the domain name of the “From” header while generating the URI of the history-info header. If the “From” does not have a domain, such as when there is no from header or the header is in TEL URI format, the SBC uses “unknown.invalid” as the domain.
- Removing the identified Called Party Number prior to interworking if the last digit is an ‘F’, which indicates ST (Stop Sending), and if the identified number is not that actual Called Party Number.

Furthremore, the SBC follows these rules to populate cause URI parameter values:

- For placeholder history-info entries, the SBC uses the value 404.
- For history-info entries created from called party number (using the last HI entry), The SBC populates the cause parameter based on the redirecting reason, shown in the table below.

Redirecting Reason in Redirection Information	Cause
unknown/not available (0000)	404
unconditional (0011)	302
User Busy (0001)	486
No reply (0010)	408
Deflection during alerting (0100)	487
Deflection immediate response (0101)	480
Mobile subscriber not reachable (0110)	503

- For history-info entries created from the redirecting number (the penultimate History-info entry), The SBC populates the cause parameter based on the original redirection reason as mentioned in table below, referenced from ITU-T Q.1912.5.

Original Redirection Reason in Redirection Information	Cause
unknown/not available (0000)	404
unconditional (0011)	302
User Busy (0001)	486
No reply (0010)	408

Additional ISUP IAM Generation Functions

Additional IAM generation functions include interworking Clearmode calls, user-to-user headers, and PANI headers. These functions are specified by TS 29.163.

Clearmode Calls

While interworking a SIP INVITE message containing request for Clearmode call setup, the SBC adds the User-Service-Information parameter to the IAM message it generates. The applicable **User-Service-Information** subfield parameter values include:

- Coding Standard— ITU-T standardized coding (00)
- Information transfer capability—Unrestricted digital information (01000)
- Transfer mode—Circuit mode (00)
- Information transfer rate— 64 kbits/s (10000)

User to User Header Interworking

The SBC interworks user-to-user headers if the received SIP INVITE contains a user-to-user header field with the following:

- A **purpose** header field parameter set to **isdn-uui**, (or without a **purpose** parameter)
- An **encoding** header field parameter set to **hex** (or without an **encoding** parameter)
- A "content" header field parameter set to "isdn-uui" (or without a "content" parameter)

If all of the above are true, the SBC maps the content of the **uui-data** field to the **protocol discriminator** and **user information** parameters of the **user-to-user** information element as it generates the IAM message.

The applicable **user-to-user** Information subfield parameter values include:

- user-to-user information element identifier—0100000
- protocol discriminator—Convert the received User-to-User header to ASCII and add the first two nibbles.
- user information—Remove the first two nibbles (protocol discriminator) from the received user-to-user header and convert all the digits from hex to ASCII.

PANI Header Interworking

The SBC adds the location-number parameter to the IAM message if the SIP INVITE contains a P-Access-Network-Information (PANI) header that uses the format shown in the example below.

```
P-Access-Network-Info:GSTN;operator-specific-GI="5522300";network-provided
```

The SBC follows these rules while interworking the PANI to location number:

- Only interwork PANI headers that have an **operator-specific-GI** parameter.
- Only interwork PANI headers with an **operator-specific-GI** parameter that contain a sequence of digits.
- If multiple instances of PANI are present, only interwork the instance that has the access network specific component that contains a **network-provided** parameter.
- If there are multiple instances that contain a **network-provided** parameter, use the first.
- If none of the PANIs are **network-provided**, interwork the first PANI that is a GSTN type.

The SBC interworks **location number** subfields as follows:

- Nature of address indicator—**National (significant) number (national use)** (0000011)
- Internal Network Number indicator—**Routing to internal number not allowed** (1)
- Numbering Plan indicator—**ISDN** (001)
- APRI:
  - If the Privacy header is present with a value of **header**, use **Presentation Restricted** (01)
  - Otherwise, use **Presentation Allowed** (00)
- Screening Indicator:
  - If the **network-provided** parameter is present, use **Network Provided** (11)

  - Otherwise, use **user provided, verified and passed** (01)
- Address Signal—Value of **operator-specific-GI**

Interworking SIP Response to ISUP Cause Values

If a Reason header as described in IETF RFC 6432 is included in a SIP 4xx, 5xx, 6xx response, and you have enabled the **sip-config**, **sip-response-code** parameter, theSBC maps the Cause Value of the Reason header to the ISUP Cause Value field in the **REL** message, as follows:

SIP Response	ISUP Cause Value
400, 414, 416, 420, 421	111 (Protocol Error, Unspecified)
403, 417, 501	79 (Not implemented, Unspecified)
404, 485	1 (Unallocated number)
408, 504	102 (Timer expiry)
410	22 (Number Changed)
422	31 (Normal, Unspecified)
433	24 (Call Rejected due to feature at destination)
480	20 (Subscriber Absent)
483	25 (Exchange routing Error)
484	28 (Invalid Number Format)
486, 600	17 (User Busy)
488	50 (Request Facility Not Supported)
502	27 (Destination Out of order)
503	41 (Temporary Failure)
513	95 (Invalid Message, Unspecified)
603	21 (Call Rejected)
604	2 (No route specified)
606	88 (Incompatible Destination)
Other	127 Interworking Unspecified)

For errors received from the SIP-I side, the SBC inserts a Reason Header into the SIP body, if it is not generated by some other process. This also requires that you have enabled the **add-reason-header** within the **sip-config**. The SBC inserts the SIP Reasons according to the ISUP Cause Value (cause attribute and reason) mapping from ITU-T Q.850.

In-Band Announcement Indication

You can configure the SBC to add the Event Indicator parameter with value "progress" while generating a CPG by enabling the applicable **iwf-for-183** parameter.

When enabled, the SBC performs this function if the received "183 Session Progress" message either:

- Doesn't contain a PEM header, or
- Contains a PEM with the value inactive or recvonly

The SBC adds the OBCI parameter to the generated ACM if it receives a 180 or 183 message with a PEM header with value sendonly or sendrecv. The SBC always performs this interworking for 180 messages. For 183 messages, you must have also set the `iwf-for-183` parameter to `enabled` or `pem-controlled`.

Supplementary Service for COLP and COLR

During SIP-I to SIP calls, if the 200 OK message received from the IMS node contains a P-Asserted-Identity header, the SBC encodes some parameters of ISUP messages. The SBC only performs this function if the P-Asserted-Identity header is in SIP, SIPS or Tel-URI format. If Tel-URI format is accompanied by SIP and/or SIPS format, theSBC uses the header in Tel-URI format.

Connected Number

Connected Number subfield interworking includes:

- Nature of address—The SBC inserts the following values into the IAM, based on the conditions described:
  - **National**, if there is no **+** in the number (encoded to 0000011).
  - **International**, if there is a **+** in the number (encoded to 0000100).
- Numbering Plan Indicator—ISDN (001)
- APRI—This interworking is dependent on the value of either the SIP privacy header or the priv-value presented in the History-info:
  - If the privacy header is present and contains a value other than 'none', map to **restricted** (01).
  - If the privacy header is absent, or present with the value 'none', map to **allowed** (00).
- Screening Indicator—Network Provided (11)
- Address Signal—P-Asserted-Id user-part

Interworking for Call Hold and Resume Cases

IMS nodes can send UPDATE or Re-INVITE messages to the SBC during a call to put the session on hold or to resume it. In turn, the SBC encapsulates these messages, into INVITE, re-INVITE, or UPDATE messages as call progress messages (CPG) within MIME content and sends them for interworking out the SIP-I side of the call.

A caller can put a call on hold at any time after the alerting phase has begun using an UPDATE. A callee can only put a call on hold or resumed after it has answered the call. Both caller and callee can initiate hold/resume using an UPDATE or Re-INVITE. As a result, a call can only be put on hold in the SIP to SIPI direction by the caller before the call is answered. After the call is answered, either side can hold and resume the call. Furthermore, after the call is in progress, either side can effectively put the other side on hold. These actions are signaled using the a attribute in the SDP.

To put a session on hold, an IMS node sends an UPDATE or re-INVITE message with an "inactive" or a "sendonly" SDP attribute. Upon receipt of the hold request, the SBC encapsulates a CPG message containing the Generic notification indicator parameter with notification indicator set to "remote hold" while forwarding the UPDATE or re-INVITE towards the SIP-I side.

The SBC populates the following Event Information and Generic Notification Indicator subfields for hold cases in the CPG:

- Event Information:
  - Event Indicator—PROGRESS (0000010)

  - Event Presentation—No indication (0)
- Generic Notification Indicator:
  - Notification Indicator—REMOTE HOLD (1111001)

To resume a session, an IMS node sends an UPDATE or re-INVITE message with a "recvonly" or "sendrecv" SDP attribute, depending on the current state of the session. Upon receipt of the resume request, the SBC encapsulates a CPG message containing the Generic notification indicator parameter with notification indicator set to "remove retrieval" while forwarding UPDATE / re-INVITE towards SIP-I side.

The SBC populates the following subfields of Event Information for remote resume cases:

- Event Information:
  - Event Indicator—PROGRESS (0000010)
  - Event Presentation—No indication (0)
- Generic Notification Indicator:
  - Notification Indicator—REMOTE RETRIEVAL (1111010)

ISUP Version Interworking

ISUP messages can be carried in SIP messages through either a standard body or through a multipart MIME encoded body. While ANSI and ITU are the two major ISUP format types, each contains many specific variants. To facilitate instances where two sides of an ISUP-to-ISUP call use different ISUP versions, the Oracle Communications Session Border Controller (SBC) supports interworking between the following ISUP formats: ANSI, ITU, ETSI-356 (an ITU variant), GR-317 (an ANSI variant) and (Signalisation Pour l'Interconnexion des Réseaux Ouverts/Signaling for the Interconnection of Open Networks) SPIROU. To do so, the SBC can move, delete, and add parameters to various sections of the message.

The ISUP message version is determine by one of two things:

- The content type of the SIP message
- The MIME content-type

When the base and version parameters do not match, the SBC first uses the base parameter value to determine the format. If there is no base, the SBC then checks the version parameter. If there is neither, the SBC uses the **isup-version** configured in the **sip-isup-profile** configuration from the inbound realm, session agent, or SIP interface. Available values for that parameter are **ansi-2000**, **itu-99**, **gr-317**, **etsi-356**, or **spirou**. The SBC considers unknown any value for the version that fails to match one of these or is missing.

An example of the base and version information set to SPIROU is as follows:

```
application/ISUP;base=spirou;version=spirou
```

Messages that contain an unknown ISUP format pass through the SBC untouched. If there are operations to be performed on them, however, SIP-SIPI interworking takes place. After the body has been converted, the SBC updates both the base and version parameters of the content-type.

Custom formats are not supported.

Configuring SIP-SIPI Interworking

The following tasks explain how to configure SIP-SIPI Interworking on the SBC.

Configuring SIP SIPI IWF using HMR

Descriptions and procedures for using Header Manipulation Rules (HMR) to perform SIP-SIPI interworking are documented in the *Header Manipulation Rules Resource Guide*.

Configuring SIP-SIPI IWF using Session Translation

Session Translation for SIP-SIPI IWF configuration consists of three procedures performed in order:

1. Create **translation-rules** elements
2. Apply your **translation-rules** to **session-translation** elements
3. Apply your **session-translation** elements to **session-agents** or **realms**

Each of these tasks are presented below.

Configuring translation rules

Use this procedure to create one of more translation rules to apply to the SIP-SIPI IWF components of a **session-translation**.

1. In Superuser mode, navigate to the **session-router** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#session-router
ORACLE(session-router)#
```

2. Type **translation-rules** and press Enter.

```
ORACLE(session-router)# translation-rules
ORACLE(translation-rules)#
```

3. **ID**—Set a descriptive ID name for this translation rule.
4. **type**—Set the type of translation rule you want to configure. The default value is **none**. The valid values are:
   - **add**—Adds a character or string of characters to the address
   - **delete**—Deletes a character or string of characters from the address
   - **replace**—Replaces a character or string of characters within the address
   - **none**—Translation rule is disabled
5. **add-string**—Enter the string to be added during address translation to the original address. The value in this field should always be a real value; i.e., this field should not be populated with at-signs (@) or dollar-signs ($). The default value is a blank string.
6. **add-index**—Enter the position, 0 being the left most position, where you want to add the string defined in the **add-string** parameter. The default value is zero (**0**). The valid range is:

- Minimum—0
- Maximum—999999999

7. **delete-string**—Enter the string to be deleted from the original address during address translation. Unspecified characters are denoted by the at-sign symbol (@).

   The @ character only works if the type parameter is set to delete. This parameter supports wildcard characters or digits only. For example, valid entries are: delete-string=@@@@@, or delete-string=123456. An invalid entry is delete-string=123@@@. When the type is set to replace, this value is used in conjunction with the add-string value. The value specified in the delete-string field is deleted and the value specified in the add-string field is inserted. If no value is specified in the delete-string parameter and the type field is set to replace, then nothing will be inserted into the address. The default value is a blank string.

8. **delete-index**—Enter the position, 0 being the left most spot, where you want to delete the string defined in the **delete-string** parameter. This parameter is only used if the **delete-string** parameter is set to one or more at-signs. The default value is zero (**0**). The valid range is:

   - Minimum—0
   - Maximum—999999999

9. Save your work.

Next, you apply your **translation-rules** to a **session-translation**.

Applying translation-rules to session-translations

These steps assume you have created **translation-rules** elements.

1. In Superuser mode, navigate to the **session-router** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)#
```

2. Enter the **session-translation** element.
3. **ID**—Set a descriptive ID name for this session translation.
4. **rules-asserted-id**—Enter the applicable rules in the order in which they should be applied. Multiple rules should be included in quotes and separated by spaces. The example output below displays rules applied to the **rules-asserted-id** and **rules-isup-cdpn** SIPI objects to modify. Refer to the description of each session-translation object that you can manipulate to identify the signaling changes you are targeting.

```
ORACLE(session-translation)# rules-asserted-id "rule1 rule2 rule3"
```

   The following is an example of what a session translation configuration might look like:

```
session-translation
        id                              lrules-out
        rules-asserted-id               rule1 rule2 rule3
        rules-isup-cdpn                 rule4
```

5. Save your work.

Next, you apply your **session-translation** to a **realm** or a **session-agent**.

Applying session-translations to realms or session-agents

These steps assume you have created **session-translation** elements. Identify and employ the criteria by which you choose whether to apply the translations to **realms** or **session-agents**. In addition, determine whether you want to apply your translations to ingress of egress traffic.

1. In Superuser mode, navigate to the **session-router** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)#
```

2. Navigate to the chosen element, in this example a session-agent.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

3. Specify the appropriate **session-translation** object as the **in-translationid** and/or the **out-translationid** in the **session-agent** element.

```
session-agent
        in-translationid
        out-translationid               lrules-out
```

   Consider the following parameter descriptions to determine how to apply your translation rule(s):

   - **in-translationid**—Enter the configured session translation that you want to affect the incoming traffic in this parameter.
   - **out-translationid**—Enter the configured session translation that you want to affect the outgoing traffic in this parameter.

4. Save your work.

Configuring Native SIP-SIPI Interworking

You set the **sipi-behavior** option in the **sip-config**.

1. In Superuser mode, access the **sip-config** using the following navigation.

```
ORACLE#configure terminal
ORACLE(configure)#session-router
ORACLE(session-router)#sip-config
ORACLE(sip-config)#
```

2. Select the **sip-config**.

```
ORACLE(sip-config)# select
ORACLE(sip-config)#
```

3. Set the **sipi-behavior** option using the following syntax.

```
ORACLE(sip-config)# options +sipi-behavior=iam-anm
ORACLE(sip-config)#
```

4. Save your work.

Configuring the SIP-ISUP Profile

To set up a SIP ISUP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-isup-profile** and press Enter.

```
ORACLE(session-router)# sip-isup-profile
ORACLE(sip-isup-profile)#
```

4. **name**—Enter the name of the SIP ISUP profile. You will use this name when you apply this profile to realms, session agents, and SIP interfaces. This parameter is required, and it has no default value.
5. **isup-version**—Specify the ISUP version you want used in this profile in order to support SIP-T: **ansi-2000** (default), **itu-99**, **gr-317**, **etsi-356**, or **spirou**.
6. **convert-isup-format**—Enable this to convert outgoing ISUP messages to isup-version format
7. **iwf-for-183**—Disable to prevent the SBC from adding ACM or CPG for 183.
8. **country-code**—Specify the text string to use for country code interworking, specifying the string inserted into the IAM as the code party number parameter. This happens when the NPRN parameter is present in the Req-URI of an incoming INVITE, and requires that **portability-method** be set to **concatenate**.
9. **portability-method**—Specify whether the SBC performs number portability interworking on the IAM by encoding the called-party-number in the IAM based on the NPDI and NPRN parameters. Values include **none** (default) or **concatenate**.
10. Save your work.

Configuring to Prevent the Interworking of 183 Messages to ACMs

To use this option, you must have previously set **sipi-behavior** option to **iam-anm**.

1. In Superuser mode, access the **sip-isup-profile** using the following navigation.

```
ORACLE# configure terminal
ORACLE# (configure) session-router
ORACLE# (session-router) sip-isup-profile
```

2. Create a new profile, or select the desired **sip-isup-profile**.

3. Disable **iwf-for-183** within the profile using the following syntax.

```
ORACLE(sip-isup-profile)#iwf-for-183 disabled
```

4. Save your work.

Configuring Number Portability Support for IAM Interworking

This section show you how to set up Number Portability Support for IAM Interworking. First, you configure a SIP ISUP profile, and then you apply it to a realm, session agent or SIP interface.

To set up a SIP ISUP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-isup-profile** and press Enter.

```
ORACLE(session-router)# sip-isup-profile
ORACLE(sip-isup-profile)#
```

4. **name**—Enter the name of the SIP ISUP profile. You will use this name when you apply this profile to realms, session agents, and SIP interfaces. This parameter is required, and it has no default value.
5. **country-code**—Specify the text string to use for country code interworking, specifying the string inserted into the IAM as the code party number parameter. This happens when the NPRN parameter is present in the Req-URI of an incoming INVITE, and requires that **portability-method** be set to **concatenate**
6. **portability-method**—Set this parameter to **concatenate** if you want to perform interworking for Number Portability Support within the IAM. The default is **disabled**, meaning that this feature is turned off.
7. Save your work.

Configuring SIP-ISUP Redirection Interworking

To use this feature, configure:

- sip-profile—Defines the redirection behavior
- sip-isup-profile—Defines the ISUP version to use when supporting SIP-I (and SIP-T)

You can then apply these profiles to realms, session agents, and SIP interfaces where you want this feature enabled.

The **sip-profile** configuration contains information that defines redirection behavior for the configuration where you apply it. You can set the redirection behavior to: **none**, **isup**, or **diversion**. You also uniquely identify the profile so that you can apply it by name in other

configurations. This is a multiple-instance configuration, meaning that you can set up as many SIP profiles as needed.

To set up a SIP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-profile** and press Enter.

```
ORACLE(session-router)# sip-profile
ORACLE(sip-profile)#
```

4. **name**—Enter the name of the SIP profile. You will use this name when you apply this profile to realms, session agents, and SIP interfaces. This parameter is required, and it has no default value.
5. **redirection**—Choose the redirection mode you want to use: **none** (default), **isup**, **diversion**, or **history-info**. The **inherit** value is reserved for future use. Note that when you set this parameter to **isup**, you should configure along with it a SIP ISUP profile; this will avoid any possible incompatibility when support for this feature expands (as expected).
6. Save your work.

Configuring ISUP Format Interworking

To use this feature, set the version for the applicable interface using **isup-version** and enable the **convert-isup-format** parameter in the **sip-isup-profile**.

To set ISUP format interworking:

1. In Superuser mode, access the **sip-isup-profile** using the following navigation.

```
ORACLE# configure terminal
ORACLE# (configure) session-router
ORACLE# (session-router) sip-isup-profile
```

2. Create a new profile, or select the desired **sip-isup-profile**.
3. Set the **isup-version** within the profile using the following syntax to, for example, spirou.

```
ORACLE(sip-isup-profile)#isup-version spirou
```

4. Enable **convert-isup-format** within the profile using the following syntax.

```
ORACLE(sip-isup-profile)#convert-isup-format enabled
```

5. Apply your **sip-isup-profile** to the appropriate **sip-interface**.
6. Save your work.

Configuring ISUP Parameter Extraction

Configure this parmameter to define a list of ISUP parameters the SBC extracts and interworks during IAM to INVITE interworking for SIP-I to SIP and SIP-I to SIP-I calls. In addition, for SIPI outbound calls, both SIP to SIPI and SIPI to SIPI calls, the SBC interworks parameters from ACM/CPG messages into 18x messages.

To use this feature, set the parameters to extract for the applicable interface using the **extract-isup-params** parameter in the **sip-isup-profile**.

To set ISUP format interworking:

1. In Superuser mode, access the **sip-isup-profile** using the following navigation.

   ```
   ORACLE# configure terminal
   ORACLE# (configure) session-router
   ORACLE# (session-router) sip-isup-profile
   ```

2. Create a new profile, or select the desired **sip-isup-profile**.
3. Set the **extract-isup-params** within the profile referring to the following syntax.

   ```
   ORACLE(sip-isup-profile)#extract-isup-params calling-party-number
   ```

   - calling-party-number
   - generic-number
   - location-number
   - user-to-user
   - inband-announcement
4. Repeat the step above to add additional ISUP values for extraction. (This parameter accepts only one value at a time.)
5. Apply your **sip-isup-profile** to the appropriate **sip-interface**.
6. Refine you list, if necessary, using the **remove-isup-params** command followed by the parameter name you want to remove.

   ```
   ORACLE(sip-isup-profile)#remove-isup-params user-to-user
   ```

7. Save your work.

Apply Your Profiles to a Session Agent

When you want to enable this feature for a realm, session agent, or SIP interface, you configure the **sip-profile** and **sip-isup-profile** parameters with the name of the profile you want applied.

The following example shows this feature being applied to a session agent, but the realm and SIP interface configurations also have the same two parameters you use to set up the feature.

To apply a SIP profile and a SIP ISUP profile to a session agent:

1. In Superuser mode, type **configure terminal**, and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router**, and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-agent** , and press Enter.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. **sip-profile**—Enter the name of the SIP profile, which defines the redirection behavior. This is the value you entered in the **name** parameter of the SIP profile configuration. This parameter has no default value.
5. **sip-isup-profile**—Enter the name of the SIP ISUP profile, which defines the ISUP version to use for SIP Diversion SIP-ISUP interworking. This is the value you entered in the **name** parameter of the SIP ISUP profile configuration. This parameter has no default value.
6. Save your work.

Configuring SIP-ISUP Reason Header IWF

To use this feature, enable the **add-reason-header** parameter in the **sip-config**.

To enable SIP-ISUP reason header interworking:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **add-reason-header**—Enable this parameter to perform reason header interworking during native SIP-ISUP interworking. The default is disabled.
5. Save your work.

SIP Session Timer Feature

SIP does not have a keepalive mechanism for established sessions and it does not have the capability of determining whether a session is still active. User agents (UAs) may be able to

determine whether a session has timed out by using session specific mechanisms, but proxies cannot always determine when sessions are still active.

The Oracle Communications Session Border Controller provides a SIP session timer feature that, when enabled, forwards the re-INVITE or UPDATE requests from a User Agent Client (UAC) to a User Agent Server (UAS) in order to determine whether or not a session is still active. This refresh feature works for both UAs and proxies. The following paragraphs provide additional information about the session timers on the Oracle Communications Session Border Controller.

How the Session Timer Feature Works

During an active SIP call session, when a UA fails to send a BYE message at the end of the session, or when the BYE message gets lost due to network problems, the proxy cannot determine when the session has ended. Therefore, the proxy may hold onto resources associated with the call session for indefinite periods of time causing the session to never time out.

The SIP session timer feature adds the capability to periodically refresh SIP sessions by sending repeated INVITE (re-INVITE) or UPDATE Session Refresh Requests. These requests are sent during active call legs to allow UAs or proxies to determine the status of a SIP session. The Session Refresh Requests along with the session timers determine if the active sessions stay active and completed sessions are terminated.

When you enable the session timer feature on the SBC, it periodically sends out a Session Refresh Request (re-INVITE or UPDATE). The Response that is returned to the SBC contains a success status code (2xx) that contains a session timer interval. The SBC then refreshes the session timer each time it receives the 2xx Response containing that session timer interval.

The initial INVITE message sent from the UAC to the UAS contains two fields that make up the session timer interval in the SIP Session Header:

- Session-Expires (SE) - Specifies the maximum amount of time, in seconds, that can occur between session refresh requests in a dialog before the session is considered timed out.
- Minimum-SE (Min-SE) - Specifies the minimum allowed value, in seconds, for the session expiration.

The following displays the session timer interval values inserted in the SIP session INVITE message per RFC 4028:

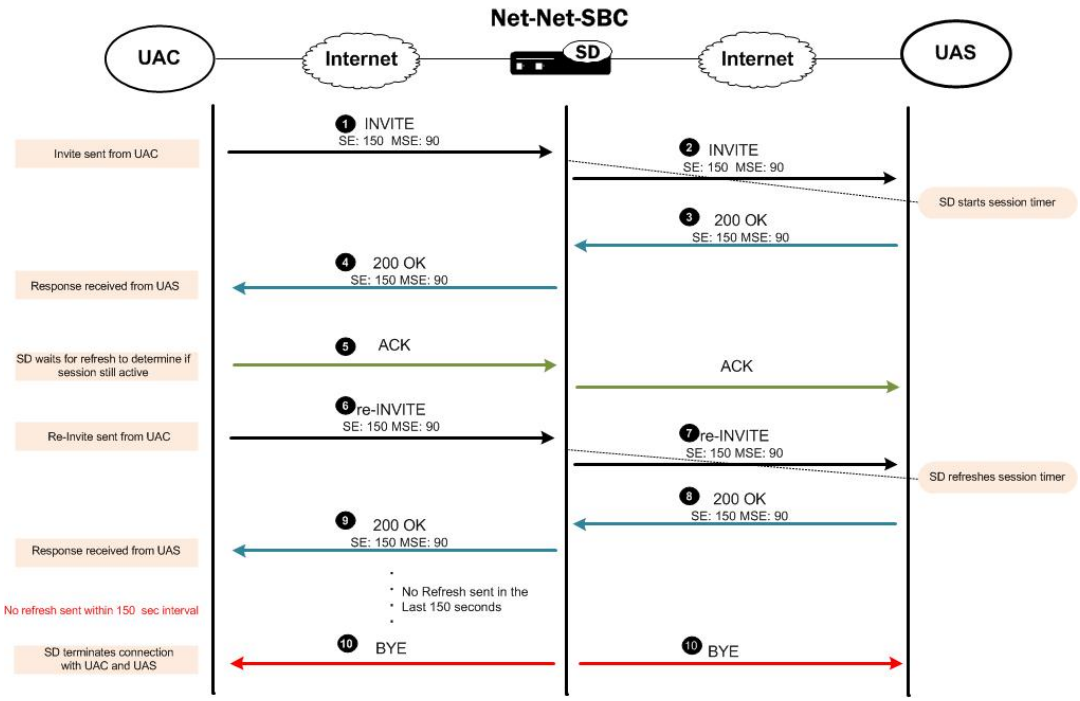
```
INVITE sip:9109621001@192.168.200.99 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.49:5060;branch=z9hG4bK0g6t23200gd0res4l580.1
Max-Forwards: 69
From: <sip:rick@192.168.1.48>;tag=SDr08od01-188c3fbc-b01a-4d68-
b741-09e5dc98a064
To: sip:149@192.168.1.49
Contact: <sip:rick@192.168.200.49:5060;transport=udp>
Call-ID: SDr08od01-9c12b48e3b0f7fad39ff3a2e0ced5ed3-v3000i1
CSeq: 3941 INVITE
Allow: INVITE, ACK, BYE, CANCEL, UPDATE, PRACK
Supported:  timer
Session-Expires: 1800
Min-SE: 90
Content-Type: application/sdp
Content-Length: 236

v=0
```

```
o=- 3462189550 3462189550 IN IP4 192.168.200.49
s=pjmedia
c=IN IP4 192.168.200.49
t=0 0
m=audio 20000 RTP/AVP 0 8 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=sendrecv
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

If the SBC receives an INVITE from the UAC with a Session-Expires header, it starts a new session timer, or refreshes an existing session timer and then forwards the INVITE to the UAS. The subsequent 2xx Responses and re-INVITES also include the session timer intervals. If the SBC does not receive a session refresh within the time specified in the session timer interval, the session timer expires, and the SBC terminates the session between the UAC and the UAS.

The following occurs when you enable the session timer feature on the SBC:

1. The UAC sends an INVITE to the SBC with the SE and min-SE values (session timer interval). The SBC starts a session timer.
2. The SBC forwards the INVITE to the User Agent Server (UAS) with the same values.
3. The UAS sends the 200 OK Response to the SBC with the session interval values.
4. The SBC forwards the Response to the UAC.
5. The UAC sends an ACK (Acknowledge) to theSBC, and the SBC forwards the ACK to the UAS.
6. The UAC sends out a re-INVITE (Session Refresh Request) to the SBC with the session interval values. The SBC refreshes the session timer.
7. The SBC forwards the re-INVITE to the UAS.
8. The UAS sends the 200 OK response to the SBC with the session interval values.
9. The SBC sends the 200 OK response to the UAC.
10. If the SBC does not receive a Response within the session interval (150 seconds in the following illustration), the timer expires, and the SBC terminates the session between the UAC and the UAS. The following illustration shows an example of a dialog between the UAC, the SBC, and the UAS during an active session.



When the SBC terminates a session it sends a BYE to both the ingress and egress call legs. If accounting is configured, the SBC also sends a RADIUS stop record with Acct-Terminate-Cause = Session-Timeout. You can enable or disable the use of the session timers using the ACLI interface at **session-router**, **sip-config**, **options**.

SIP Session Timer Configuration

You can configure the session timer feature on the Oracle Communications Session Border Controller to periodically refresh SIP sessions and determine whether or not a session is still active. If the Oracle Communications Session Border Controller determines that a session is no longer active, it terminates the session based on the session timer interval settings.

To configure the session timer feature on the Oracle Communications Session Border Controller:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter to access the SIP config-related configurations. The system prompt changes to let you know that you can begin configuring individual parameters for this object.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. Enter options followed by the following value:

- +session-timer-support

```
ORACLE(sip-config)# options +session-timer-support
```

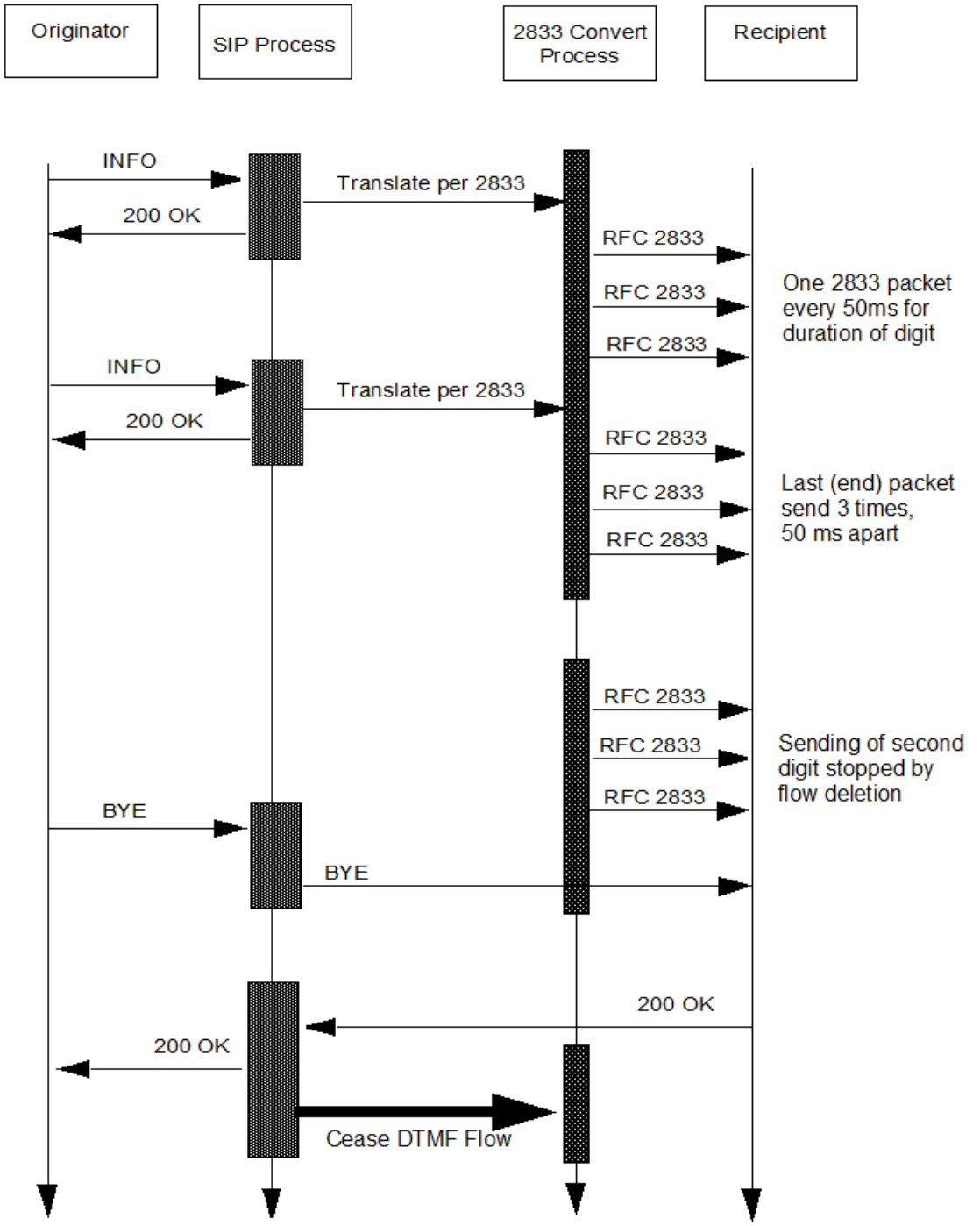
This value enables the system to start the session timer for session refreshes coming from the UAC. The system determines whether or not a session is active based on session refreshes or responses. It terminates the session when no session refreshes occur within the session timer interval. To disable the session timer feature, enter options followed by **-session-timer-support**.

```
Oracle Communications Session Border Controller(sip-config)# options -
session-timer-support
```

> **Note:**
>
> To disable session timers, you must add a minus sign (-) before the session-timers-support value.

DTMF Conversion Processing

Release S-CX6.3F1 provides a configurable SIP option to implement DTMF-to-RFC2833 tone translation. The current, default implementation, which performs well in most network topologies is shown below.



When the Oracle Communications Session Border Controller receives an INFO DTMF request, the SIP process determines whether or not it needs to perform DTMF-to-RFC2833 translation. If translation is required, the process forwards the DTMF to the 2833 convert process for translation and transmission to the recipient. Immediately after off-loading the DTMF, the SIP process sends a 200 OK response for the INFO. As shown in the figure, the 2833 convert process generates a number of RFC2833 packets to represent received DTMF digits.

Specifically, the 2833 convert process generates one RFC 2833 packet every 50 milliseconds for the duration of the DTMF digit, whose length is specified in the INFO request, and two retransmits of the last packet (known as the end packet) 50 milliseconds apart.

Consequently, the time interval between the 200 OK and the actual transmission of the RFC 2833 translation is the sum of the DTMF duration and 100 ms.

> **Note:**
>
> This time interval can be shortened to 100 ms by enabling the rfc2833-end-pkts-only-for-non-sig parameter in media-manger which results in SD only generating the last packet and its two retransmits.

The early 200 OK allows the endpoint to send the next DTMF digit before the SD has sent all the RFC2833 packets, resulting in the next digit being queued internally by the 2833 convert process before being sent.

A problem arises if the SIP process receives a BYE request from the DTMF originator while queued digits are awaiting translation and transmission. In such an event, the SIP process immediately forwards the BYE request to the recipient, ending the session with DTMF digits awaiting translation and transmission.

An alternative DTMF conversion model provides for a feedback mechanism from the 2833 convert process to the SIP process. With this model enabled, the SIP process buffers a received BYE until it obtains confirmation that all queued DTMF digits have been translated and transmitted. Only after obtaining confirmation, does it forward the BYE to terminate the session.

This processing model is enable by a SIP option, **sync-bye-and-2833**, and requires that **rfc2833-mode** parameter on the egress interfaces is NOT set to dual , any value other than dual , is supported.

1. From superuser mode, use the following command sequence to access sip-config configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. Use the SIP option **sync-bye-and-2833** to delay BYE processing until DTMF-to-RFC2833 translation has been completed.

```
ORACLE(sip-config)# options +sync-bye-and-2833="enabled"
ORACLE(sip-config)#
```

3. Use the **done** and **exit** commands to complete configuration.

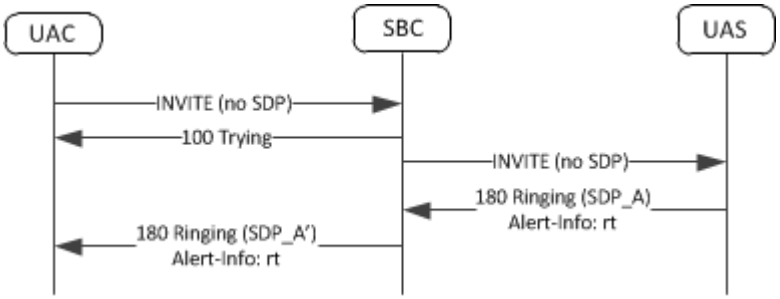
```
ORACLE(sip-config)# done
ORACLE(sip-config)# exit
ORACLE(session-router)#
```

LMSD Offerless INVITE handling

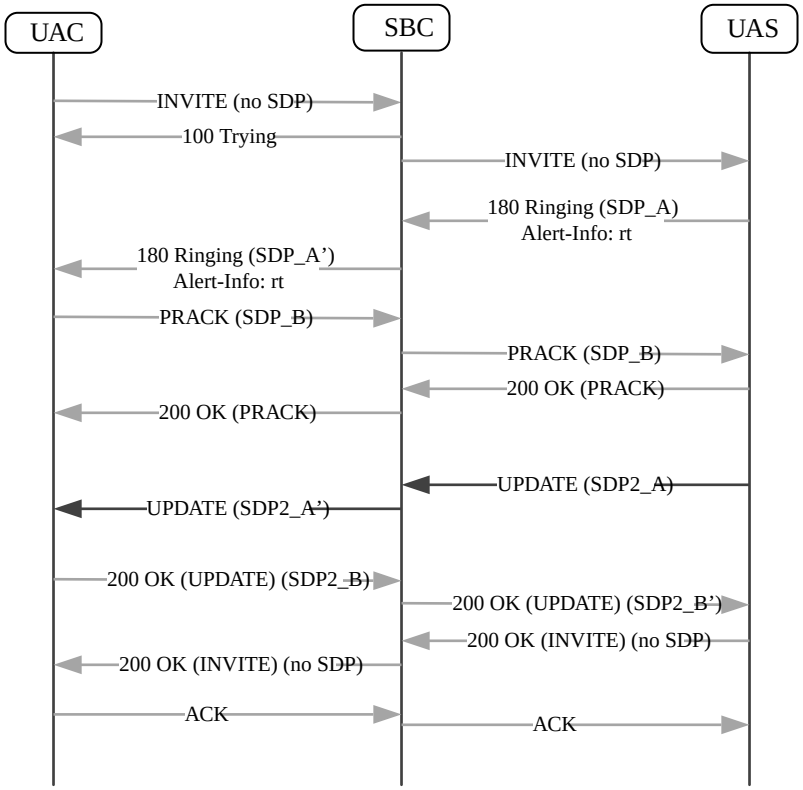
To enhance LMSD interworking, the Oracle Communications Session Border Controller does not remove SDP from a 180 response sent back to the UAC when the initial request did not

contain SDP. The Oracle Communications Session Border Controller also forwards UAS-side UPDATE requests to the UAC; it does not respond locally. These represent behavioral changes and require no configuration.

When LMSD Interworking is configured on a SIP interface, and when the UAS includes SDP in the 180 Ringing message (and the Alert-Info header is set to rt), the Oracle Communications Session Border Controller no longer strips the SDP when forwarding the 180 back to the UAC.



When LMSD interworking is configured on a SIP interface, and in the case that an early dialog has been established (a dialog where the final 2xx class response to INVITE request has not yet arrived), some call flows include an UPDATE later in the call. The Oracle Communications Session Border Controller now forwards the Update message, SDP included, directly to the UAC, whereas before it responded to the UAS's UPDATE locally.



RFC 4028 Session Timers

The Oracle Communications Session Border Controller supports RFC 4028 Session Timers. In this role, it acts as a B2BUA between two endpoints and then enforces the timer values on each call leg independently. The RFC 4028 abstract states:

This document defines an extension to the Session Initiation Protocol (SIP). This extension allows for a periodic refresh of SIP sessions through a re-INVITE or UPDATE request. The refresh allows both user agents and proxies to determine whether the SIP session is still active. The extension defines two new header fields:

- Session-Expires—which conveys the lifetime of the session
- Min-SE—which conveys the minimum allowed value for the session timer

The following parameters in the session-timer-profile configuration element are used for this feature:

session-expires—The value of the session expires header in seconds

min-se—The value of the Min-SE header in seconds (this is a minimum session expires value)

force-reinvite—Sets if the Oracle Communications Session Border Controller will send a reINVITE to refresh the session timer when applicable.

request-refresher—Set on the outbound side of a call what the Oracle Communications Session Border Controller sets the refresher parameter to. Valid values are uac, uas, or none.

response-refresher—Set on the inbound side the value of the refresher parameter in the 200OK message. Valid values are uac or uas.

In this section, the notion that a UAC or UAS supports Session Timers is indicated by the presence of the Supported: timer header and option tag.

Ingress Call Leg

Setting 200 OK's Session-Expire value

The session timer is based on the negotiation between each side's session expires value. The final value on the ingress leg is returned by the Oracle Communications Session Border Controller to the UAC, unless there is an error.

The Oracle Communications Session Border Controller can always reduce the session expires value it returns to the UAC. It checks that the Session-Expires: header is larger than the SIP Interface's min-se value. The Oracle Communications Session Border Controller then compares the received Session-Expires: header to the configured session-expires configuration element and uses the lower value for the 200 OK's Session-Expires: header. If this outbound Session-Expires: value is lower than the received Min-SE: header, it will be bumped up to the Min-SE: header's value.

If the Oracle Communications Session Border Controller's Min-SE value is larger than the Session-Expires: header, a 422 (Session Interval Too Small) message is returned to the UAC containing the Oracle Communications Session Border Controller's configured Min-SE value.

When the UAC supports (but does not require) Session Timers and the Oracle Communications Session Border Controller does not support session timers, a 200 OK is returned to the UAC with no indication of session timer support.

Refresher

The initial UAC, the side that sends the INVITE, can set itself to be the refresher (uac) or the Oracle Communications Session Border Controller as the refresher (uas). Whoever is the refresher is indicated in the 200 OK. If the UAC does not specify any refresher, the Oracle Communications Session Border Controller uses it's response-refresher value in the 200 OK. If

that value is set to uas, the Oracle Communications Session Border Controller creates and sends a re-INVITE toward the UAC with previously negotiated session expiration values.

Once the Oracle Communications Session Border Controller becomes the refresher, it does not relinquish that role. Then, when the Oracle Communications Session Border Controller sends refresh requests, it does not change any parameters (refresher role & timers) from the initial request negotiation.

UAC does not Support Session Timers

If the UAC's initial request does not include a Session-Expires: header, then the 200 OK will include the **session-timer-profile**, **session-expires** value on the ingress leg in the Session-Expires: header.

The Oracle Communications Session Border Controller also inserts the refresher parameter as configured. The orientation of UAC/UAS on the Oracle Communications Session Border Controller's view of a call leg can change if later in the call flow the endpoint designates the Oracle Communications Session Border Controller as the refresher.

> **Note:**
>
> When the request doesn't support Session Timers, the Oracle Communications Session Border Controller's reply adds session timer support according to configuration.

If the Oracle Communications Session Border Controller receives a message with a Require: timer header, and the inbound SIP interface or the final UAS do not support session timers, a 420 (Bed Extension) is returned to the UAC.

Egress Call Leg

Outbound INVITE Message

When the Oracle Communications Session Border Controller's outbound interface is configured with session timers, it forwards an INVITE to the UAS with the following headers:

Session-Expires— Oracle Communications Session Border Controller inserts the outbound SIP interface's session-timer parameter

Session-Expires refresher parameter— Oracle Communications Session Border Controller inserts the request-refresh parameter

Min-SE— Oracle Communications Session Border Controller inserts the outbound SIP interface's session-timer parameter

Supported—Supported header has the timer option tag

> **Note:**
>
> Require/Proxy-Require—If the timer parameter is present in the Require or Proxy-Require: header field in the request received from the UAC, it will be removed.

No Session Timer Configuration

If the ingress SIP interface supports session timers, and the original INVITE from the UAC included session timer support, the INVITE request sent to the UAS will have no session timer support. However, the Supported: timer header will be created. This ensures that the Oracle Communications Session Border Controller does not get a 421 response for 'timer' from the UAS.

If the ingress SIP interface supports session timers, and the UAC's initial INVITE did not include session timer support, then the INVITE sent to the UAS will have no session timer support (headers) as well.

If the ingress SIP interface does not support session timers, the INVITE is forwarded with no Session Timer alteration.

UAS Initial Response

Upon receiving a 200 OK from the UAS, if the response specifies uac as the refresher, the 200 OK includes a Session-Expires header and specifies uac as the refresher, the Oracle Communications Session Border Controller will assume the refresher role. If the 200 OK does not include a Session-Expires header, and the egress interface supports session timers, then the Oracle Communications Session Border Controller assumes the refresher role.

UAS Returns Errors

422 Session Interval Too Small—The Oracle Communications Session Border Controller in response sends the request again with new values in the 'Session-Expires' header field based on the 'Min-SE' value present in the 422 response.

421 Extension Required for 'timer'—This response can only happen if none of the other three entities (UAC, ingress SIP interface and egress SIP interface) support session timers. The 421 is forwarded through the system to the original UAC.

420 Bad Extension for 'timer'—This response should never happen because the Oracle Communications Session Border Controller will never send Require: timer header. But the event this error is received, it will be forwarded to the original UAC.

Session Refreshes

On either side of the call, the Oracle Communications Session Border Controller can be responsible for initiating the session refreshes or responding to the session refreshes.

Oracle Communications Session Border Controller as Refresher

The Oracle Communications Session Border Controller sends the refresh request when half the session expiration has elapsed. The Oracle Communications Session Border Controller always wants to remain the refresher and maintain the initially agreed upon session expiration timers.

Creating the Refresh Message

The refresh message takes the form of a re-INVITE when the force-reinvite parameter in the session timer profile is enabled. If this parameter is disabled and the remote end supports UPDATE requests, an UPDATE message will be sent.

UPDATE messages contain no SDP information.

Re-INVITE messages contain the SDP that is the same as what was sent before.

The refresh request's Session-Expires: header value is set to the existing value for the session. The refresher parameter is set to uac since the Oracle Communications Session Border Controller acts like a UAC for this refresh transaction. The Min-SE header is also included.

Processing the Refresh Response

The session expires value in the 2xx response is accepted and the timer restarts.

If the remote end does not include any session expiration parameters, the Oracle Communications Session Border Controller continues to support session timers, and assumes that the refresh interval is the same as before.

Any response that is 422 Session Interval Too Small is handled as expected. The Oracle Communications Session Border Controller resends the refresh request again with new values based on the 422 response. Any other response to the refresh request that is not a dialog/ usage destroying response is treated like a 200 OK response.

Subsequent refresh requests are created and sent after half the previous refresh interval. If non-2xx, dialog / usage destroying responses are received, the Oracle Communications Session Border Controller reduces the following refresh intervals by half, as long as the final interval is not less than 32 seconds. The Oracle Communications Session Border Controller then uses this period for sending refresh requests until it successfully receives a 2xx response.

Oracle Communications Session Border Controller as Refresh Responder

Processing the Refresh

The refresh request is processed similarly to the initial request regarding the session timer parameters. The session timer for this call leg is restarted when the Oracle Communications Session Border Controller when it sends the 200 OK response for the refresh request.

Forwarding the Refresh

When the Oracle Communications Session Border Controller receives an UPDATE request, it is forwarded to the other end since the Oracle Communications Session Border Controller cannot determine whether this request is only for session refreshing, or for other purposes as well.

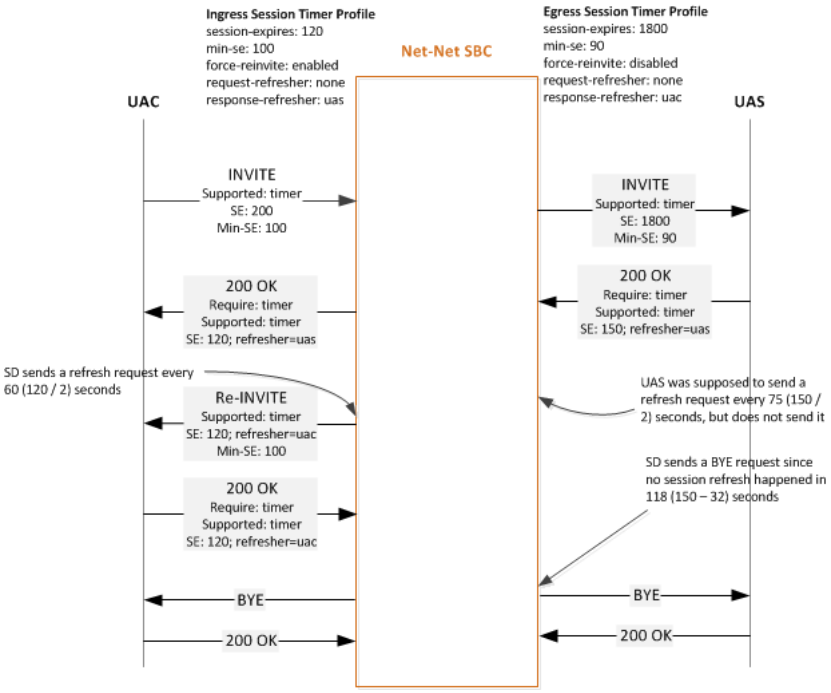
When the Oracle Communications Session Border Controller receives a re-INVITE request, it will determine whether this request needs to be suppressed, or should it be forwarded to the other end.

Timer Expiration

If the SBC fails to receive a session refresh request before the session expiration, the session will be terminated before the full session time. This is computed according to:

```
time=period-min(period/3,32)
```

After the real expiration time elapses, the SBC sends a BYE request in both directions to terminate the session.



Interaction with SIP Features

Consider the following sections that have interactions with RFC 4028 Support.

sip-config option session-timer-support

A configured session-timers-profile on a SIP interface overrides the session-timer-support option in the SIP config. The Oracle Communications Session Border Controller can still act in proxy mode for some calls and B2BUA for other calls considering which SIP interfaces session timer profiles are not configured for.

sip-feature Support

When the UAC sends a Require: timer header is in the initial request and the Oracle Communications Session Border Controller does not support session timers, and no sip-feature configuration element is configured for 'timer' for that realm, the Oracle Communications Session Border Controller replies with a 420 (Bad Extension) response for 'timer'.

When the UAC sends a Require: timer header is in the initial request and the Oracle Communications Session Border Controller does support session timers, the timer tag is removed from the Require: header even if a sip-feature configuration element is configured for 'timer'. This also applies for the Proxy-Require: header.

Oracle recommends you do not configure a sip feature configuration element while using the Session Timers feature.

sip-interface option suppress-reinvite

SIP re-INVITE suppression is automatically enabled for a SIP interface when session timers are enabled. This behavior prevents re-INVITEs whose purpose is only for session refreshes from being forwarded to the other call leg. The first re-INVITE received from a UAS on the

terminating call leg will be passed to the UAC on the originating call leg since the Oracle Communications Session Border Controller has no prior INVITE request coming from the UAS to match against.

Re-INVITEs are suppressed only when the Oracle Communications Session Border Controller receives the same INVITE request back-to-back without any intervening re-INVITE request in the opposite direction, or an UPDATE or PRACK request in either direction.

> **Note:**
>
> The SIP reINVITE Supression parameters are not replicated on the standby system in an HA environment. The first re-INVITE after a switchover will be forwarded to the far end.

Examples

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
1	INVITE → • Supported: timer SE: 200 • ← 200 OK Require: timer SE: 200; refresher=uas • INVITE → Supported: timer SE: 1200; refresher=uas ← 200 OK Require: timer SE: 500; refresher=uas	session-expires: 500 min-se: 200 request-refresher: none response-refresher: uas this element becomes refresher	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas	• INVITE→ Supported: timer SE: 500 Min-se: 400 • ← 200 OK Require: timer SE: 400; refresher=uas
2		session-expires: 500 min-se: 200 request-refresher: none response-refresher: uac this element becomes refresher	session-expires: 500 min-se: 400 request-refresher: uas response-refresher: uas this element becomes refresher	INVITE → Supported: timer SE: 500; refresher=uas Min-se: 400 _____ _____ ← 200 OK
3	INVITE → Supported: timer SE: 1200; refresher=uac Min-se: 800 _____ _____ ← 200 OK Require: timer SE: 800; refresher=uac	session-expires: 500 min-se: 200 request-refresher: none response-refresher: uas	No session timer configuration this element becomes refresher	INVITE → Supported: timer _____ _____ ← 200 OK Require: timer SE: 400; refresher=uac

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
4	INVITE → Supported: timer SE: 200; refresher=uac _____ _____ ← 200 OK Require: timer SE: 200; refresher=uac	session-expires: 500 min-se: 200 request-refresher: none response-refresher: uas	No session timer configuration	INVITE → Supported: timer _____ _____ ← 200 OK
5	INVITE → Supported: timer SE: 200 _____ _____ ← 200 OK	No session timer configuration	session-expires: 500 min-se: 400 request-refresher: uas response-refresher: uas this element becomes refresher	INVITE → Supported: timer SE: 500; refresher=uas Min-se: 400 _____ _____ ← 200 OK
6	INVITE → Supported: timer SE: 200 _____ _____ ← 200 OK Require: timer SE: 400; refresher=uas	No session timer configuration SBC behavior stays same as current behavior	No session timer configuration	INVITE → Supported: timer SE: 200 _____ _____ ← 200 OK Require: timer SE: 400; refresher=uas
7	INVITE → Require: timer SE: 200 _____ _____ ← 420 Unsupported: timer	No SIP feature for timer No session timer configuration SD behavior stays same as current behavior	No session timer configuration	
8	INVITE → Require: timer SE: 200 _____ _____ ← 420 Unsupported: timer	SIP feature configured for timer No session timer configuration SD behavior stays same as current behavior	No session timer configuration	INVITE → Required: timer SE: 200 _____ _____ ← 420 Unsupported: timer
9	INVITE → Require: timer SE: 200 _____ _____ ← 200 OK	SIP feature configured for timer No session timer configuration	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas this element becomes refresher	INVITE → Supported: timer SE: 500 Min-se: 400 _____ _____ ← 200 OK Require: timer SE: 500; refresher=uac

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
10	INVITE → Require: timer SE: 200 ——————— ——————— ← 200 OK Require: timer SE: 200; refresher=uac	No SIP feature for timer session-expires: 500 min-se: 200 request-refresher: none response-refresher: uac	session-expires: 500 min-se: 400 request-refresher: uas response-refresher: uas	INVITE → Supported: timer SE: 500; refresher=uas Min-se: 400 ——————— ——————— ← 200 OK Require: timer SE: 500; refresher=uas
11	INVITE → Require: timer SE: 200 ——————— ——————— ← 420 Unsupported: timer	No SIP feature for timer No session timer configuration	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas	
12	INVITE → SE: 200 ——————— ——————— ← 200 OK SE: 500; refresher=uas	session-expires: 500 min-se: 500 request-refresher: none response-refresher: uas this element becomes refresher	No session timer configuration	INVITE → ——————— ——————— ← 200 OK
13	INVITE → ——————— ——————— ← 200 OK	No session timer configuration	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas	INVITE → Supported: timer SE: 500 Min-se: 400 ——————— ——————— ← 200 OK Require: timer SE: 400; refresher=uas
14	INVITE → ——————— ——————— ← 421 Require: timer	No session timer configuration SD behavior stays same as current behavior	No session timer configuration	
15	INVITE → Supported: timer SE: 200 ——————— ——————— ← 422 Min-se: 400	session-expires: 500 min-se: 400		

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
16	INVITE → Supported: timer SE: 200 _____ _____ ← 200 OK Require: timer SE: 200; refresher=uac	session-expires: 800 min-se: 90 request-refresher: none response-refresher: uac	session-expires: 800 min-se: 90 request-refresher: none response-refresher: uac	INVITE → Supported: timer SE: 800 Min-se: 90 _____ _____ ← 422 Min-se: 900 _____ _____ INVITE → Supported: timer SE: 900 Min-se: 900 _____ _____ ← 200 OK Require: timer SE: 900; refresher=uas

RADIUS Interim record Generation

When refresh requests (UPDATE or Re-INVITE) are sent by the Oracle Communications Session Border Controller , no RADIUS Interim records are generated because session parameters do not change when these requests are sent.

When UPDATE requests are received by the Oracle Communications Session Border Controller , no RADIUS Interim records are generated.

When Re-INVITE requests are received by the Oracle Communications Session Border Controller , RADIUS Interim records are generated if the generate-interim parameter is enabled.

ACLI Configuration

To configure a session timer profile object:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-timer-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-timer-profile
ORACLE(session-timer-profile)#
```

4. **name**—Enter a name for this session timer profile.
5. **session-expires**—Enter the session timer value in seconds you wish this object to use natively.
6. **min-se**—Enter the minimum session timer value in seconds for this object.
7. **force-reinvite**—Leave the default of enabled for the Oracle Communications Session Border Controller to always use reINVITEs for session refreshes. Set this parameter to disabled for the Oracle Communications Session Border Controller to try using UPDATEs for session refreshes.
8. **request-refresher**—Set this to the value to insert in the refresher parameter in the Session-Expires: header on the originating call leg that the Oracle Communications Session Border Controller includes in the 200 OK response message. Valid values are **uac** and **uas**.
9. **response-refresher**—Set this to the value to insert in the refresher parameter in the Session-Expires: header on the terminating call leg that the Oracle Communications Session Border Controller includes in the INVITE message. Valid values are **uac**, **uas**, **none**.
10. Type **done** to save your work and continue.

    To apply a session timer profile to a SIP interface:

11. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

12. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

13. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

    If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

14. **session-timer-profile**—Enter the name of a session timer profile object you have configured and want applied to this SIP interface.
15. Type **done** to save your work and continue.

Verify Config Validation

The Oracle Communications Session Border Controller's verify-config function checks that the value configured in all sip-interfaces' session-timer-profiles correspond to a configured session-timer-profile name. The following is be generated when this check fails:

```
ERROR: sip-interface [id] has reference to session-timer-profile [xyz] which
does not exist
```

show sipd status

The show sipd status command now contains new statistic, called Refreshes Sent which reflects the number of refresh requests that the Oracle Communications Session Border Controller has sent. For example:

```
ORACLE#show sipd status
SIP Status                -- Period -- -------- Lifetime --------
                Active     High   Total      Total  PerMax     High
Sessions             0        1       0          2       1        1
Subscriptions        0        0       0          0       0        0
Dialogs              0        2       0          4       2        2
CallID Map           0        2       0          4       2        2
Rejections           -        -       0          0       0
ReINVITEs            -        -       0          2       1
ReINV Suppress       -        -       0          1       1
Media Sessions       0        1       0          2       1        1
Media Pending        0        0       0          0       0        0
Client Trans         0        3       2         10       3        3
Server Trans         0        0       0          6       3        3
Resp Contexts        0        0       0          6       3        3
Saved Contexts       0        0       0          0       0        0
Sockets              2        2       0          2       2        2
Req Dropped          -        -       0          0       0
Refreshes Sent       0        0       0          0       0        0
DNS Trans            0        0       0          0       0        0
DNS Sockets          0        0       0          0       0        0
DNS Results          0        0       0          0       0        0
Rejected Msgs        0        0       0          0       0        0
```

305 Response to Registrations on Secondary Interfaces

Certain devices are provisioned with point of contact (POC) lists for registration. In the context of geographic redundancy, if a UE is unable to register with its primary POC, it proceeds to its secondary POC. It is desirable to designate certain Oracle Communications Session Border Controller as secondary in order to redirect the UE to re-register with the primary SBC.

This feature is designed for Oracle Communications Session Border Controllers sitting behind Oracle Communications Subscriber-Aware Load Balancers (SLBs). This feature allows users to designateSBCs sitting behind SLBs as secondary. When registered endpoints in a realm attempt to register with the primary POC, and registration fails, the UEs proceed to their secondary point of contact (POC). Once connection to the primary POC has been restored, users can execute a command to offload the UE registrations in a given realm with a 305

response to the UEs in order for the UEs to re-register with the next POC. When the scenario consists of two POCs, the endpoint re-registers with the primary POC.

You enable this feature with an option through the SIP interface of the secondary SBC. The **notify sipd offload-users <realm>** command marks endpoints associated with a given realm to receive a 305 message. Once this command is executed, any subsequent requests from UEs in this designated realm receive a 305 message. Once a 305 message is issued, the registration cache is cleared to allow the UE request to be forwarded to the primary SBC. If the UE re-registers again with the secondary SBC, it will not receive a second 305 message. The **notify sipd offload-users <realm>** command must be executed again in order to repeat the process.

The **show registration** command is expanded to include a counter for the number of registrations received on secondary interfaces, as well as the number of endpoints marked to receive a 305.

SNMP support for this feature tracks the total count of registrations on secondary interfaces. The **apSipSecInterfaceRegThresholdExceededTrap** is generated when the total number of registrations on secondary interfaces exceeds the configured threshold, and **apSipSecInterfaceRegThresholdClearTrap** is generated when the total count falls below the configured threshold.

ACLI Instructions and Examples

To enable this feature, you must select an option for the SIP-interface of the secondary Oracle Communications Session Border Controller:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ACMEPACKET(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(session-router)# sip-interface
   ACMEPACKET(sip-interface)#
   ```

4. **options**—Type **secondary** to designate this system as the secondary point of contact.

   ```
   ACMEPACKET(session-router)# options secondary
   ```

5. Save and activate your configuration.

notify sipd offload-users

The **notify sipd offload-users <realm>** command marks the endpoints associated with the given <realm> to receive a 305 message. Once this command is executed, any re-registrations received from an endpoint in that realm receives a 305 message.

A confirmation message is sent if the command succeeds:

```
Realm <realm> is marked to offload users at HH:MM:SS
```

If the given <realm> is not a secondary realm, the following message is displayed:

```
Realm <realm> is not secondary
```

If the given <realm> does not exist, the following message is displayed:

```
Realm <realm> is not found
```

> **Note:**
>
> This command is only valid for secondary systems. If the secondary option is not enabled for theOracle Communications Session Border Controller, this command produces an error message.

show registration

The **show registration** command displays the total number of endpoints associated with secondary interfaces, and the total number of endpoints marked to receive a 305 response.

> **Note:**
>
> The total number of SIP calls to receive a 305 response are included in the Registrations marked for 305 counter.

The total number of SIP calls associated with secondary realms are included in this count.

```
AcmePacket# show registration
11:02:13-102
SIP Registrations          -- Period -- -------- Lifetime --------
                 Active    High   Total      Total  PerMax    High
User Entries          1       1       1          1       1       1
...
Secondary Interface Registrations=NNN
Registrations marked for 305=MMM
access1: N1
access2: N2
```

show registration sipd

The **show registration sipd** command includes the following parameters:

show registration sipd sec-by-ip—displays the registrations sent to secondary interfaces by IP address.

```
Registration Cache                    MON FEB 06 2012  16:33:49
                                                        Num
  IP Address      User                                  Contacts Registered at
---------------- ------------------------------ -------- ---------
. . . .
```

show registration sipd sec-by-user—displays the registrations sent to secondary interfaces by user name.

```
Registration Cache                    MON FEB 06 2012  16:33:49
                                                        Num
  Phone Number    User                                  Contacts Registered at
---------------- ------------------------------ -------- ------------
. . . .
```

show sipd endpoint-ip

If the registered endpoint is associated with a secondary SIP interface, the show sipd endpoint-ip command displays SIP Interface: secondary.

```
# show sipd endpoint-ip 1
User <sip:12341@172.16.101.67>
  Contact exp=290
    UA-Contact: <sip:12341@172.16.26.1:5060> UDP keep-acl
         realm=net172 local=172.16.101.67:5060 UA=172.16.26.1:5060
    SD-Contact: <sip:12341-2k86jbp9bj925@192.168.101.67:5060> realm=net192
    Call-ID: 1-839@172.16.26.1'
    SA=192.168.26.2
    Service-Route='<sip:h19216806003.dg.com;lr>'
  SIP Interface: secondary
```

SNMP Configuration

To configure the threshold for the number of registrations on secondary SIP interfaces in order to generate a trap:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ACMEPACKET(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(session-router)# sip-config
   ACMEPACKET(sip-config)#
   ```

4. **options**—Type **sec-reg-threshold =X,** where X equals the number of registrations received on all secondary interfaces. Once that number has been reached, the apSipSecInterfaceRegThresholdExceededTrap is generated.

   ```
   ACMEPACKET(session-router)# options secondary
   ```

5. **options**—Type **sec-reg-threshold-clear =Y.** When the number of registrations received on all secondary interfaces drop below Y, the apSipSecInterfaceRegThresholdExceededClearTrap is generated.

> **Note:**
>
> The value of sec-reg-threshold-clear =Y must be less than sec-reg-threshold =. The values cannot be equal.

```
ACMEPACKET(session-router)# options secondary
```

6. **options**—Type **sec-reg-threshold-alarm-severity =** [minor | major | critical] to set the alarm severity for APP_ALARM_SEC_INTF_REG_EXCEED.
7. Save and activate your configuration.

SNMP

The apSipSecInterfaceRegThresholdExceededTrap trap is generated when the total count of all registrations on the secondary interfaces cross **sec-reg-threshold-exceed** is exceeded.

Object Identifier Name: apSipInterfaceRegNotificationsGroup

Trap Name	Object Identifier Name: Number	Description
apSipSecInterfaceRegThresholdExceededTrap	.1.3.6.1.4.1.9148.3.15.2.1.2.0.1	The trap will be generated if the total number of registrations on all secondary SIP interfaces exceed threshold.
apSipSecInterfaceRegThresholdClearTrap	.1.3.6.1.4.1.9148.3.15.2.1.2.0.2	The trap will be generated if the total number of registrations on all secondary SIP interfaces go below clear threshold.

A hardware alarm is also generated for this event. The severity of the alarm is configured in the **sec-reg-threshold-alarm-severity** option.

Alarm Name	Alarm ID	Alarm Severity	Cause(s)	Example Log Message	Trap Generated (Trap Reference)
APP_ALARM_SEC_INTF_REG_EXCEED	0X50018	Minor by default. The severity is configurable.	The total number of registrations on all secondary SIP interfaces exceeds the configured threshold.	Number of Secondary Interface registrations <total> has exceeded the secondary interface registration threshold of <max>.	apSyslogMessageGenerated (ap-slog.mib) apEnvMonStatusChangeNotification (ap-env-monitor.mib) apSysMgmtFanTrap (ap-smgmt.mib)

The following objects, with the object identifier name apSipInterfaceRegObjectsGroup, monitor the number of registrations on secondary interfaces:

SNMP GET Query Name	Object Identifier Name: Number	Description
apSipSecInterfaceTotalRegistrations	.1.3.6.1.4.1.9148.3.15.1.1.1.1.0	The total number of registrations on all secondary SIP interfaces.

SNMP GET Query Name	Object Identifier Name: Number	Description
apSipSecInterfaceRegThreshold	.1.3.6.1.4.1.9148.3.15.1.1.1.2.0	The max threshold for registrations on all secondary SIP interfaces beyond which trap and alarm will be raised
apSipSecInterfaceClearThreshold	.1.3.6.1.4.1.9148.3.15.1.1.1.3.0	The threshold for registrations on all secondary SIP interfaces below which if alarm was raised before, it will be cleared.

6

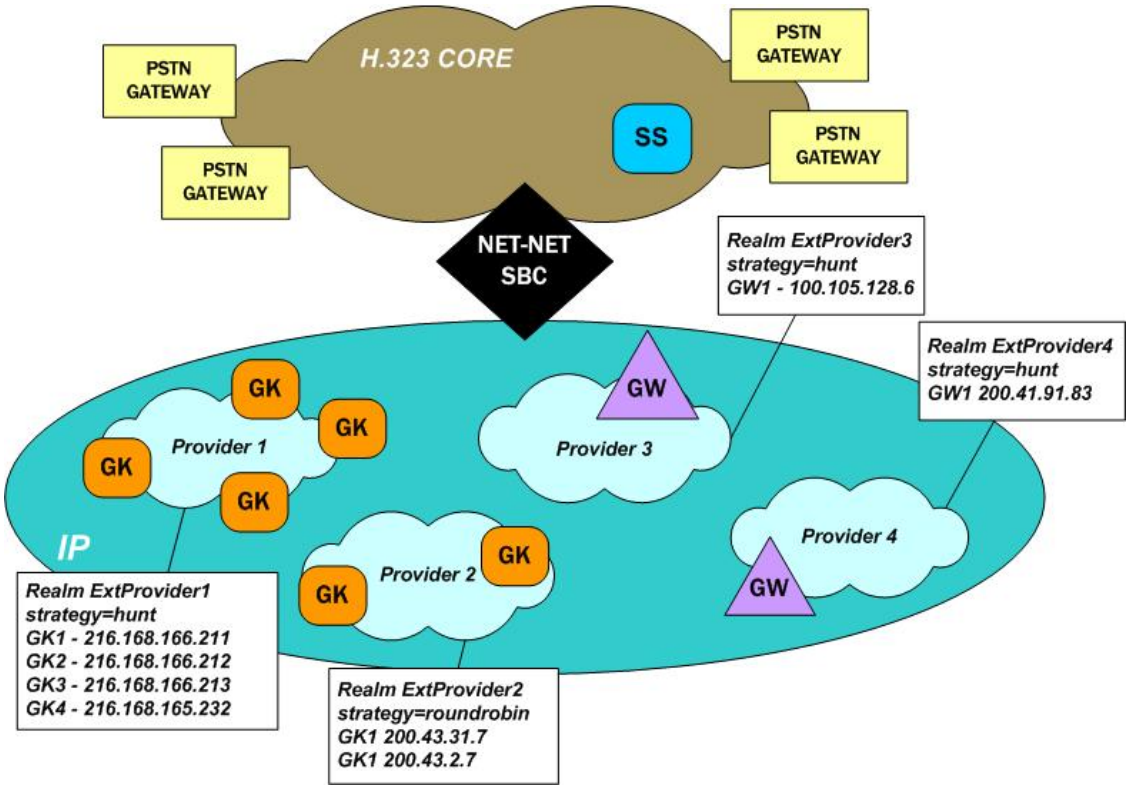
H.323 Signaling Services

The Oracle Communications Session Border Controller supports H.323 signaling in a way that permits interworking between different H.323 configurations from different providers and carriers. H.323 signaling capabilities on the Oracle Communications Session Border Controller include:

- H.323 V4—Improves on previous versions of the protocol in functionality, scalability, and reliability
- H.225 call signaling with RAS—Establishes connections between H.323 endpoints so real-time data can be exchanged
- H.245—Establishes the type of media flow and manages that flow after it has started
- H.245 tunneling—Encapsulates H.245 messages within H.225/Q.931 messages; when enabled and used with a firewall, one less TCP port is needed for incoming connections
- Fast Start (and Fast Start with parallel H.245)
- H.323 Annex E support for UDP signaling—Provides for multiplexed call signaling over UDP to increase potential call volume and enhance performance

Peering Environment for H.323

The following diagram shows a peering environment for H.323, with the Oracle Communications Session Border Controller positioned between the H.323 core and external providers.



The configuration information shown in the diagram can help you to understand how some basic Oracle Communications Session Border Controller concepts work. The providers in this depiction are configured as realms, and the strategies you see are for session agent group. What you do not see in this diagram is the fact that the Oracle Communications Session Border Controller is configured with sets of H.323 interfaces within it. These interfaces are internal (for an internal provider) and external (for the external providers you see).

Overview

Using H.323 on your Oracle Communications Session Border Controller, you can implement different signaling modes and use features to enhance H.323 capabilities. In the information that follows, you will find detailed explanations of the H.323 signaling mode and of the features available. This chapter gives operational details and later outlines the steps you need to take when features require configuration.

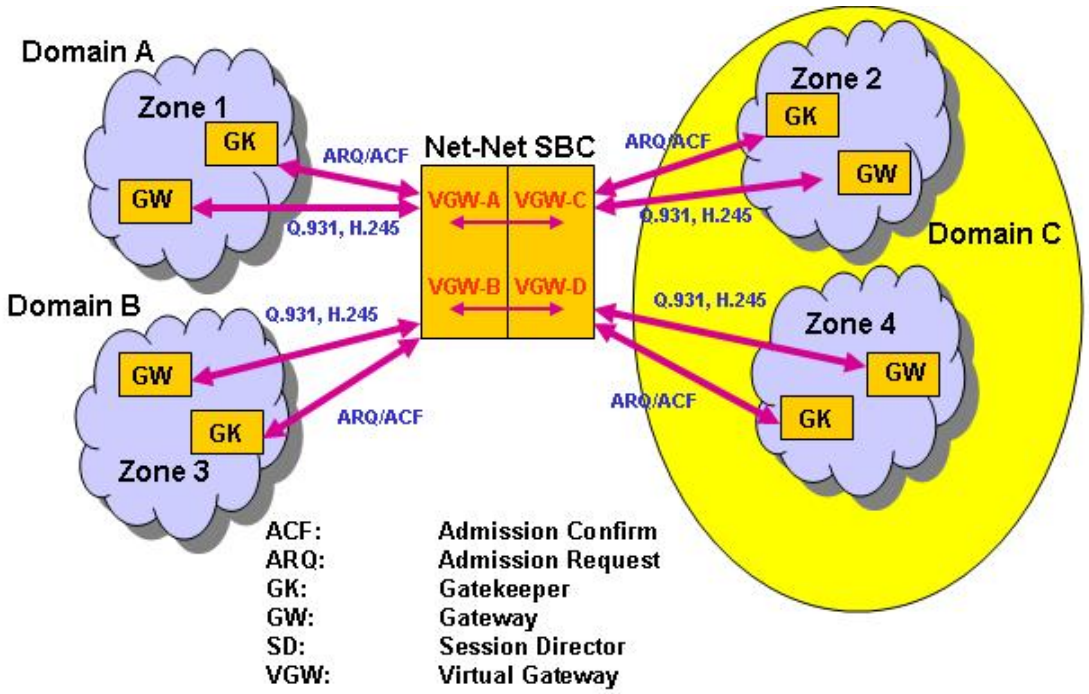
Signaling Modes of Operation

Your Oracle Communications Session Border Controller can operate in different H.323 signaling modes:

- Back-to-back gateway signaling
- Back-to-back gatekeeper proxy and gateway
- Interworking gatekeeper/gateway

Back-to-Back Gateway Signaling

This section explains how signaling takes place when the SBC functions as a B2BGW for H.323. The following diagram illustrates the SBC acting as a B2BGW.



When configured as a B2BGW, the SBC appears as multiple H.323 gateways to multiple networks. You can think of the SBC as having virtual gateways, that discovers and registers with a gatekeeper in its respective domain. In this configuration, you need to set the service mode (**isgateway**) parameter for the H.323 interface to enabled for two H.323 interfaces. These interfaces are related either through their outgoing interface (**assoc-stack**) parameters or through routing policies.

If you configure your SBC to operate in this mode, it does not issue or respond to LRQs by either confirming them or rejecting them.

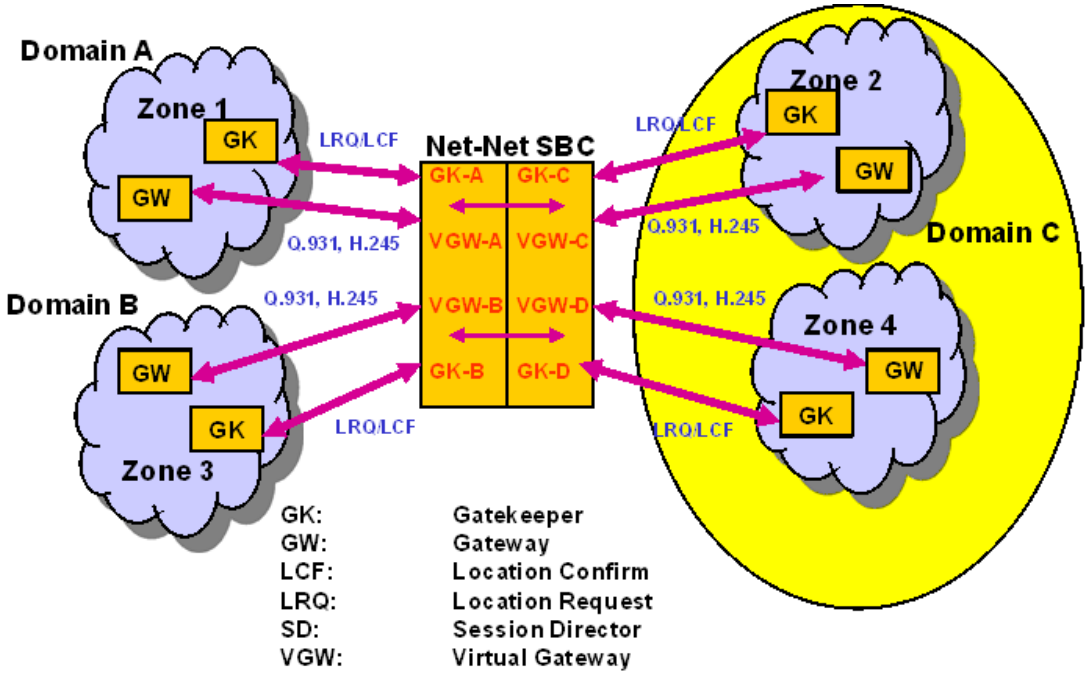
In the diagram above, the SBC sends ARQs to the corresponding gatekeeper in its zone when a call is received on the associated interface. In this behavior, the SBC acts as a gateway, complying with the H.323 standard, and registers with the configured gatekeeper in its assigned zone. You set all parameters related to the gateway registrations, such as gateway prefix numbers, in the H.323 interface configuration.

In this mode, you can also configure the SBC to run like a gateway without a gatekeeper by turning off automatic discovery (auto-gk-discovery) for the remote gatekeeper. When the SBC receives a Setup message, it does not send an ARQ and there is no registration for admission requests. Without automatic gateway discovery, the SBC uses the local policy to find the appropriate destination for the call. This destination is normally the IPv4 address of the endpoint or gateway, using the well-known port 1720.

If you enable this capability, then the SBC finds a gatekeeper.

Back-to-Back Gatekeeper Proxy and Gateway

This section explains how signaling takes place when the SBC functions as a back-to-back gatekeeper proxy and gateway for H.323. The following diagram illustrates theSBC acting as a B2B gatekeeper proxy and gateway.



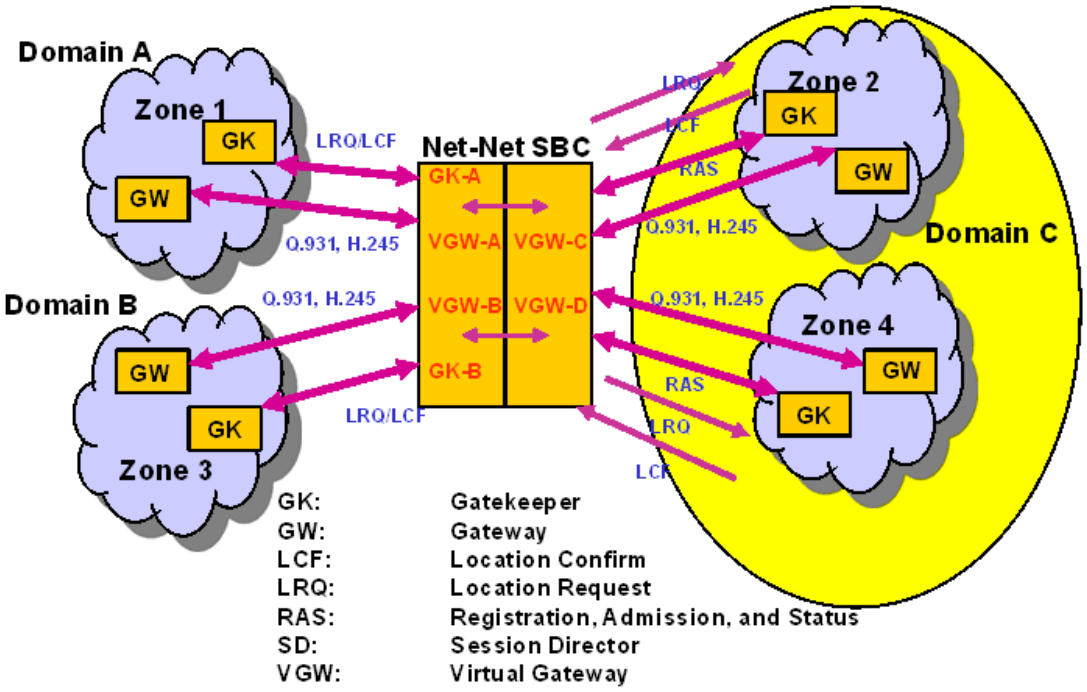
In this application, with the service mode (**isgateway**) parameter set to disabled, the SBC responds to LRQs and issues LCFs and LRJs. It sends LRQs and LCFs/LRJs to the local IPv4 address for the H.323 interface. The SBC responds to the LRQs by providing a signaling address that performs gateway functions.

When you use it as a back-to-back gatekeeper proxy and gateway, the SBC does not issue ARQs. In addition, all parameters related to registration, such as gateway prefix numbers, are ignored.

When you do not configure a gatekeeper, the SBC uses the local policy to find the appropriate destination for the call. If there is a matching local policy, the SBC returns an LCF to the originating gateway. If no local policy matches, the SBC rejects the call by sending an LRJ.

Interworking Gatekeeper-Gateway

This section explains how signaling takes place when the SBC functions as an interworking gatekeeper-gateway for H.323. The following diagram shows the SBC acting as an interworking gatekeeper-gateway.



When you configure your SBC for interworking gatekeeper-gateway mode, one H.323 interface behaves as a B2BGW and its associated interface for the corresponding network behaves like a gatekeeper proxy and gateway. The interface for the gatekeeper proxy and gateway issues and responds to LRQ messages on its network. If the SBC knows the gatekeeper in the network of the gateway interface (Zone 2), it sends an LRQ to that gatekeeper. If the gatekeeper responds with an LCF or LRJ, the SBC forwards it.

If the gatekeeper (in Zone 2) is unknown, then the SBC responds to LRQs on the gatekeeper-gateway network (Zone 1) by using the local policy to determine the appropriate destination for the LRQ. If there is no local policy that matches, then the SBC sends an LRJ.

For this configuration, the gateway interface has its service mode (**isgateway**) set to enabled, and the gatekeeper interface has its service mode (**isgateway**) set to disabled.

Realm Bridging with Static and Dynamic Routing

The Oracle Communications Session Border Controller uses static routing and policy-based, dynamic routing to handle H.323 traffic. These types of routing have to do with the way that the outgoing stack is selected.

- Static routing—The incoming H.323 stack always uses the associated H.323 stack that you configure for outgoing traffic; no other stacks are considered.
- Dynamic routing—When there is not an associated stack configured, the Oracle Communications Session Border Controller performs policy-based, dynamic routing known as realm bridging. In this type of realm bridging, the Oracle Communications Session Border Controller checks the configured local policies for address information corresponding to the incoming traffic and finds an address that matches. Next, it checks the next hop in the local policy to determine a realm and uses the first H.323 interface that matches it.

Before You Configure

In order to run H.323 on your Oracle Communications Session Border Controller, you need to configure the basic parameters: physical and network interfaces; global system parameters; SNMP, trap receiver, and accounting support, and any holiday information you might want to set.

You should also decide how you want to set up realms and routing (including the use of session agents and session agent groups) to support H.323 operations.

Global H.323 Settings

When you configure H.323 signaling for your Oracle Communications Session Border Controller, you set global and per-interface parameters. The global parameters govern how the Oracle Communications Session Border Controller carries out general H.323 operations, and these settings are applied to all interfaces you configure for H.323 use. For example, you can turn H.323 support on and off for the entire Oracle Communications Session Border Controller using these settings.

Global H.232 Settings Configuration

For the ACLI, global H.323 parameters are:

```
state          State of the H.323 protocol
log-level      Log level for H.323 stacks
response-tmo   maximum waiting time in sec for response to a SETUP message
connect-tmo    maximum waiting time in sec for establishment of a call
options        optional features/parameters
```

Accessing Global H.323 Parameters

To access the global H.323 configuration parameters in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# h323
   ```

   From this point, you can configure global H.323 parameters. To view all H.323 configuration parameters, enter a **?** at the system prompt. Access to the H.323 interface (**h323-stack**) configuration also appears.

Global H.323 Settings

To configure global H.323 parameters:

1. **state**—Enable or disable the state of H.323 signaling. The default value is **enabled**. Valid values are:
   - enabled | disabled
2. **response-tmo**—Enter the amount of time in seconds that the Oracle Communications Session Border Controller waits between sending a Setup message and tearing it down if there is no response. The default value is **4** and we recommend you leave this parameter set to this value. The valid range is:
   - Minimum—0
   - Maximum—999999999

   A response might be any of the following messages: Call Proceeding, Connect, or Alerting.
3. **connect-tmo**—Enter the amount of time in seconds that the Oracle Communications Session Border Controller waits between sending a Setup message and tearing it down if it does not specifically receive a Connect message from the endpoint. The default is **32** and we recommend that you leave this parameter set to this value. The valid range is:
   - Minimum—0
   - Maximum—999999999

   Receiving a Proceeding or Alert message from the endpoint does not keep this timer from expiring.
4. **options**—Set any options for H.323 features that you want to use. This parameter has a global impact on H.323 behavior, rather that being applied on a per-interface basis.

   If you do not configure options for global H.323 behavior, none appears in the configuration display.
5. **log-level**—Set the process log level for monitoring all H.323 activity on the Oracle Communications Session Border Controller. The default is **INFO** and leaving this parameter set to this value provides an intermediate amount of detail in the logs. Other valid values are:

> **Note:**
>
> Any log level you set here overrides the log level you set in the system configuration's process log level parameter.

Numerical Code	Acme Packet Log Enumeration	Description
1	EMERGENCY	Logs conditions of the utmost severity that require immediate attention.
2	CRITICAL	Logs events of serious condition that require attention as soon as possible.
3	MAJOR	Logs conditions indicating that functionality is seriously compromised.

Numerical Code	Acme Packet Log Enumeration	Description
4	MINOR	Logs conditions indicating that functionality has been impaired in a minor way.
5	WARNING	Logs conditions indicating irregularities in performance.
6	NOTICE	For Acme Packet customer support.
7	INFO	
8	TRACE	
9	DEBUG	

H.323 Interfaces

You need to configure H.323 interfaces for inbound and outbound traffic. When you configure H.323 interfaces, you can set:

- Identity and state
- Realm and H.323 interface associations
- H.323 interface settings for the interface's IPv4 address, RAS and Q. 931 ports, maximum number of Q.931 ports to allow, and any Annex E support you need
- H.323 system resource allocation

H.232 Interfaces Configuration

These are the ACLI parameters that you set:

```
name                 Name of the stack
state                State of the stack
isgateway            Enable the stack to run as a gateway
terminal-alias       List of aliases for terminal
ras-port             Listening port for RAS request
gk-identifier        Gatekeeper's identifier
q931-port            Q.931 call signalling port
alternate-transport  Alternate transport addresses/ports
q931-max-calls       Maximum number of Q.931 calls
max-calls            Stack's maximum number of calls
max-channels         Maximum number of channels per channel
```

To access the H.323 interface (h323-stack) and service mode parameters:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

```
ORACLE(session-router)# h323
```

4. Type **h323-stacks** and press Enter.

```
ORACLE(h323)# h323-stacks
ORACLE(h323-stacks)#
```

From this point, you can configure H.323 interface and service mode parameters. To view all H.323 interface parameters, enter a **?** at the system prompt. The display also includes H.323 service mode parameters.

Identity and State

To set the identity and state of the H.323 interface:

1. **name**—Enter a name for the H.323 interface using any combination of characters entered without spaces. For example: InternalGK1.
2. **state**—Enter the state of this H.323 interface. The default value is **enabled**. Valid values are:
   - enabled | disabled

Realm and Interface Associations

To link this H.323 interface to a realm and to an outgoing H.323 interface:

1. **realm-id**—Enter the identifier for the realm served by this H.323 interface. This parameter must be configured with a valid identifier value from a realm configuration.
2. **assoc-stack**—Enter the name of the outgoing H.323 interface that you want to associate with the H.323 interface you are configuring. To use realm bridging with static routing, you need to set the outgoing H.323 interface. If you do not enter a name, the Oracle Communications Session Border Controller uses dynamic, policy-based selection using the local policy.

H.323 Signaling Interface Settings

You can set the following parameters to define basic settings for your H.323 interface. This is where you set the IPv4 address for opening sockets, the RAS and Q.931 ports, and the maximum number of Q.931 calls that you want to allow.

This is also where you establish Annex E alternate transport. Annex E supports multiplexed call signaling over UDP so that call volume and performance are potentially enhanced. If you do not configure Annex E support, then this H.323 interface does not listen for Annex E requests.

To configure H.323 interface settings:

1. **local-ip**—Enter the IPv4 address that the H.323 interface uses when opening sockets; this is the default H.323 interface IPv4 address. You must use a valid IPv4 address. For example: 192.168.2.5. The default value is **0.0.0.0**.
2. **ras-port**—Enter the number of the port on the local IPv4 address (**local-ip**) on which the Oracle Communications Session Border Controller listens for RAS requests. We

recommend that you set this parameter to its default, the well-known port **1719**. The valid range is:

- Minimum—0
- Maximum—65535

  If you set this parameter to **0**, the Oracle Communications Session Border Controller uses a port assigned by the operating system.

3. **q931-port**—Enter the number for the port on the local IP address for the Q.931 call signaling port. We recommend that you leave this parameter set to its default, **1720**. The valid range is:
   - Minimum—0
   - Maximum—65535
4. **q931-max-calls**—Enter the maximum number of concurrent Q.931 calls you want to allow. The default value is **200**; however, this value should be less than the maximum number of calls you set when configuring H.323 features. The valid range is:
   - Minimum—0
   - Maximum—65535

     If the number of received Q.931 calls exceeds this number, the H.323 interface returns a busy state.
5. **alternate-transport**—Enter a list of one or more Annex E IPv4 address and port combinations for alternate transport. If you do not configure this list, then the Oracle Communications Session Border Controller does not listen for incoming Annex E requests. You must enter the IPv4 address and port combination in the following format, where the two are separated by a colon: **IPv4Address:Port**.

H. 323 System Resource Allocation

You can set the following parameters to determine how many concurrent calls and concurrent channels you want to allow for each H.323 interface.

To allocate H.323 system resources:

1. **max-calls**—Enter the maximum number of concurrent calls allowed on this H.323 interface. The default value is **200**. The valid range is:
   - Minimum—0
   - Maximum—4294967295
2. **max-channels**—Enter the maximum number of concurrent channels allowed for each call associated with this H.323 interface. The default value is **6**. The valid range is:
   - Minimum—0
   - Maximum—4294967295

     The Oracle Communications Session Border Controller checks this parameter on initialization to reserve the appropriate network resources.

H.323 Service Modes

When you set the H.323 service mode, you configure parameters that define what type of service an H.323 interface provides. These parameters govern how the interface functions when you want it to behave as a gatekeeper or as a gateway.

This is also where you set options that support particular H.323 features for a specific interface. These options are different from the ones you set in the global H.323 configuration because they apply only to the interface where you specify them.

H.232 Service Modes Configuration

These are the ACLI parameters that you set:

```
isgateway             Enable the stack to run as a gateway
registration-ttl      Number of seconds before the registration becomes
invalid
terminal-alias        List of aliases for terminal
auto-gk-discovery     Enable automatic gatekeeper discovery
multicast             RAS multicast address
gatekeeper            Gatekeeper's address and port
gk-identifier         Gatekeeper's identifier
h245-tunneling        Enable H.245 Tunneling support
prefixes              List of supported prefixes
process-registration  Enable Registration Request processing
allow-anonymous       allowed requests from H.323 realm
```

To configure the service mode for the H.323 interface:

1. **allow-anonymous**—Enter the admission control of anonymous connections from an H.323 realm accepted and processed by this H.323 stack. The default value is **all**. The valid values are:
   - **all**—Allow all anonymous connections
   - **agents-only**—Allow requests from session agents only
   - **realm-prefix**—Allow session agents and addresses matching the realm prefix
2. **is-gateway**—To use this interface as an H.323 gateway, leave this parameter set to **enabled**, its default value. If you want to use this interface as an H.323 gatekeeper, set this parameter to **disabled**. Valid values are:
   - enabled | disabled
3. **terminal-alias**—Enter a list of one or more aliases that identify the H.323 interface. This value is either the gateway alias or the gatekeeper identifier, depending on the mode you configure for the interface. The aliases are set in the sourceInfo information element of outgoing ARQs.

Configuring Gateway Only Settings

If you are using the H.323 interface as a gateway, you might want to set registration time-out and address prefix parameters.

To configure gateway only settings:

1. **registration-ttl**—Enter the number of seconds before a registration becomes invalid. This value is used during the initial registration process. However, when a registration is confirmed, the time-to-live (TTL) value set by the gatekeeper in the Registration Confirm (RCF) message overrides this value. The default value is **120**. The valid range is:
   - Minimum—0
   - Maximum—4294967295

2. **prefixes**—Enter a list of prefixes for this H.323 interface. Possible prefix types include:

   - H.323 ID | E.164 | URL | IPv4 address

     These prefixes are sent from a gateway interface to a gatekeeper and indicate valid prefixes accepted by that interface for incoming calls. They are used if the interface is configured as a gateway (the **is-gateway** parameter is set to enabled).

     Your entries for this parameter must appear as they do in the following example:

     ```
     e164=17817566800 url=http://www.companyname.com
     h323-ID=xyz email=user@companyname.com
     ipAddress=63.67.143.4:2000
     ```

Gatekeeper Proxy Settings

If you are using the H.323 stack as a gatekeeper proxy, you might want to set:

- Whether registration processing is enabled or disabled
- Whether or not this H.323 interface is signaling-only
- At what H.225 call stage the H.245 procedures should be initiated

To configure gatekeeper proxy settings:

1. **process-registration**—To have the Oracle Communications Session Border Controller drop all RRQs, meaning that it does not acknowledge any requests, leave this parameter set to **disabled**, its default. To have the Oracle Communications Session Border Controller process any RRQs that arrive on this H.323 interface, set this parameter to **enabled**. Valid values are:

   - enabled | disabled

     When registration processing is enabled and the Oracle Communications Session Border Controllerreceives an RRQ on this H.323 interface, it will route the request to the appropriate gatekeeper. After the gatekeeper confirms that registration with an RCF, the Oracle Communications Session Border Controller also confirms it with the endpoint that sent the RRQ. Then the registration becomes part of the Oracle Communications Session Border Controller's registration cache. If this endpoint does not confirm the registration, then the Oracle Communications Session Border Controller will reject the registration with an RRJ and will not cache it.

2. **proxy-mode**—Set this field to the proxy mode that you want to use for the signaling only operation mode. Valid values are:

   - H.225 | H.245

     You can leave this field blank (default) if you are not using a proxy mode.

3. **h245-stage**—Set this field to the stage at which the Oracle Communications Session Border Controller transfers the H.245 address to the remote side of the call, or acts on the H.245 address sent by the remote side. The default value is **connect**. Valid values are:

   - Setup | Alerting | Connect | Proceeding | Early | Facility | noh245 | Dynamic

H.323 Features

This section provides general descriptions of the H.323 features available on the Oracle Communications Session Border Controller and instructs you in how to configure them. Not all of the features described in that chapter require configuration.

Fast Start Slow Start Translations

The Oracle Communications Session Border Controller can translate between Fast Start H.323 endpoints and Slow Start H.323 endpoints. Using this feature, you can reduce delay in establishing media, improve performance, and reduce network congestion caused by a high number of messages being exchanged. Fast Start and Slow Start calls handle information about media for a session in different ways. In a Fast Start call, information about the media is contained in the Setup message. In a Slow Start call, that information is exchanged between endpoints after the session has been established.

When you Fast Start/Slow Start translation, the Oracle Communications Session Border Controller can take a Slow Start call from an H.323 endpoint that does not support Fast Start and re-initiate that call as Fast Start. It also allows an H.323 endpoint that does not support Fast Start to receive a Slow Start call from a Fast Start source because the Oracle Communications Session Border Controller performs all necessary translations.

For the ACLI, the following parameters apply:

```
fs-in-first-msg       Fast Start must be sent in 1st response to Setup message
call-start-fast       Enable outgoing Fast Start call
call-start-slow       Enable outgoing Slow Start call
media-profiles        list of default media profiles used for outgoing call
```

Fast Start to Slow Start Translation

The Oracle Communications Session Border Controller supports translations from H.323 Fast Start to Slow Start. Using this feature, an H.323 endpoint that only supports Slow Start can call from a Fast Start source when that call goes through the Oracle Communications Session Border Controller.

In a Fast Start call, the originating H.323 endpoint sends a fastStart element in its Setup message. This element contains H.245 OLC messages that allow Fast Start endpoints to establish a media stream when the call is connected. As a result fewer messages are exchanged between the H.323 endpoints than there would be for a Slow Start call (where the fastStart element does not appear). Because media information is sent in the Setup request for the session, there is no need to use the media profiles when converting a Fast Start call to Slow Start.

When you enable the slow start option in the H.323 stack configuration, the Oracle Communications Session Border Controller performs Fast Start to Slow Start conversion. During the translation, the Oracle Communications Session Border Controller retains the media information included in the incoming Fast Start call as it negotiates a connection with the Slow Start endpoint. After a connection with the Slow Start endpoint has been established, the Oracle Communications Session Border Controller negotiates the media capabilities.

Slow Start to Fast Start Translation

When you configure your Oracle Communications Session Border Controller to support H.323 Slow Start to Fast Start translations, you enable an H.323 endpoint that only supports Slow Start to initiate and sustain communication with an H.323 Fast Start endpoint. The Oracle Communications Session Border Controller resolves the Slow Start limitation of exchanging information about media (OLC messages) after the call is connected. The OLC message opens a logical channel, or a unidirectional or bi-directional path used to transmit media packets. Using the Oracle Communications Session Border Controller, you can negotiate the construction of media flows differently, which is described in this section.

When you enable the Fast Start option for calls in the H.323 stack configuration, the Oracle Communications Session Border Controller performs the translation of a Slow Start call into Fast Start. When it receives a Slow Start call, the Oracle Communications Session Border Controller determines its destination and the H.323 stack it uses for the outgoing call.

It is a requirement of this kind of translation that you configure and use media profiles. Since a Slow Start call does not negotiate media until after the call is connected, there needs to be an assumption made about the media to set up a Slow Start to Fast Start call. Media profiles fill this role, and they are assumed to be part of a correct configuration.

The following describes possible scenarios for Slow Start to Fast Start translations.

- When a Slow Start call arrives at the Oracle Communications Session Border Controller and matches one of the session agents that has a media profiles list configured, the outgoing call is set up as a Fast Start call. The session agent's media profiles are used for the logical channels. You must configure the media profiles to reference a codec the endpoint accepts.
  If there are no media profiles configured for the session agent, then the Oracle Communications Session Border Controller uses the media profiles list in the H.323 stack configuration to open the logical channels.
- If a Slow Start calls arrives at the Oracle Communications Session Border Controller and its destination does not match one of the session agents, the Oracle Communications Session Border Controller uses the media profiles list in the H.323 stack configuration for the outgoing call. If there is a list of media profiles, the outgoing call is set up as a Fast Start call with the media profiles list used to open the logical channels.
  If there is no list of media profiles for the outgoing H.323 interface, the Oracle Communications Session Border Controller does not perform Slow Start to Fast Start translation. The Slow Start call exits the Oracle Communications Session Border Controller as it arrived—as a Slow Start call.
- If the egress H.323 interface has the Fast Start option disabled, then the outgoing call uses the Slow Start mode, and the Oracle Communications Session Border Controller does not perform Slow Start to Fast Start translation. In this case, the Slow Start call also exits the Oracle Communications Session Border Controller as it arrived—as a Slow Start call.

Slow Start Fast Start Prerequisites

To perform Fast Start/Slow Start translations, you need to have a standard two-interface configuration already in place.

If you are using the Slow Start to Fast Start translations, you must configure appropriate entries in the media profiles list which is part of the translation parameters. The Fast STart/Slow Start Translations section of the Oracle Communications Session Border Controller Feature chapter describes how the media profiles are used. The list contains the names of media profiles that you configure in the media profile configuration.

Some media profiles are configured by default. If the information you have configured for a media profile does not match up with the defaults, your configuration takes precedence. If there are no configuration overlaps, then the Oracle Communications Session Border Controller loads the configured and default profiles. The default media profiles are:

Type	Payload	Encoding	Bandwidth
audio	0	PCMU	0
audio	2	G726-32	0
audio	4	G723	0
audio	8	PCMA	0

Type	Payload	Encoding	Bandwidth
audio	9	G722	0
audio	15	G728	0
audio	18	G729	0
audio	101	telephone-events	0

Ensure that you use the name of a configured media profile when you enter values in the media profiles list.

Media Profile Configuration

In the ACLI, you can set media profiles that are required for translating H.323 Slow Start to Fast Start. In the ACLI, you set the following:

```
name                encoding name used in sdp rtpmap attribute
media-type          media type used in sdp m lines
payload-type        rtp payload type used in sdp m lines
transport           transport protocol used in sdp rtpmap attribute
req-bandwidth       amount of bandwidth in kilobits required
frames-per-packet   maximum number of frames per packet
parameters          list of <name=value> pairs separated by space
average-rate-limit average rate limit of rtp flow
```

To configure a media profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ```

   From this point, you can configure media profiles parameters. To view all media profiles configuration parameters, enter a **?** at the system prompt.

4. **name**—Enter the encoding name used in the SDP rtpmap attribute. You must enter a name to uniquely identify the media profile, and you will use this value to make lists of media profiles in H.323 interface configurations.
5. **media-type**—Leave this parameter set to its default, **audio**. Valid values are:
   - audio | video | application | data | image | text
6. **payload-type**—Enter the payload type number that corresponds to the encoding name you entered in Step 4. This value identifies the format in the SDP m lines. There is no default value for this parameter. The About Payload Types section contains a table of standard audio and visual encodings.

> **Note:**
>
> When you use the RTP/AVP transport method, this value must be numeric.

7. **transport**—Enter the type of transport protocol used in the SDP rtpmap attribute. The default is **RTP/AVP**. Valid values are:
   - RTP/AVP | UDP
8. **req-bandwidth**—Enter the total bandwidth in kilobits that the media requires. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—4294967295
9. **frames-per-packet**—Enter the maximum number of frames to use per RTP packet. Leaving this parameters set to **0**, its default value means that it is not being used. The valid range is:
   - Minimum—0
   - Maximum—256

     The interpretation of this value varies with codec type and with specific codec.
   - For frame-based codecs, the frame size is specific to each. For example, a G.729 frame contains ten milliseconds of audio, while a G.723.1 codec frame contains thirty milliseconds.
   - For sample-based codecs such as G.711, each frame contains one millisecond of audio.
10. **parameters**—Enter additional codec information. For example, the G.723.1 codec can have an additional silenceSuppression parameter.
11. **average-rate-limit**—Enter the maximum speed in bytes per second for the flow that this media profile applies to. The default value is **0**. The valid range is:
    - Minimum—0
    - Maximum—125000000
12. **peak-rate-limit**—Enter the peak rate for RTP flows in bytes per seconds. The default is **0**. The valid range is:
    - Minimum—0
    - Maximum—125000000
13. **max-burst-size**—Enter the maximum data size at peak rate in bytes. The default is **0**. The valid range is:
    - Minimum—0
    - Maximum—125000000
14. **sdp-bandwidth**—Enable this parameter to use the AS bandwidth modifier in the SDP in the conditions for the application specific bandwidth modifier. The default is **disabled**. Valid values are:
    - enabled | disabled
15. **sdp-rate-limit-headroom**—Specify the percentage of headroom to be added while using the AS bandwidth parameter while calculating the **average-rate-limit** (rate limit for the RTP flow). The default is **0**. The valid range is:

- Minimum—0
- Maximum—100

Fast Start/Slow Start Configurations

When you configure an H.323 interface, you configure it for either Fast Start to Slow Start translation or for Slow Start to Fast Start translation. You cannot configure one H.323 interface for both translation modes.

In the ACLI, you will set the following:

```
fs-in-first-msg       Fast Start must be sent in 1st response to Setup message
call-start-fast       Enable outgoing Fast Start call
call-start-slow       Enable outgoing Slow Start call
media-profiles        list of default media profiles used for outgoing call
```

To configure H.323 interfaces for Fast Start/Slow Start translations:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stacks)#
   ```

   From this point, you can configure H.323 interface and service mode parameters. To view all H.323 interface parameters, enter a **?** at the system prompt. The display also includes H.323 service mode parameters.

5. **fs-in-first-msg**—Enable this parameter if you want to include Fast Start fields in the first message that the Oracle Communications Session Border Controller uses to respond to a Setup message. Usually, the first message sent is a Proceeding message If you do not want Fast Start fields included, leave this parameter set to its default value **disabled**. Valid values are:

   - enabled | disabled

6. **call-start-fast**—Enable this parameter if you want Slow Start calls to be translated to Fast Start when this H.323 interface is chosen as the outgoing interface. If this parameter is **enabled**, **call-start-slow** has to remain disabled. The default value is **enabled**. Valid values are:

   - enabled | disabled

     If you set this parameter set to disabled (default), the outgoing call will be set up in the same mode as the incoming call.

7. **call-start-slow**—Enable this parameter if you want Fast Start calls to be translated to Slow Start when this H.323 interface is chosen as the outgoing interface. If this parameter is **enabled**, **call-start-fast** has to remain disabled. The default value is **disabled**. Valid values are:

   - enabled | disabled

     If you leave this parameter set to **disabled**, the outgoing call will be set up in the same mode as the incoming call.

8. **media-profiles**—Enter the list of media profiles that you want to use when translating Slow Start calls to Fast Start. This information is used to open logical channels for the outgoing call.

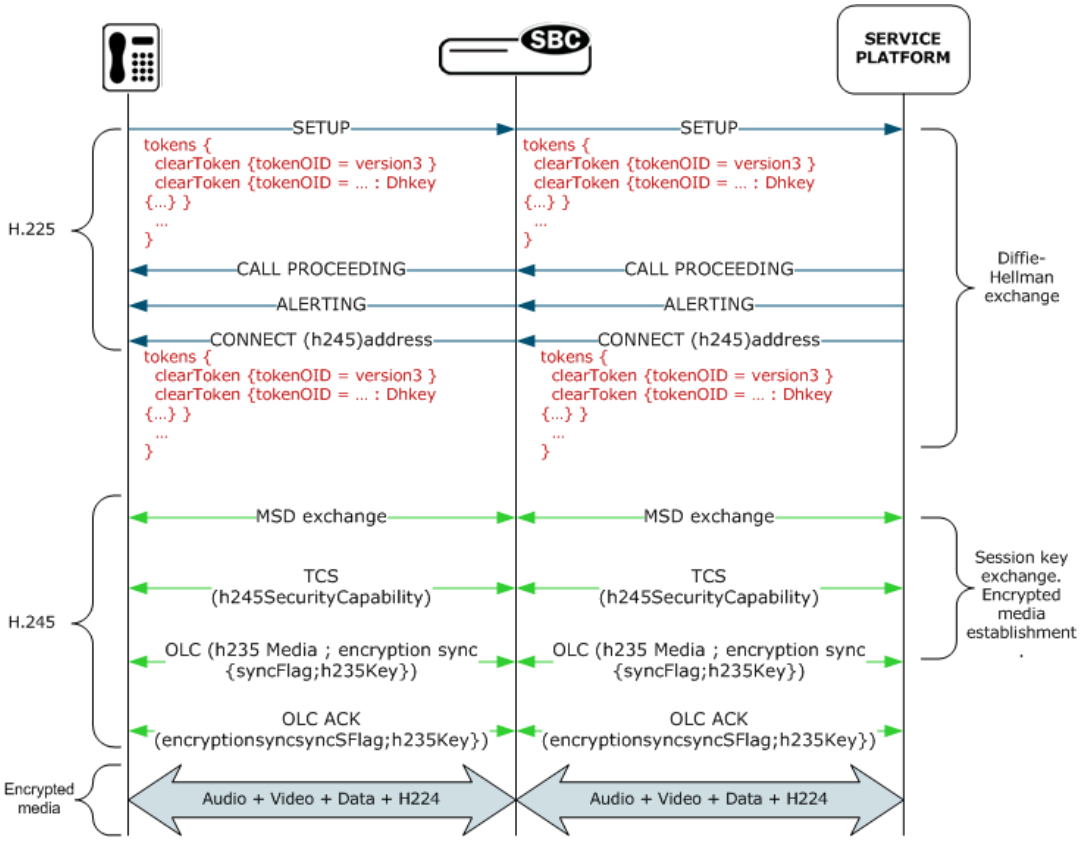
   If you enter the name of a media profile that does not exist, the Oracle Communications Session Border Controller will not perform translation. If you leave this parameter empty, the Oracle Communications Session Border Controller will not perform translation.

H.235 Encryption

Following the ITU-T H.235 encryption standard, the SBC allows media (audio, video, and data) media that has already been encrypted by endpoints to pass through it, thereby supporting videoconferencing applications where media confidentiality is key. The ITU-T standard provides a profile with key management using Diffie-Hellman keys and the specification of an encryption algorithm.

Specifically, the SBC permits the following:

- H.225 Setup and connect—The tokens parameter and its subfields in H.225 Setup and Connect message to pass transparently through the SBC
- H.245Teminal CapabilitySet—The H.245 TerminalCapabilitySet messages to pass transparently through the SBC, including:
  - Audio, video, and data capabilities
  - The h235SecurityCapability capability
- H.245 OpenLogicalChannel and OpenLogicalChannelAck—OLC messages with dataType h235Media to pass transparently through the SBC; to accomplish this, the SBC uses the mediaType subfield instead of the dataType field when the dataType is h235Media. The encryptionSync parameter and its subfields found in OLC and OLCAck messages to pass transparently through the SBC.



You do not need to follow special configuration steps to enable this functionality; it works automatically.

RFC 2833 DTMF Interworking

This section explains the Oracle Communications Session Border Controller's support of transporting Dual Tone Multi-Frequency (DTMF) in Real-Time Transport Protocol (RTP) packets (as described in RFC 2833) to H.245 User Input Indication (UII).

Multimedia devices and applications must exchange user-input DTMF information end-to-end over IP networks. The Oracle Communications Session Border Controller provides the interworking capabilities required to interconnect networks that use different signaling protocols. Also, the Oracle Communications Session Border Controller provides DTMF translation to communicate DTMF across network boundaries.

The Oracle Communications Session Border Controller supports RFC 2833 to H.245 UII translation for H.323-to-H.323 calls, when one side is a version 4 H.323 device requiring RFC-2833 DTMF event packets, and the other side is a pre-version 4 H.323 device that only uses H.245 UII.

About RFC 2833

RFC 2833 specifies a way of encoding DTMF signaling in RTP streams. It does not encode the audio of the tone itself, instead a signal indicates the tone is being sent. RFC 2833 defines how to carry DTMF events in RTP packets. It defines a payload format for carrying DTMF digits used when a gateway detects DTMF on the incoming messages and sends the RTP payload instead of regular audio packets.

About H.245 UII

H.245 provides a capability exchange functionality to allow the negotiation of capabilities and to identify a set of features common to both endpoints. The media and data flows are organized in logical channels. H.245 provides logical channel signaling to allow logical channel open/close and parameter exchange operations. The H.245 signaling protocol is reliable, which ensures that the DTMF tones will be delivered.

H.245 User Input Indication (UII) plays a key role in all the services that require user interaction. For video messaging, typical uses of UII include selection of user preferences, message recording and retrieval, and typical mailbox management functions. H.245 UII provides two levels of UII, alphanumeric and signal.

About 2833 to H.245 UII Interworking

The Oracle Communications Session Border Controller provides 2833 to H.245-UII interworking by checking 2833-enabled RTP streams for packets matching the payload type number for 2833. It then sends the captured packet to the host for processing and translation to H.245 UII messages. A H.245 UII message received by the Oracle Communications Session Border Controller is translated to 2833 packets and inserted into the appropriate RTP stream.

About DTMF Transfer

DTMF transfer is the communication of DTMF across network boundaries. It is widely used in applications such as interactive voice response (IVR) and calling card applications.

The multiple ways to convey DTMF information for packet-based communications include:

- In-band audio: DTMF digit waveforms are encoded the same as voice packets. This method is unreliable for compressed codecs such as G.729 and G.723
- Out-of-band signaling events:
  H.245 defines out-of-band signaling events (UII) for transmitting DTMF information. The H.245 signal or H.245 alphanumeric methods separate DTMF digits from the voice stream and send them through the H.245 signaling channel instead of through the RTP channel. The tones are transported in H.245 UII messages.

  All H.323 version 2 compliant systems are required to support the H.245 alphanumeric method, while support of the H.245 signal method is optional.
- RTP named telephony events (NTE): uses NTE to relay DTMF tones, which provides a standardized means of transporting DTMF tones in RTP packets according to section 3 of RFC 2833.

Of the three RTP payload formats available, the Oracle Communications Session Border Controller supports RTP NTE.

RFC 2833 defines the format of NTE RTP packets used to transport DTMF digits, hookflash, and other telephony events between two peer endpoints. With the NTE method, the endpoints perform per-call negotiation of the DTMF transfer method. They also negotiate to determine the payload type value for the NTE RTP packets.

The NTE payload takes the place of codec data in a standard RTP packet. The payload type number field of the RTP packet header identifies the contents as 2833 NTE. The payload type number is negotiated per call. The local device sends the payload type number to use for 2833 telephone event packets using a SDP or H.245 Terminal Capability Set (TCS), which tells the other side what payload type number to use when sending the named event packets to the

local device. Most devices use payload type number 101 for 2833 packets, although no default is specified in the standard.

The 2833 packet's RTP header also makes use of the timestamp field. Because events often last longer than the 2833 packets sending interval, the timestamp of the first 2833 packet an event represents the beginning reference time for subsequent 2833 packets for that same event. For events that span multiple RTP packets, the RTP timestamp identifies the beginning of the event. As a result, several RTP packets might carry the same timestamp.

See RFC 2833 and draft-ietf-avt-rfc2833bis-07.txt for more information.

Preferred and Transparent 2833

To support preferred (signaled) 2833 and transparent 2833, the Oracle Communications Session Border Controller provides 2833 detection and generation (if necessary) when the endpoint signals support for 2833.

- Preferred: the Oracle Communications Session Border Controller only generates and detects 2833 for endpoints if they negotiate support for 2833 through signaling
- Transparent: the Oracle Communications Session Border Controller behaves as it has prior to this release, offering and answering based on end-to-end signaling and transparently relaying 2833

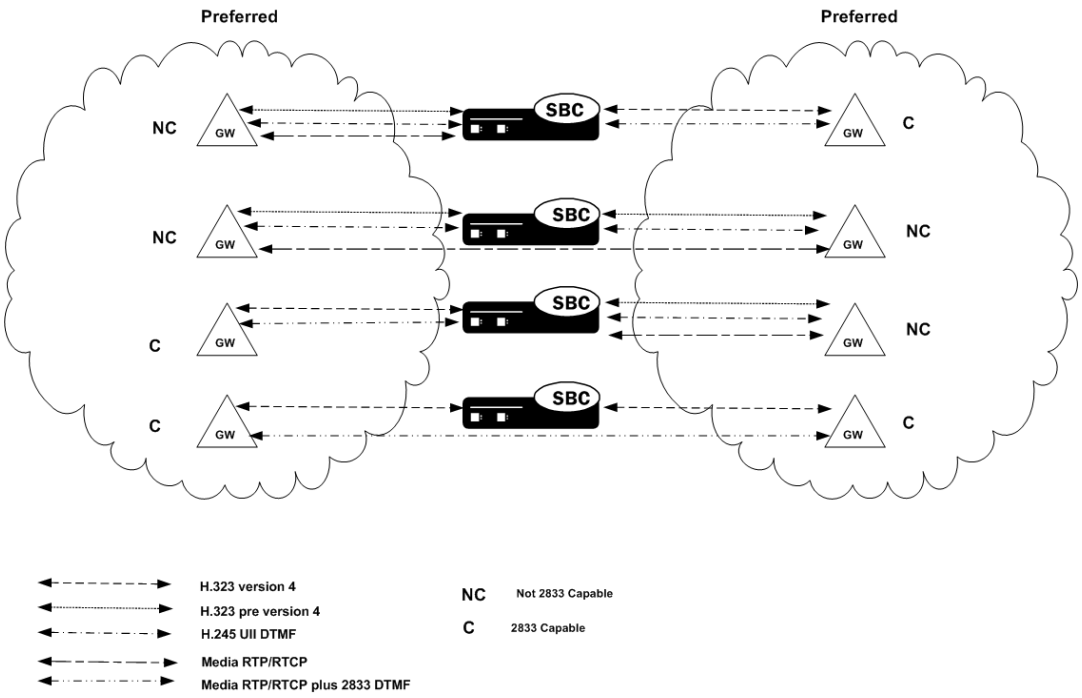
Preferred 2883 Support

If one side of the call, or a session agent, is configured for preferred 2833, the SBC only generates and detects 2833 for endpoints if they signal support for 2833. The SBC will offer 2833 in the TCS SDP, even if the originating caller did not.

- When the SBC manages calls originating from a preferred source going to a preferred target, it:
  Performs 2833 translation for an endpoint when the originating side requests 2833 but the target does not negotiate 2833

  Allows 2833 to pass through if the originating side and target of the call are configured as preferred and negotiate 2833
- When the SBC manages calls originating from a preferred source going to a transparent target, it:
  Performs 2833 translation when the originating side requests 2833 but the target is configured as transparent and does not negotiate 2833.

  Allows 2833 to pass through if the originating side and the target of the call are configured as transparent and negotiate 2833. The SBC does not perform active translation because both ends support 2833.

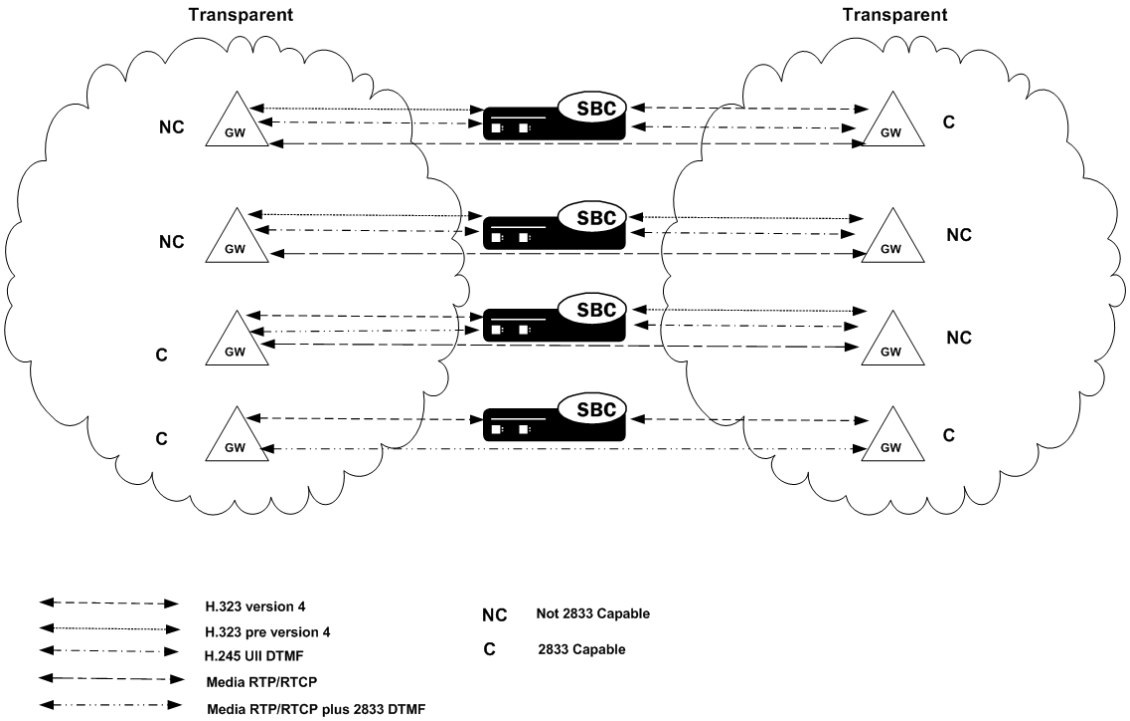


Transparent 2833 Support

The default configuration of the Oracle Communications Session Border Controller for H.323 is transparent 2833. The SBC passes on the offered capabilities to the next-hop signaling element. If the next-hop endpoint is for a transparent 2833 target, typical capability negotiation determines the DTMF method. The SBC transparently relays the DTMF as it has in previous releases.

With transparent 2833, the SBC acts as a typical B2BUA or B2BGW/GK. However when the target of the call is configured as preferred 2833, the Oracle Communications Session Border Controller:

- Relays the 2833 packets if the originating endpoint signals 2833 and the next-hop endpoint for the preferred target signals 2833
- Performs 2833 translation if the originating endpoint does not signal 2833 and the next-hop endpoint for the preferred target does signal 2833
- Does not perform 2833 translation or transparently relay 2833 if the originating endpoint signals 2833 and the next-hop endpoint for the preferred target (or even a transparent 2833 target) does not signal 2833.



Basic RFC 2833 Negotiation Support

If H.323 or session agents on either side of the call are configured for preferred 2833 support, the Oracle Communications Session Border Controller supports end-to-end signaled negotiation of DTMF on a call-by-call basis. If the calling party is not configured for preferred support but sends 2833, the Oracle Communications Session Border Controller sends 2833 to the next-hop called party. If the calling party sends H.245 signals or alphanumeric UII, the Oracle Communications Session Border Controller sends H.245 signals or alphanumeric UII to the next-hop called party (if it is an H.323 next-hop).

The Oracle Communications Session Border Controller also supports hop-by-hop negotiation of DTMF capability on a call-by-call basis, if the signaling protocols or session agents on either side of the call are configured for preferred 2833 support.

H.323 to H.323 Negotiation

The Oracle Communications Session Border Controller serves as the H.323 called gateway. It answers RFC 2833 audio telephony event capability in the version 4 H.323/H.245 TCS when it receives a call from an H.323 endpoint configured for preferred RFC 2833.

If the Oracle Communications Session Border Controller is the answering device, configured for preferred support, and the calling device sends 2833, the Oracle Communications Session Border Controller accepts the 2833 regardless of the next-hop's DTMF capabilities. The received dynamic RTP payload type is used for detecting 2833 packets, while the response dynamic payload type is used for generating 2833 packets.

The Oracle Communications Session Border Controller supports:

- RFC-2833 audio telephony events in the version 4 H.323/H.245 TCS as the H.323 calling gateway, when the Oracle Communications Session Border Controller calls an H.323 endpoint configured for preferred RFC 2833 support. The Oracle Communications Session

Border Controller sends 2833 to the called party regardless of whether the calling party sends it.

- H.245 UII and RFC-2833 packets sent at the same time, to the same endpoint, even if only half of the call is being provided 2833 support by the Oracle Communications Session Border Controller.
  If one half of the call supports H.245 UII, and the other half is being provided 2833 translation by the system, the Oracle Communications Session Border Controller can also forward the H.245 UII it receives to the 2833 endpoint. For example, when the signaling goes through a gatekeeper or third party call control, sending the H.245 UII in the signaling path allows those devices to learn the DTMF digits pressed.

Signal and Alpha Type Support

The Oracle Communications Session Border Controller supports:

- H.245 signal and alpha type UII in the H.323/H.245 TCS as the H.323 calling gateway when the:
  Oracle Communications Session Border Controller calls an H.323 endpoint configured for transparent 2833 support

  calling endpoint's target is configured as preferred

  If the originating preferred side also sends 2833, the Oracle Communications Session Border Controller forwards it to the transparent side. The Oracle Communications Session Border Controller sends signal and alpha UII support to the called party regardless of whether the calling party sends it, if the call originates from a preferred side to a transparent side.
- H.245 alphanumeric UII for DTMF for H.323 endpoints that do not signal 2833 or contain explicit H.245 UII capability, for stacks configured for transparent 2833 support.
  When the other half of the call is an H.323 endpoint of a stack configured for preferred 2833, the Oracle Communications Session Border Controller translates incoming H.245 UII on the transparent side, to 2833 packets on the preferred side, and vice versa. If the other half of the call is an H.323 endpoint of a transparent stack, the Oracle Communications Session Border Controller relays the H.245 UII messages.
- H.245 signal type UII for DTMF for H.323 endpoints that do not signal 2833, but do signal explicit H.245 UII capability, for stacks configured for transparent 2833 support.
  When the other half of the call is an H.323 endpoint of a stack configured for preferred 2833, the Oracle Communications Session Border Controller translates incoming H.245 signaled UII on the transparent side, to 2833 packets on the preferred side, and vice versa. If the other half of the call is an H.323 endpoint of a transparent stack, the Oracle Communications Session Border Controller relays the H.245 UII messages if both sides support it.

H.323 Endpoints

Because there are different H.323 endpoints based on different versions of H.323, the DTMF can be either be transferred out-of-band as UII or in-band using RFC 2833. Most H.323 endpoints:

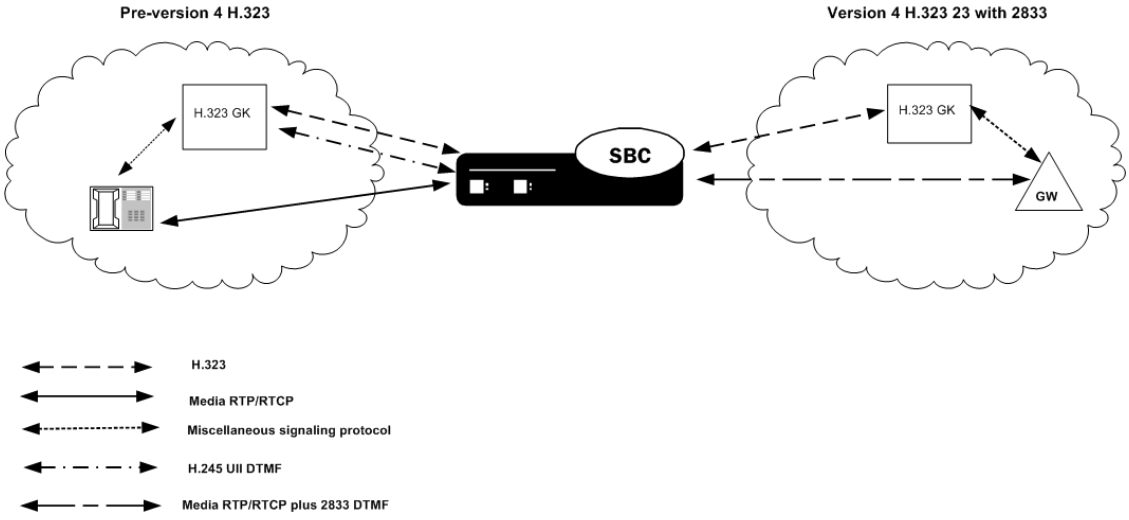
- version 4 and above support RFC 2833
- version 2 and pre-version 4 support UII-Signal
- version 1 and pre-version 2 support UII-Alphanumeric

Translating H.245 UII to 2833 for H.323 Calls

A majority of H.323 endpoints are not version 4 H.323 compliant and do not support RFC 2833 for DTMF transfer. However, some networks include version 4 H.323 devices that require the DTMF events to be signaled in 2833 packets. Network-based version 4 H.323 gateways use RFC 2833 instead of H.245 UII. (Version 4 H.323 devices should support H.245 UII.)

The Oracle Communications Session Border Controller translates 2833 to H.245 UII for H.323-to-H.323 calls when one side is a version 4 H.323 device requiring RFC-2833 DTMF event packets, and the other side is a pre-version 4 H.323 device which only uses H.245 UII.

The Oracle Communications Session Border Controller can translate H.245 UII to RFC2833 and back, based on the admin configuration and H.245 TCS exchanges. This translation enables DTMF to work end-to-end.



RFC 2833 Mode Configuration

To configure RFC 2833 mode:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stack)#
   ```

From this point, you can configure H.323 stack parameters. To view all H.323 stack parameters, enter a **?** at the system prompt.

5. **rfc2833-mode**—Set the RFC2833 mode. The default value is **transparent**. The valid values are:
   - **transparent**:—The Oracle Communications Session Border Controller and H.323 stack behave exactly the same way as before and the 2833 or UII negotiation is transparent to the Oracle Communications Session Border Controller.
   - **preferred**—The H323 stack uses 2833 for DTMF transfer, which it signals in its TCS. However, the remote H323 endpoint makes the decision. If the endpoint supports 2833, 2833 is used. If not, the H.323 stack reverts back to using UII. You configure the payload format by configuring the h323-config element.

RFC 2833 Payload Configuration

To configure the RFC 2833 payload in preferred mode:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# h323
   ```

   From this point, you can configure global H.323 parameters. To view all H.323 configuration parameters, enter a **?** at the system prompt.

4. **rfc2833-payload**—Enter a number that indicates the payload type the Oracle Communications Session Border Controller will use for RFC 2833 packets while interworking 2833 and UII. The default value is **101**.The valid range is:
   - Minimum—96
   - Maximum—127

   You configure session agents with:
   - payload type the Oracle Communications Session Border Controller wants to use for RFC 2833 packets while interworking 2833 and UII.
     The default value for this attribute is **0**. When this value is zero, the global rfc2833-payload configured in the h323-configuration element will be used instead. For SIP session agents, the payload defined in the SIP interface is used, if the SIP interface is configured with the preferred RFC 2833 mode.
   - 2833 mode
     A value of transparent or preferred for the session agent's 2833 mode will override any configuration in the h323-stack configuration element.

RFC 2833 SA Configuration

To configure session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **rfc2833-mode**—Set the RFC 2833 mode you want the session agent to use. The default is **none**. The valid values are:
   - **none**—2833 to UII interworking is based on the H.323 stack configuration.
   - **transparent**:—The 2833 or UII negotiation is transparent to the Oracle Communications Session Border Controller. This overrides the H.323 stack configuration, even if the stack is configured for preferred mode.
   - **preferred**:—2833 for DTMF transfer is preferred, which is signaled in the TCS. If the endpoint supports 2833, 2833 is used. If not, the H.323 stack configured as preferred will revert back to using UII. This overrides any configuration in the h323-stack even if the stack is configured for transparent mode.
5. **rfc2833-payload**—Enter a number that indicates the payload type the session agent will use for RFC 2833 packets while interworking 2833 and UII. The default value is **0**. The valid range is:
   - Minimum—0, 96
   - Maximum—127

H.323 Registration Proxy

The Oracle Communications Session Border Controller provides a registration proxy feature that allows a gatekeeper to authenticate a registration before accepting it. This feature is key when two factors are present: authentication is required, and an RRQ from an endpoint includes a token and/or cryptographic token. If authentication for that endpoint is to work, the Oracle Communications Session Border Controller must forward the registration requests received from the endpoint to the gatekeeper separately.When you do not use the H.323 registration proxy, the Oracle Communications Session Border Controller combines all registrations received from H.323 endpoints into a single RRQ and sends it to the gatekeeper. Using the H.323 registration proxy, you can configure the Oracle Communications Session Border Controller to use separate forwarding.

When registration requests are forwarded separately, each RRQ must have a unique CSA. This means that the Oracle Communications Session Border Controller must perform a one-to-one translation of the CSA in the incoming RRQ to a distinct transport address. The translated address replaces the endpoint's CSA in the outgoing RRQ. Then the Oracle Communications Session Border Controller must listen for incoming calls that arrive at this translated transport address for the registered endpoint.

H.235 Authentication Transparency

When operating in this mode, H.235 authentication tokens (cryptotokens) in RAS messages proxied though the Oracle Communications Session Border Controller are passed through transparently.

For applications where Oracle Communications Session Border Controller is between H.323 gateways and a network hosted gatekeeper, the H.235 cryptotokens are passed through unmodified in RAS messages: RRQs, ARQs, and DRQs. This feature allows for secure gateway authentication.

Unique CSA Per Registered Gateway

When operating in this mode, each CSA is mapped to a registered gateway for call routing. The core gatekeeper does not support additive registrations, so a different CSA must be used for each unique registration that goes to the gatekeeper. The gatekeeper does not overwrite previously registered aliases. Also, since the gatekeeper initiates calls to an endpoint on the CSA specified in the RRQ, the Oracle Communications Session Border Controller must listen on the assigned address for incoming calls to that client as long as the client is registered.

Virtual Call Signaling Address

You can configure the Oracle Communications Session Border Controller with:

- A TCP port range for Q.931—Q.931 ports that are frontend ports handled by a real backend socket, and are therefore virtual
- ATCP port range for separate H.245 TCP connections—Actual sockets that the Oracle Communications Session Border Controller handles separately

Virtual call signaling address is an H.323 call signaling address that is registered with a gatekeeper, but does not have a corresponding listening socket in the Oracle Communications Session Border Controller. Using the virtual call signaling address means that numerous network transport addresses do not need to be allocated.

Virtual call signaling addresses work by attaching a range of TCP server ports to a single listening TCP socket. After a connection is accepted, the accepting socket created by the server socket operated normally, as though it were created by the server socket that listens on the same transport address as the destination of the arriving packet.

To use virtual call signaling addresses, you specify a Q.931 port range from which the Oracle Communications Session Border Controller can allocate ports. This port range is associated with the virtual call signal IPv4 address you specify. To bind dynamic TCP connections to aport within a port range, you configure a dynamic H.245 port range. The dynamic H.245 port range refers to the separate TCP connection for H.245 that takes place when tunneling is not being used. This enables the Oracle Communications Session Border Controller to select the port to which the TCP socket is bound. These two port ranges cannot overlap.

When a new RRQ has to be forwarded to the gatekeeper, the Oracle Communications Session Border Controller caches the registration and then forwards a modified copy of the RRQ. The Oracle Communications Session Border Controller allocates a virtual call signal address on the gateway stack and uses it to replace the CSA of the registering endpoint in the forwarded RRQ.

Virtual RAS Address

The Oracle Communications Session Border Controller also allocates a virtual RAS address for each endpoint registration. Before forwarding an RRQ from an endpoint, the Oracle Communications Session Border Controller replaces the RAS address of the registering endpoint with the virtual RAS address on the gateway interface.

RAS Message Proxy

When the Oracle Communications Session Border Controller's registration proxy feature is configured, RAS messages to and from endpoints are forwarded, except for the following: GRQ, GCF, GRJ, IRQ, IRR, IACK, and INACK. If the system receives a valid GRQ on the RAS port of the gatekeeper stack that supports H.323 registration, it responds with a GCF message. Otherwise, it sends a GRJ message.

If the gateway interface receives IRR or IRQ messages, the Oracle Communications Session Border Controller attempts to respond based on the information about the call, and does not forward the messages.

Other RAS messages are forwarded after some modifications:

- Translating the transport address
- Deleting fields that the Oracle Communications Session Border Controller does not support

About Setting Port Ranges

When you configure the H.323 registration proxy feature, you set the Q.931 port range and the dynamic H.245 port range for H.245 connections. If you configure a Q.931 port range, you must also configure a dynamic H.245 port range.

These port ranges cannot overlap because of TCP ports must be unique. The dynamic H.245 port range is used to allocate a real TCP socket, but the Q.931 port range allocates a virtual call signaling address that does not have an associated listening TCP socket.

> **Note:**
>
> You should choose these sockets with future Oracle Communications Session Border Controller features about security in mind because future development will support performing admission control based on these port ranges. You will be able to set up filtering rules to allow only inbound packets to configured port ranges.

The following table shows how the Q.931 and dynamic H.245 port ranges work. If you set the start port of 1024 and the number of ports to 1024, you will have configured a port range that starts at 1024 and ends at 2047. So the final port in the range is the start port number added to the number of points, minus 1. Remember that you cannot overlap the Q.931 and dynamic H.245 port ranges. Notice that the higher the number of the start ports, the fewer ranges of ports you have remaining from which to choose.

Number of Ports	Start Port	n
1024	1024 * n	1-63
2048	2048 * n	1-31

Number of Ports	Start Port	n
4096	4096 * n	1-15
8192	8192 * n	1-7
16384	16384 * n	1-3
32768	32768 * n	1

H.323 Registration Proxy Configuration

In the ACLI, the parameters that apply to this feature are:

```
q931-start-port       Starting port number for port range used for Q.931
call signalling
q931-number-ports     Number of ports in port range used for Q.931 call
signalling
dynamic-start-port    Starting port number for port range used for
dynamic TCP connections
dynamic-number-ports  Number of ports in port range used for dynamic TCP
connections
```

To configure the H.323 registration proxy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stack** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stack)#
   ```

5. **q931-start-port**—Enter the number where you want the Q.931 port range to start. The default value is **0**. Valid values are:
   - 0 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768
6. **q931-number-ports**—Enter the number of ports to be included in the Q.931 port range to use for the call signalling address forwarded in the RRQ. The default value is **0**. Valid values are:
   - 0 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768

> **Note:**
>
> If you have enabled process registration for this H.323 interface, this value must be set to zero because the interface is a gatekeeper that does not support the virtual call signaling address feature.

7. **dynamic-start-port**—Enter the number where you want the dynamic H.245 port range to start. The default value is **0**. Valid values are:
   - 0 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768
8. **dynamic-number-ports**—Enter the number of ports to be included in the Q.931 port range to use for the call signalling address forwarded in the RRQ. The default value is **0**. Valid values are:
   - 0 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768

H.323 Registration Caching

The Oracle Communications Session Border Controller can cache and proxy an H.225 RRQ between an H.323 endpoint and a gatekeeper. Registration caching has two benefits:

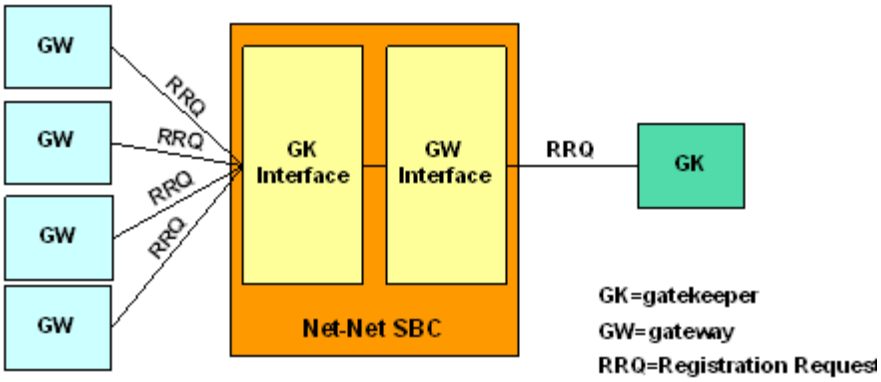
- It allows the aggregation of RRQs sent to a gatekeeper stack and proxies those requests through the gateway stack. If the external gatekeeper associated with the outbound (gateway) interface does not support additive registration, then the SBC consolidates the requests by placing them all in the same packet. Otherwise, additive registration is used on the outbound (gateway) interface.
- It allows the gatekeeper stack to use the registration information to route calls from other realms to the endpoints in its realm.

For registration caching, you need to configure at least two H.323 interfaces:

- One gatekeeper interface to receive registrations
- One gateway interface to proxy registrations

The SBC caches all successful registrations, using the cache to route calls back to the associated endpoint.

The following diagram shows how RRQs flow during registration caching.



Caveats for Registration Caching

This feature has the following caveats:

- If a gateway stack receives a URQ message from the gatekeeper, it confirms the request with an UCF message. It flushes all registration caching for that stack. However, the Oracle Communications Session Border Controller does not send URQs to the registered endpoints.
- The Oracle Communications Session Border Controller must be rebooted so that the gateway interface can rediscover the gatekeeper under the following circumstances:

Automatic gateway discovery is turned on for the gateway interface by setting the automatic gateway discovery parameter to enabled.

Configuration Requirements

For the Oracle Communications Session Border Controller to determine where to route an RRQ, either the associated stack parameter or the gatekeeper identifier field is used.

First, the Oracle Communications Session Border Controller uses the associated interface (assoc-stack) of the gatekeeper interface to find the interface for the outgoing RRQ. If you do not configure an associated interface and the incoming RRQ has a gatekeeperIdentifier field, the Oracle Communications Session Border Controller finds a configured gateway interface with a matching gk-identifier field and use it as the outgoing interface. If the incoming RRQ does not have a gatekeeperIdentifier field and the gatekeeper interface has a configured gatekeeper identifier, the Oracle Communications Session Border Controller finds a gateway interface with a gatekeeper identifier that matches the one set for the gatekeeper interface and then use it as the outgoing interface. If an outgoing interface cannot be determined, the Oracle Communications Session Border Controller rejects the RRQ with the reason discoveryRequired.

A configured H.323 interface can be the gateway interface for more than one gatekeeper interface. If a call is received on the gateway interface, the registration cache will be queried to find a registration matching the call's destination. If a registration is found, the interface on which the registration was received will be used as the outgoing interface for the call.

Subsequent ARQ or URQ messages coming from a registered endpoint will be proxied to the gatekeeper using the outgoing gateway interface established during registration. If a registration is not found, an ARJ or a URJ will be sent to the endpoint originating the ARQ or URQ.

A gatekeeper interface can respond to a GRQ if the GRQ is received on its RAS interface. The Oracle Communications Session Border Controller supports GRQ on a multicast address.

H.323 Registration Caching Configuration

In the ACLI, the parameters that apply to this feature are:

```
isgateway             Enable the stack to run as a gateway
registration-ttl      Number of seconds before the registration becomes
invalid
terminal-alias        List of aliases for terminal
gatekeeper            Gatekeeper's address and port
gk-identifier         Gatekeeper's identifier
```

To configure the gateway interface parameters for registration caching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stack** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stack)#
   ```

5. **isgateway**—Enable H.323 stack functionality as a Gateway. Leave this parameter set to its default, **enabled**, so the H.323 stack runs as a Gateway. When this field is set to **disabled**, the H.323 stack runs as a Gatekeeper proxy.Leave this parameter for the service mode set to it's default, **enabled**. Valid values are:

   - enabled | disabled

     Enabling this parameter ensures that registration with the gatekeeper upon startup. It also ensures that all calls will be preceded by an ARQ to the gatekeeper for admission control.

6. **registration-ttl**—Set the registration expiration parameter to the value of the timeToLive field in the RRQ sent to the gatekeeper. The default is **120**. The valid range is:

   - Minimum—0
   - maximum—4294967295

     When the Oracle Communications Session Border Controller receives an RCF from the gatekeeper, it extracts the timeToLive field and uses that value as the time interval for keeping the registration of the gateway interface alive. The Oracle Communications Session Border Controller sends a keep-alive RRQ about ten seconds before the registration expires.

     The registration expiration you set value should not be too low because some gatekeepers simply accept the timeToLive in the RRQ, resulting in a potentially high volume of RRQs.

7. **terminal-alias**—Set this parameter if the gatekeeper requires at least one terminal alias in an RRQ. On startup, the gateway interface registers with the gatekeeper using this terminal alias.

   When the Oracle Communications Session Border Controller forwards an RRQ from an endpoint and if the gatekeeper does not support additive registration, the RRQ has the interface's terminal alias, the aliases of the registering endpoint, and other aliases of all registered endpoints. Otherwise, the RRQ only contains the aliases of the registering endpoint.

8. **gatekeeper** and **gk-identifier**—Configure these parameters if you do not want the Oracle Communications Session Border Controller to perform automatic gatekeeper discovery. If the gatekeeper identifier is empty, then the Oracle Communications Session Border Controller learns the gatekeeper identifier from the gatekeeperIdentifier field in the GCF.

Configuring the Gatekeeper Interface for Registration Caching

In the ACLI, the parameters that apply to this feature are:

```
isgateway             Enable the stack to run as a gateway
gatekeeper            Gatekeeper's address and port
gk-identifier         Gatekeeper's identifier
registration-ttl      Number of seconds before the registration becomes
invalid
```

To configure the gatekeeper interface parameters for registration caching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stack** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stack)#
   ```

5. **isgateway**—Set this parameter to **disabled** to run the H.323 stack as a Gatekeeper proxy.
6. **gatekeeper**—Leave this parameter empty.
7. **auto-discovery**—Disable the Automatic Gatekeeper discovery feature upon start-up. Set this parameter to **disabled**.
8. **gk-identifier**—Set this parameter to the identification of the gatekeeper to which RRQs received on this interface must be proxied.
9. **registration-ttl**—Enter the number of seconds to set the timeToLive field in the RCF destined for an endpoint. If you do not configure another value, this timer is set to **120** seconds (default).

   This value should not be set too high or too low:

   - Setting a value that is too high causes the registration to be alive too long. If an endpoint reboots during this interval and re-registers with the same terminal aliases (but changes its call signaling address), the registration will be rejected with the reason duplicateAlias.
   - Setting a value that is too low puts an unnecessary load on the Oracle Communications Session Border Controller because it has to handle keep-alive registrations from the endpoint constantly, especially when there are many registered endpoints. If an endpoint does not set the timeToLive field in its RRQ, the registration of that endpoint will not expire.

     If an endpoint registers again without first unregistering itself (e.g., when it crashes and reboots), the Oracle Communications Session Border Controller rejects the registration

using the reason duplicateAlias. The Oracle Communications Session Border Controller uses this reason when the endpoint's call signaling address (IP address and port) is changed but its terminal aliases remain the same.

ACLI Registration Caching Configuration Example

In the following example, the H.323 gatekeeper interface (h323-stack) is private and the gateway interface (h323-stack) is public.

```
h323-config
state                          enabled
log-level                      DEBUG
response-tmo                   4
connect-tmo                    32
h323-stack
name                           private
state                          disabled
realm-id                       private
assoc-stack                    public
local-ip                       192.168.200.99
max-calls                      200
max-channels                   4
registration-ttl               120
terminal-alias
prefixes
ras-port                       1719
auto-gk-discovery              disabled
multicast                      0.0.0.0:0
gatekeeper                     0.0.0.0:0
gk-identifier
q931-port                      1720
alternate-transport
q931-max-calls                 200
h245-tunneling                 disabled
fs-in-first-msg                disabled
call-start-fast                disabled
call-start-slow                disabled
media-profiles
process-registration           enabled
anonymous-connection           disabled
proxy-mode
filename
h323-stack
name                           public
state                          enabled
isgateway                      enabled
realm-id                       public
assoc-stack                    private
local-ip                       192.168.1.99
max-calls                      200
max-channels                   2
registration-ttl               120
terminal-alias
prefixes
ras-port                       1719
auto-gk-discovery              disabled
```

```
multicast                       0.0.0.0:0
gatekeeper                      192.168.1.50:1719
gk-identifier                   gk-public.acme.com
q931-port                       1720
alternate-transport
q931-max-calls                  200
h245-tunneling                  disabled
fs-in-first-msg                 disabled
call-start-fast                 disabled
call-start-slow                 disabled
media-profiles
process-registration            disabled
anonymous-connection            disabled
proxy-mode
filename
```

H.245 Stage

The Oracle Communications Session Border Controller allows you to set the earliest stage in an H.323 call when the Oracle Communications Session Border Controller initiates the procedure to establish an H.245 channel for the call. If you have enabled H.245 tunneling by setting the h245-tunneling parameter to enabled, then you do not need to configure your system for this feature.

The Oracle Communications Session Border Controller initiates the H.245 procedure by either:

- Sending its H.245 address, or
- Creating a TCP connection to an H.245 address that it has received

You can set this parameter to any of the following stages of an H.323 call: setup, proceeding, alerting, connect, early, facility, noh245, and dynamic. With the exception of early, noh245, and dynamic, these values correspond to types of H.225/Q.931 messages. The dynamic value is described in detail in the next section.

When you configure the early value, your Oracle Communications Session Border Controller begins the H.245 procedure at the time the Setup message is sent or received, or when the Connect message is received.

While these values allows for some flexibility about when the H.245 process is started, they are inherently static. All calls in the H.323 stack configuration use the same value, and it cannot be changed from call to call on that stack.

Dynamic H.245 Stage Support

You can configure your Oracle Communications Session Border Controller for dynamic H.245 support, meaning that the point at which the H.245 process begins can be determined dynamically. To support dynamic H.245, the Oracle Communications Session Border Controller sends its H.245 address in the incoming call when it receives an H.245 address in the outgoing call.

Dynamic H.245 Stage for Incoming Calls

When a call comes in on an H.323 interface that you have configured for dynamic H.245 stage support.

The Oracle Communications Session Border Controller includes its H.245 address in the h245Address field of the first H.225/Q.931 message. The Oracle Communications Session Border Controller does this after it receives the first H.225/Q.931 message with an H.245 address in the outgoing call. Based on the first H.225/Q.931 message received by the Oracle Communications Session Border Controller that has an H.245 address, the Oracle Communications Session Border Controller selects the message in which to include the H.245 address as outlined in the table below.

Message Received with H.245 Address	Message Sent with H.245 Address
Call Proceeding	Call Proceeding, Progress, Alerting, Connect or Facility. The H.245 address is sent in the Call Proceeding message if the system has not sent a Call Proceeding message in the incoming call. This is true only when you enable the Fast Start in first message parameter for the incoming stack; this parameter establishes whether or not Fast Start information must be sent in the first response to a Setup message. Otherwise, the message in which the H.245 address is sent depends on what message is received after the Call Proceeding message. This is because the Oracle Communications Session Border Controller sends its Call Proceeding message directly after receiving the Setup message.
Progress	Progress
Alerting	Alerting
Connect	Connect
Facility	Facility

When it receives the first H.225/Q.931 message with an H.245 address in the outgoing call, the Oracle Communications Session Border Controller creates a listening socket on the incoming interface. It also includes the socket address and port in the H.245 address of the next H.225/Q.931 message that it sends. If there is no pending H.225/Q.931 message for the Oracle Communications Session Border Controller to send, it instead sends a Facility message with the reason startH245. Then the H.245 channel is established when a TCP connection is made to the listening socket.

For the outgoing leg of a call that came in on the H.323 stack configured for H.245 dynamic stage support, the Oracle Communications Session Border Controller starts establishing the H.245 channel when it receives the first H.225/Q.931 message with H.245 address information. It also starts to establish a TCP connection to the address and port specified in the H.245 address information. The H.245 channel for the outgoing call is established while the H.245 address (h245Address) is sent in the incoming call as described above.

Dynamic H.245 Stage for Outgoing Calls

This section describes what happens when a message exits the Oracle Communications Session Border Controller on an H.323 stack that you have configured for dynamic H.245 stage support.

When the Oracle Communications Session Border Controller receives the first H.225/Q.931 message that has H.245 address information, it establishes an H.245 channel. The Oracle Communications Session Border Controller initiates a TCP connection to the address and port specified in the H.245 address information.

If the incoming call for the session is also on an H.323 stack with dynamic H.245 configured, the Oracle Communications Session Border Controller starts the H.245 procedure in the

incoming call. Otherwise, the system sends its H.245 address in the incoming call based on the H.245 stage support that you have configured.

The process is different when the Oracle Communications Session Border Controller receives a TCS message on the outgoing call before the incoming call reaches its H.245 stage. In this instance, the Oracle Communications Session Border Controller sends a Facility message with the reason startH245 with its H.245 address in order to start the H.245 procedure. The reason is needed in order for the Oracle Communications Session Border Controller to exchange TCS messages with the incoming side of the call.

H.245 Stage Configuration

To configure H.245 stage support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stacks)#
   ```

5. **h245-stage**—Set this field to the stage at which the Oracle Communications Session Border Controller transfers the H.245 address to the remote side of the call, or acts on the H.245 address sent by the remote side. The default value is **Connect**. Valid values are:
   - Setup | Alerting | Connect | Proceeding | Early | Facility | noh245 | Dynamic

H.323 HNT

This section explains how H.323 hosted NAT traversal (HNT) works and how to enable this capability on your SBC.

The feature enables endpoints behind NATs to originate and terminate calls by resolving the address differences between the NAT and the actual endpoint.

H.323 communication through a NAT becomes an issue when engaging in RAS messaging. While the H.323 standard specifies specific information elements in the RAS messages that indicate the address to which the replies should be sent, these addresses will be behind the NAT and therefore unroutable. The SBC solves this problem by sending RAS replies to the layer 3 address from which the associated RAS request was received.

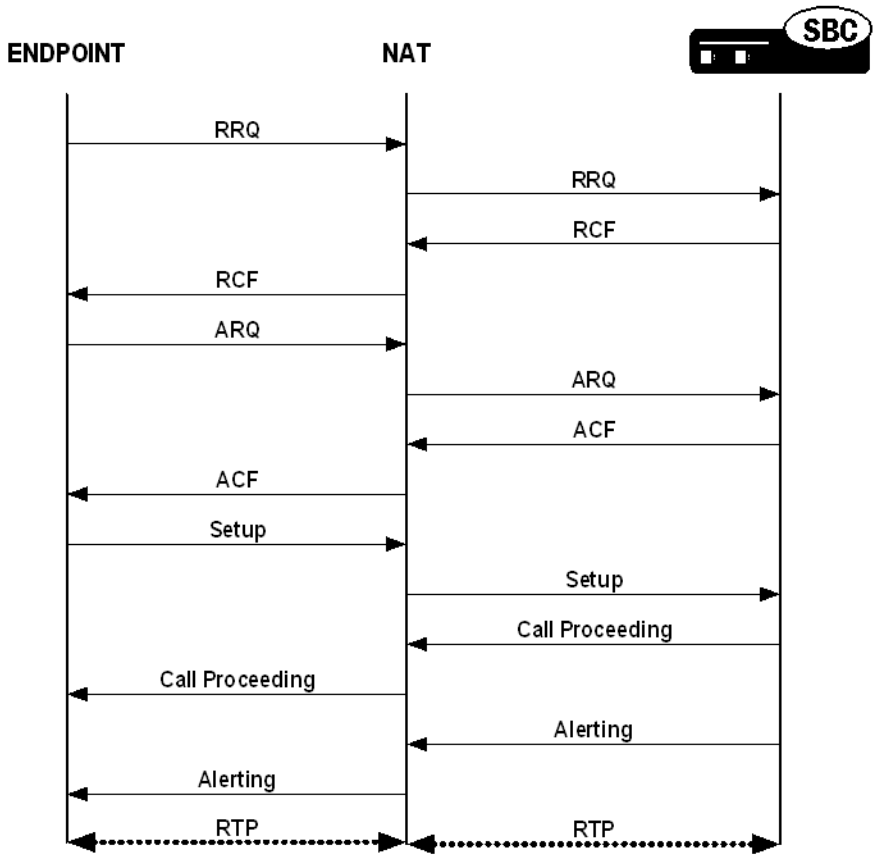
A second issue exists for media channels as the address specified in the H.323 OLC message will be behind the NAT and likewise unroutable. This is resolved by relying on the fact that the forward and reverse channels will utilize the same address and port on the endpoint. By sending media packets to the same address from which the packet are received, media and flow through the NAT.

If you do not use H.323 HNT, the following behavior will occur:

- When an H.323 endpoint is behind a NAT and it registers with a gatekeeper through the SBC, the SBC tries to send a response back to the endpoint's RAS address rather than to the NAT from which the request was received.
- The same is true for LRQ and IRQ messages because responses without H.323 HNT for outbound sessions, responses were being sent back to the replyAddress or the rasAddress.
- In addition, the SBC always induces one-way media because it tries to send the RTP to the media IP address and port it receives in the OLC messages rather than the ephemeral port on the intermediary NAT.

With this ability enabled, however, the SBC sends RAS responses back to the address from which the request was received (the NAT). It does not send responses to the endpoint's rasAddress or replyAddress mentioned in the signaling message. The same is true for RTP. With H.323 HNT for outbound sessions enabled, the SBC sends RTP to the IP address and port from which it receives the RTP packets (the NAT).

The call flow below illustrates how this feature works:



Caveats

Keep in mind the following caveats when you are enabling H.323 HNT for outbound sessions on your Oracle Communications Session Border Controller:

- This capability does not apply to calls that require IWF translation between SIP and H.323.

H.323 HNT Configuration

You can enable this capability for specific H.323 interfaces.

To enable H.323 HNT:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stack** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters for the H.323 interface.

   ```
   ORACLE(h323)# h323-stack
   ```

5. If you are adding this service to a new H.323 interface that you are creating, type **options hnt** (to enable H.323 HNT), and then press Enter.

   ```
   ORACLE(h323-stack)# options hnt
   ```

6. If you are adding this service to an H.323 interface that already exists, type **select** to select the interface to which you want to add the service. Then use the options command and prepend the option with a plus (+) sign.
   - If you know the same of the interface, you can type the name of the interface at the name: prompt and press Enter.
   - If you do not know the name of the interface, press Enter at the name: prompt. A list of interfaces will appear. Type the number corresponding to the interface you want to modify, and press Enter.
   - **If are adding service to an existing interface and you type options hnt without a plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.**

H.323 Party Number-E.164 Support

Some H.323 gateways cannot handle partyNumber alias addresses in H.225 messages. The system lets you convert this address type to dialedDigits (E.164). This conversion applies to sourceAddress, destinationAddress, and destExtraCallInfo aliases in Setup messages.

To enable this feature, use the convertPNToE164 value in the **options** field of the H.323 stack configuration.

Signaling Only Operation

When you set the Oracle Communications Session Border Controller to operate in signaling-only mode, it acts like a signaling server. It proxies the call signaling messages between two endpoints. Note, however, that the NOracle Communications Session Border Controller does not function as a RAS proxy; it does not proxy RAS messages.

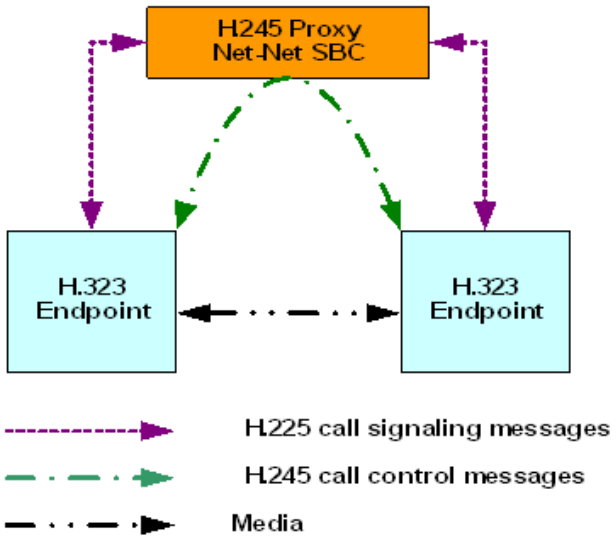
You have two options for the proxy mode:

- H.245 proxy mode—The Oracle Communications Session Border Controller handles call signaling (H.225) and call control (H.245) messages.
- H.225 proxy mode—The Oracle Communications Session Border Controller handles call signaling

To use this feature, you need to set the proxy mode parameter in the H.323 interface configuration to H.225 or H.245.

H.245

When in H.245 proxy mode, the Oracle Communications Session Border Controller proxies or passes through the call signaling (H.225) messages and the call control (H.245) messages. It allows media to flow between the two H.323 endpoints, as shown in the following diagram.



In some deployments, the media might be treated by a NAT device. When the Oracle Communications Session Border Controller is in H.245 proxy mode, any tunneled H.245 message on the ingress side is tunneled in the egress side. However, if the tunneling is refused on the egress side, a separate H.245 session is established.

H.245 proxy mode support is defined in the following table.

Ingress	Egress
Tunneled	Tunneled
Tunneled	Separate H.245 session
Separate H.245 session	Tunneled

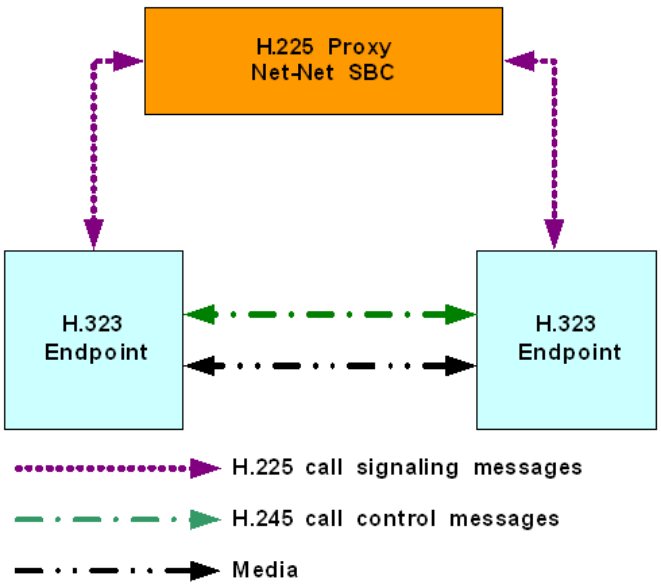
Ingress	Egress
Separate H.245 session	Separate H.245 session

H.225

When in H.225 proxy mode, the Oracle Communications Session Border Controller only proxies call signaling (H.225 messages). The call control (H.245 messages) and the media associated with the session do not go through the Oracle Communications Session Border Controller. Instead, they flow directly between the two H.323 endpoints.

> **Note:**
>
> H.225 proxy mode is only used in specific applications and should not be enabled without consultation from your Acme Packet Systems Engineer.

The following diagram shows the flow.



In certain deployments, the call control message and media are exchanged between the two H.323 endpoints themselves. When the Oracle Communications Session Border Controller is in H.225 proxy mode, any tunneled H.245 message on the ingress side is tunneled in the egress side; this is irrespective of the value configured in the value you set for the **h.245-tunneling** parameter in the H.323 stack configuration.

Maintenance Proxy Function

The Oracle Communications Session Border Controller supports a maintenance proxy function for H.323 and enhances the way the Oracle Communications Session Border Controller creates unique RAS ports. You can register endpoints through the Oracle Communications Session Border Controller with unique RAS port. You can also set the H.323 interface on the enterprise side to represent enterprise-side endpoints and thereby register on the carrier side.

The maintenance proxy creates a many-to-one association between the enterprise and the carrier side. Interfaces on the enterprise side can be associated with the carrier side interface, which also must be configured to for the maintenance proxy feature.

You configure the maintenance proxy feature by simply setting an option in the H.323 interface configuration.

Maintenance Proxy Configuration

To configure the maintenance proxy function, you need to set two values in the options parameters for the H.323 interface (h323-stack):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stacks)#
   ```

5. **options**—Set the options parameter to maintenanceProxy.

Applying TCP Keepalive to the H.323 Interface

To apply these settings individually per H.323 interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stack** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stack)# tcp-keepalive
   ```

5. **tcp-keepalive**—Disable this parameter if you do not want the TCP keepalive network parameters to be applied. The default value is **disabled**. Valid values are:
   - enabled | disabled

6. Click OK at the bottom of the window to complete configuring TCP keepalives and the maintenance proxy.

Automatic Gatekeeper Discovery

Available only when the H.323 interface is functioning as a gateway, this feature allows for automatic gatekeeper discovery on start-up.

This feature is based on the Oracle Communications Session Border Controller sending a GRQ to the multicast address of the RAS Multicast Group, which is the device group listening on this address. If you do not configure a multicast address, Oracle Communications Session Border Controller uses the well-known address and port 224.0.1.41:1718 in the address-port combination making up this parameter.

Multicast only functions when the Oracle Communications Session Border Controller is discovering an external gatekeeper. The Oracle Communications Session Border Controller does not respond to multicast gatekeeper queries.

When it receives a GCF message from a gatekeeper, the Oracle Communications Session Border Controller registers with the gatekeeper indicated in the GCF. When it receives an GRJ message that contains optional information about alternative gatekeepers, the Oracle Communications Session Border Controller attempts to register with an alternate.

If you do not use automatic gatekeeper discovery, the Oracle Communications Session Border Controller registers with the gatekeeper you configure in the gatekeeper parameter. In this case, the gatekeeper identifier you configure is included in to the RRQ. No registration a takes place if you do not establish automatic gatekeeper discovery or if you do not configure the gatekeeper and its identifier.

Automatic Gatekeeper Configuration

To configure automatic gatekeeper discovery:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stacks)#
   ```

5. **auto-gk-discovery**—Enable this parameter to use automatic gatekeeper discovery. The default value is **disabled**. Valid values are:
   - enabled | disabled

6. **multicast**—Set this parameter to the address and port where the RAS Multicast Group listens. Your entries in this field will be comprised of an IPv4 address and port values separated by a colon. The default value is **0.0.0.0:0**.

H.323 Alternate Routing

You can configure your Oracle Communications Session Border Controller to try more possible routes within given time constraints and number of retries.

Without Alternate Routing Enabled

If you do not enable H.323 alternate routing, the Oracle Communications Session Border Controller tries one possible next hop gateway when routing H.323 calls even if the applicable local policy has multiple next hops configured. If that next hop gateway fails (either because it is busy or out of service), the Oracle Communications Session Border Controller relays the failure back to the caller, who hears a busy tone.

In addition, the call will only be routed to the other available next hops if the first one is:

- A session agent that has gone out of service because it constraints have been exceeded
- A session agent that has gone out of service because it failed to respond to a Oracle Communications Session Border Controller Setup request
- A session agent group

With Alternate Routing Enabled

When you enable H.323 Alternate Routing on your Oracle Communications Session Border Controller, you enable the use of the other next hops in addition to the first one. The retry, when the other available next hops are used, is transparent to the caller. However, the number of retries is limited by the value you set for the ACLI **connect-tmo** parameter, and this feature works only if there is more than one matching local policy next hop. If there is not more than one match, even if that match is a session agent group, then the call is only attempted once and the caller must retry it.

If the Oracle Communications Session Border Controller receives a Release Complete message before it receives an Alerting message, then it will try the next hop if there are multiple matches. When there is no more than one match, or if the timer or number of retries is exceeded, the Oracle Communications Session Border Controller proxies the most recently received Release Complete message back to the caller.

The following table shows the cause codes and release complete reasons, and either of the two actions the Oracle Communications Session Border Controller takes:

- Recur—Means that the Oracle Communications Session Border Controller performs (or continues to perform) alternate routing
- Proxy—Means that alternate routing stops, and the Oracle Communications Session Border Controller sends a release complete message back to the caller

H.323 Release Complete Reason	Q.850 Cause Code	Action
No Bandwidth	34—No circuit available	Recur
Gatekeeper Resources	47—Resource unavailable	Recur
Unreachable Destination	3—No route to destination	Recur
Destination Rejection	16—Normal call clearing	Proxy
Invalid Revision	88—Incompatible destination	Recur

H.323 Release Complete Reason	Q.850 Cause Code	Action
No Permission	111—Interworking, unspecified	Recur
Unreachable Gatekeeper	38—Network out of order	Recur
Gateway Resources	42—Switching equipment congestion	Recur
Bad Format Address	28—Invalid number format	Recur
Adaptive Busy	41—Temporary Failure	Recur
In Conference	17—User busy	Proxy
Undefined Reason	31—Normal, unspecified	Recur
Facility Call Deflection	16—Normal, call clearing	Proxy
Security Denied	31—Normal, unspecified	Recur
Called Party Not Registered	20—Subscriber absent	Recur
Caller Not Registered	31—Normal, unspecified	Recur
New Connection Needed	47—Resource Unavailable	Recur
Non Standard Reason	127—Interworking, unspecified	Recur
Replace With Conference Invite	31—Normal, unspecified	Recur
Generic Data Reason	31—Normal, unspecified	Recur
Needed Feature Not Supported	31—Normal, unspecified	Recur
Tunnelled Signaling Rejected	127—Interworking, unspecified	Recur

H.323 Alternate Routing Configuration

This section describes how to enable H.323 alternate routing. There is a new parameter, and the behavior of the pre-existing **response-tmo** and **connect-tmo** parameters change when you enable this feature on your system.

To enable this feature, you need to set the new **alternate-routing** parameter in the global H.323 configuration to recur. The other option for this parameter is proxy, which means that the Oracle Communications Session Border Controller performs in the way it did prior to Release 4.1, i.e. try only the first matching local policy next hop that it finds.

You configure H.323 alternate for the global H.323 configuration.

To enable H.323 alternate routing:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. **alternate-routing**—Enable or disable H.323 alternate routing. If you want to keep the pre-4.1 behavior where the Oracle Communications Session Border Controller only tries one matching local policy next hop, leave this parameter set to its default value **proxy**. Valid values are:

   - recur | proxy

5. **response-tmo**—Enter the time in seconds for the response time-out (or T303 timer). This is the amount of time allowed to elapse during which the Oracle Communications Session Border Controller should receive a response to its Setup message. If the first response to the Oracle Communications Session Border Controller's Setup is a callProceeding, then the Oracle Communications Session Border Controller should receive an Alerting or Connect message before this timer (now T303*2) elapses.

   The default for this parameter is **4**. The valid range is:

   - Minimum—0
   - Maximum—999999999

6. **connect-tmo**—Enter the time in seconds for the connect time-out (or T301 timer). This is the amount of time allowed to elapse during which the Oracle Communications Session Border Controller should receive a Connect message.

   For alternate routing, this parameter is also used to limit the number of next hops that are tried and the length of time they are tried in case the first next hop fails. The call needs to be established before this timer expires; the call will fail after maximum of 5 retries.

   The default for this parameter is **32**.

   - Minimum—0
   - Maximum—999999999

H.323 LRQ Alternate Routing

There are networks where the Oracle Communications Session Border Controller is positioned so that it needs to send an H.225 LRQ request to one signaling entity, and then fall back to another signaling entity when there are no resources available on the first. This might be the case when network contain elements that have limited amounts of channels and/or ports.

To handle situations like this one, the Oracle Communications Session Border Controller can be configured for H.323 LRQ alternate routing.

Without this feature enabled, the Oracle Communications Session Border Controller performs H.323 alternate routing for an H.323 call by finding the alternate route for a local policy when the call setup using H.225/Q.931 fails. Some network configurations, however, require that an LRQ message be sent to a gatekeeper prior to call setup in order to request the destination call signaling address—meaning that the Oracle Communications Session Border Controller will release the call if it does not receive an LCF for that LRQ.

With H.323 LRQ alternate routing enabled, the Oracle Communications Session Border Controller can route the call even when it does not receive the LCF.

When the Oracle Communications Session Border Controller routes an H.323 call using a local policy and the applicable route specifies gatekeeper/session agent as the next hop, the Oracle Communications Session Border Controller must send that gatekeeper an LRQ to request the destination for the call signaling address. After it sends the LRQ, the Oracle Communications Session Border Controller might receive either an LCF or an LRJ, or it might receive no response at all. Upon failure—either the receipt of an LRJ or no response within a timeout period—the Oracle Communications Session Border Controller tries alternate routes (additional routing policies) until the call is either set up or the routing list ends. For each alternate route, if the next hop is a gatekeeper/session agent, the Oracle Communications Session Border Controller sends an LRQ to the gatekeeper in order to request the destination call signaling address. Otherwise, the Oracle Communications Session Border Controller simply sets up the call.

For a designated period of time, the Oracle Communications Session Border Controller waits for the a response to the LRQ from the gatekeeper. This timeout period is configured by setting two options in the global H.323 configuration: **ras-tmo** (number of seconds the Oracle Communications Session Border Controller waits before retransmitting a RAS message; default is 4) and **maxRasRetries** (maximum number of times the Oracle Communications Session Border Controller retransmits the RAS; default is 1). The Oracle Communications Session Border Controller calculates the LRQ timeout period by multiplying the **ras-tmo** by the **maxRasRetries** and adding one (**ras-tmo** x **maxRasRetries** +1).

If an out of service session agent is part of a route, the Oracle Communications Session Border Controller skips it when using alternate routing and uses other routes for the policy.

A session agent might go out of service when it exceeds the maximum number of consecutive transaction timeouts to the maximum number of allowable transaction timeouts. Applicable session agent constrain parameter of note are:

- **trans-timeouts**—Maximum number of allowable transaction timeouts (default is 5)
- **ttr-no-response**—Dictates when the SA (Session Agent) should be put back in service after the SA is taken OOS (Out Of Service) because it did not respond to the Oracle Communications Session Border Controller
- **in-service-period**—Amount of time that elapses before a session agent is put back in service after the **ttr-no-response** period has passed

By default, the Oracle Communications Session Border Controller continues to send LRQ messages to a session agent even if the session agent has already sent an LRJ. However, you might want to place a session agent out of service when it has sent a certain number of LRJs; doing so allows alternate routing to take place faster, but this is an optional feature.

To configure an LRJ threshold, you add the **max-lrj** value to an H.323 session agent's **options** parameter; instructions for how to set it and the required syntax appear below. If you do not set this option, then the Oracle Communications Session Border Controller will not put session agents out of service for matters related to LRJs.

If you do set this option (to a non-zero value), then the Oracle Communications Session Border Controller keeps a count of the LRJs received from a session agent. When it receives an LCF from a session agent, the Oracle Communications Session Border Controller resets the counter to zero. This count is used internally only and is not accessible through statistics displays.

If a session agent exceeds the maximum number of LRJs and goes out of service, it remains in that state until the **ttr-no-response** period has passed and it has transitioned through the **in-service-period** time. If the **ttr-no-response** period is zero, then the session agent is never put out of service.

Caveats

The Oracle Communications Session Border Controller does not support H.323 LRQ alternate routing for these scenarios:

- Calls that require translation between SIP and H.323 (IWF calls)
- For pure H.323 calls where the ingress H.323 interface (stack) is associated with another H.323 interface (stack) that has a valid gatekeeper defined; if there is no valid gatekeeper for the egress interface (stack), this feature may apply.

H.323 LRQ RAS Retransmission Configuration

There is no configuration for H.323 LRQ alternate routing; it is enabled by default. You do, however, need to set the ras-tmo and maxRasRetries options to set the timeout period.

If you want to set a maximum number of consecutive LRJs to be received from a session agent, you need to add the **max-lrj** value to an H.323 session agent's **options** parameter.

To configure the number of seconds before the Oracle Communications Session Border Controller retransmits a RAS message:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ACMEPACKET(h323)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **ras-tmo=x** (where X is number of the seconds that the Oracle Communications Session Border Controller waits before retransmitting a RAS message; default is 4) with a plus sign in front of it, and then press Enter.

Set the maxRasRetries option in the same way; here, X is the maximum number of times the Oracle Communications Session Border Controller retransmits the RAS; default is 1).

```
ORACLE(h323-stack)# options +ras-tmo=6
ORACLE(h323-stack)# options +maxRasRetries=2
```

If you type **options** and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new option to the h323 configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

H.323 LRJ Limit Configuration

To limit the number of LRJs received from an H.323 session agent before putting it out of service:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
```

4. Use the ACLI **select** command so that you can work with the session agent configuration to which you want to add this option.

```
ORACLE(session-agent)# select
```

5. **options**—Set the options parameter by typing **options**, a Space, the option name **max-lrj=X** (where X is the maximum number of allowed LRJs) with a plus sign in front of it, and then press Enter.

```
ORACLE(session-agent)# options +max-lrj=3
```

If you type options max-lrj=X (without the plus sign), you will overwrite any previously configured options. In order to append the new option to the session-agent's options list, you must prepend the new option with a plus sign as shown in the previous example.

H.323 CAC Release Mechanism

When an OLC message is sent to the Oracle Communications Session Border Controller and there is insufficient bandwidth available, the Oracle Communications Session Border Controller will reject the incoming OLC. Normally, endpoints decide whether they want to send new OLCs or if they want to release the call. Some endpoints in this situation do neither. When communicating with the last of endpoints, it is desirable for the Oracle Communications Session Border Controller to take action.

The system supports a option in the H.323 interface called olcRejectTimer. When this option is enabled and an OLC is rejected, the stack will:

- If there is another media channel open, the Oracle Communications Session Border Controller will behave as if the release mechanism had not been enabled
- If there are no media channels open, the Oracle Communications Session Border Controller starts a timer for 1 second.
  - If the call is released by the endpoint before the timer expires or another OLC is received from the endpoint before the timer expires, the Oracle Communications Session Border Controller stops the timer and follows expected call handling
  - If the timer expires before either of the above responses from the endpoint occur, the Oracle Communications Session Border Controller releases the call.

H.323 CAC Release Configuration

To enable the H.323 CAC release mechanism:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the media-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **h323** and press Enter.

```
ORACLE(session-router)# h323
```

4. Type **h323-stacks** and press Enter.

```
ORACLE(h323)# h323-stacks
ORACLE(h323-stacks)#
```

5. Use the ACLI **select** command so can add this feature to an existing H.323 interface.

```
ORACLE(h323-stacks)# select
```

6. Set the **options** parameter by typing **options**, a Space, the option name **olcRejectTimer**, and then press Enter.

   ORACLE(h323-stacks)# **options olcRejectTimer**

7. If you are adding this service to an H.323 interface that already exists, type **select** to select the interface to which you want to add the service. Then use the options command and prepend the option with a plus (+) sign.
   - If you know the same of the interface, you can type the name of the interface at the name: prompt and press Enter.
   - If you do not know the name of the interface, press Enter at the name: prompt. A list of interfaces will appear. Type the number corresponding to the interface you want to modify, and press Enter.
   - **If are adding service to an existing interface and type in the option without a plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign: options +olcRejectTimer.**

H.323 Per-Realm CAC

Building on the Oracle Communications Session Border Controller's pre-existing call admission control methods, CAC can be performed based on how many minutes are being used by SIP or H.323 calls per-realm for a calendar month.

In the realm configuration, you can now set a value representing the maximum number of minutes to use for SIP and H.323 session using that realm. Although the value you configure is in minutes, the Oracle Communications Session Border Controller performs CAC based on this value to the second. When you use this feature for configurations with nested realms, the parent realm will have the total minutes for all its child realms (i.e., at least the sum of minutes configured for the child realms).

The Oracle Communications Session Border Controller calculates the number of minutes used when a call completes, and counts both call legs for a call that uses the same realm for ingress and egress. The total time attributed to a call is the amount of time between connection (H.323 Connect) and disconnect (H.323 Release Complete), regardless of whether media is released or not; there is no pause for calls being placed on hold.

If the number of minutes is exhausted, the Oracle Communications Session Border Controller rejects calls with a SIP 503 Service Unavailable message (including additional information “monthly minutes exceeded). In the event that the limit is reached mid-call, the Oracle Communications Session Border Controller continues with the call that pushed the realm over its threshold but does not accept new calls. When the limit is exceeded, the Oracle Communications Session Border Controller issues an alarm and sends out a trap including the name of the realm; a trap is also sent when the alarm condition clears.

> **Note:**
>
> The Oracle Communications Session Border Controller does not reject GETS/NSEP calls based on monthly minutes CAC.

You can change the value for minutes-based CAC in a realm configuration at any time, though revising the value downward might cause limits to be reached. This value resets to zero (0) at the beginning of every month, and is checkpointed across both system in an HA node. Because this data changes so rapidly, however, the value will not persist across and HA node if both systems undergo simultaneous failure or reboot.

You can use the ACLI **show monthly minutes <realm-id>** command (where **<realm-id> is** the realm identifier of the specific realm for which you want data) to see how many minutes are configured for a realm, how many of those are still available, and how many calls have been rejected due to exceeding the limit.

Caveats

> **Note:**
>
> this feature is not supported for HA nodes running H.323.

H.323 Per-Realm CAC Configuration

This section shows you how to configure minutes-based CAC for realms and how to display minutes-based CAC data for a specific realm.

Note that setting the new monthly-minutes parameters to zero (0), or leaving it set to its default of 0, disables this feature.

To configure minutes-based CAC:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
Oracle Communications Session Border Controller(configure)# media-manager
Oracle Communications Session Border Controller(media-manager)#
```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. Select the realm where you want to want to add SIP per user CAC.

```
ORACLE(realm-config)# select
```

5. **monthly-minutes**—Enter the number of minutes allowed during a calendar month in this realm for SIP and H.323 calls. By default, this parameter is set to zero (0), which disabled monthly minutes-based CAC. You can enter a value as high as 71582788.
6. Save and activate your configuration.

Use the ACLI show monthly-minutes command to see the following information:

- How many minutes are configured for a realm
- How many of those are still available
- How many calls have been rejected due to exceeding the limit

To view information about SIP per user CAC using the IP address mode: In either User or Superuser mode, type **show monthly-minutes <realm-id>**, a Space, and the IP address for which you want to view data. Then press Enter. The **<realm-id>** is the realm identifier for the realm identifier of the specific realm for which you want data.

```
ORACLE# show monthly-minutes private_realm
```

H.323 Bearer-Independent Setup

In Release 4.1, the Oracle Communications Session Border Controller supports a new H.323 option that enables H.323 Bearer-Independent Setup (BIS). When enabled, this feature allows exception to slow-start to fast-start conversion on the Oracle Communications Session Border Controller.

H.323 BIS Disabled

Unless you enable this feature, the Oracle Communications Session Border Controller performs slow-start to fast-start conversion when a call entering the system as slow-start was routed to a an outgoing H.323 interface (stack) with **call-fast-start** set to enabled and there is a list of valid media-profiles in the configuration.

H.323 BIS Enabled

There are certain cases in access deployments where the slow-start to fast-start conversion should not be applied. This is the case when the Setup message contains the Bearer Capability information element (IE), which signals BIS.

When you enable this feature and the Oracle Communications Session Border Controller receives an incoming Setup message that does not contain a fastStart field, the Oracle Communications Session Border Controller checks for the BIS in the incoming Setup before it starts to perform the slow-start to fast-start conversion. If it finds the BIS, then it does not perform the conversion.

This feature can be enabled on a global or a per-interface basis, meaning that you can apply it to your system's entire H.323 configuration or you can enable it only for the interfaces where you want it applied.

H.323 BIS Global Configuration

This section explains how to add H.323 BIS support to your global H.323 configuration and to specific H.323 interfaces (stacks).

If you set this option on an H.323 interface (stack), you must set it on the interface (stack) that receives the Setup message with BIS in the Bearer Capability IE.

To enable the H.323 BIS feature globally:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **options +bearerIndSetup** and press Enter.

   ```
   ORACLE(h323-stacks)# options +bearerIndSetup
   ```

   If you type **options bearerIndSetup** without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

H.323 BIS Specific Configuration

To enable the H.323 BIS feature for a specific H.323 interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stacks)#
   ```

5. Select the H.323 stack to which you want to add H.323 BIS support.

```
ORACLE(h323-stacks)# select
<name>:
```

For a list of configured H.323 interfaces (stacks), press Enter at the <name>: prompt. Then enter the number corresponding to the interface where you want to apply this feature.

6. Type **options +bearerIndSetup** and press Enter.

```
ORACLE(h323-stacks)# options +bearerIndSetup
```

If you type options bearerIndSetup without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

TOS Marking for H.323 Signaling

You can configure your Oracle Communications Session Border Controller to perform TOS/DiffServ marking for H.323 signaling packets. This feature enables you to mark H.323 signaling packets so that they receive specific treatment from upstream devices. This feature assists in routing because you can configure the TOS byte inserted in the H.323 packet to mark the traffic for certain destinations. For example, you can prevent unauthorized video transmission through an audio-only session.

The Oracle Communications Session Border Controller also performs TOS/DiffServ marking for media.

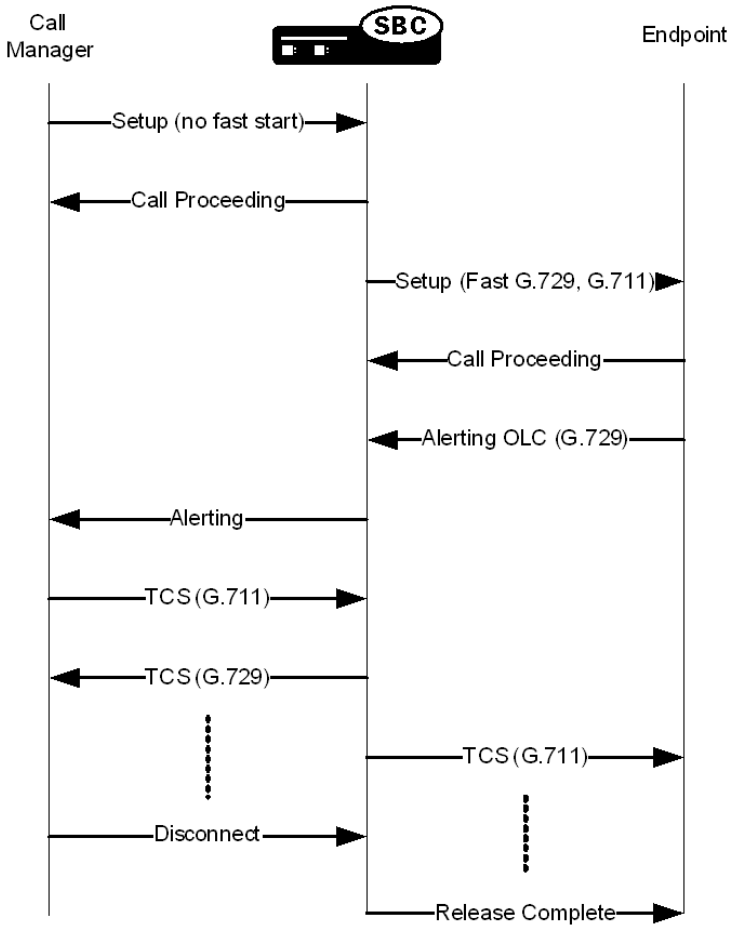
H.323 Codec Fallback

In the global H.323 configuration, you can enable a parameter that allows the Oracle Communications Session Border Controller to renegotiate—or fallback—to the preferred codec used in an incoming terminal capability set (TCS) from the slow-start side of a slow-start to fast-start H.323 call. When enabled, the Oracle Communications Session Border Controller performs this renegotiation when it detects a mismatch between the codec used in the open logical channel (OLC) opened on the fast-start side of the call, and the codec specified by the slow-start side.

Codec Fallback Disabled

With codec fallback disabled, the Oracle Communications Session Border Controller opens a channel using the codec specified by the northbound side. Since the call manager had specified another preferred codec, the result is a codec mismatch leading to a dropped call.

The following diagram shows how codec mismatches end in dropped calls.

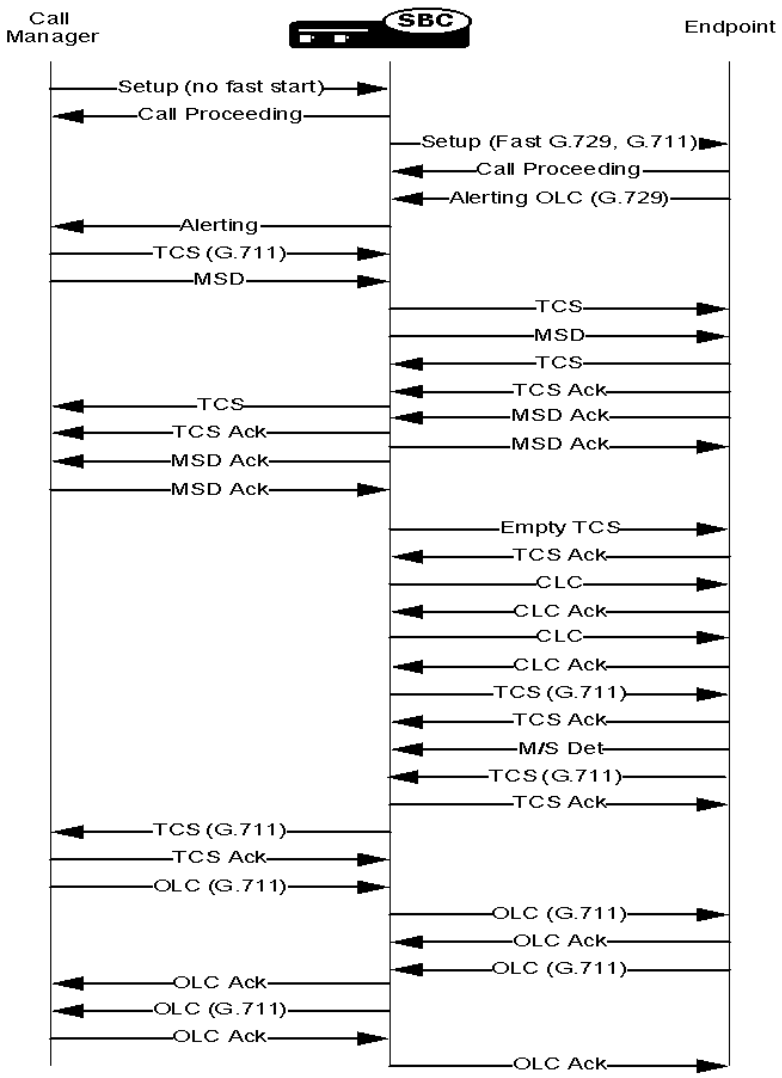


Codec Fallback Enabled

With H.323 codec fall back enabled, the Oracle Communications Session Border Controller attempts to use the preferred codec that the slow-start side of the call specifies. The Oracle Communications Session Border Controller determines matching based on the incoming TCS from the slow-start side and the OLC on the egress side. If the codecs do not match, the Oracle Communications Session Border Controller sends an empty TCS on the egress side and closes the logical channels on the outgoing side of the call.

To trigger a new capabilities exchange, the Oracle Communications Session Border Controller forwards the TCS from the ingress side of the call to the egress endpoint. Then the TCS from the egress endpoint is propagated to the ingress endpoint, and the logical channels are opened.

The following diagram shows a call scenario using the H.323 codec fallback feature.



H.323 Codec Fallback Configuration

Note that you configure this feature for your global H.323 configuration, so it has an impact on all H.323 traffic on your system.

To enable H.323 codec fallback:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter. The system prompt will change to let you know that you can configure individual

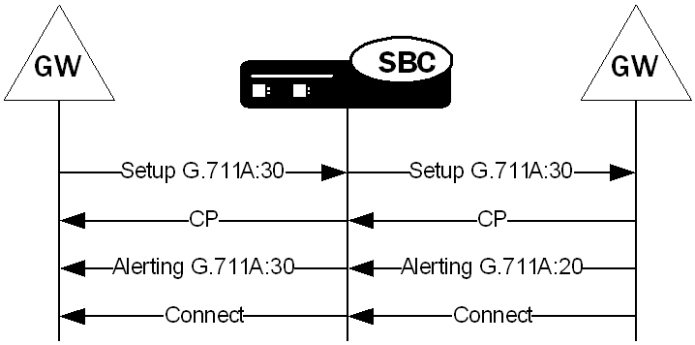
   ```
   ORACLE(session-router)# h323
   ```

4. **codec-fallback**—Enable or disable the H.323 codec fallback feature. The default value is **disabled**. Valid values are:
   - enabled | disabled

H.323 TCS Media Sample Size Preservation

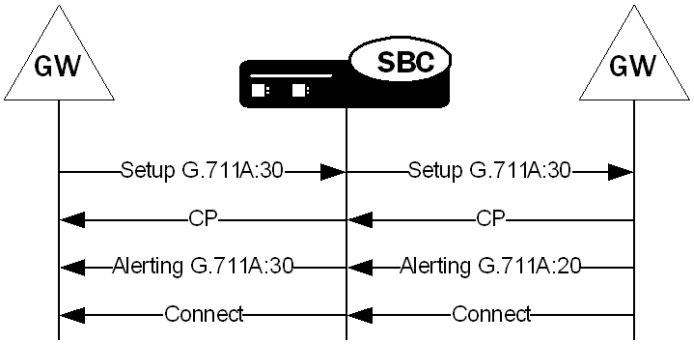
For H.323 fastStart calls, the SBC can be configured to preserve the packetization interval from the called gateway if it differs from the one offered in the Setup message the calling gateway sent.

When this feature is disabled and in accordance with the ITU H.323 recommendation, the SBC changes the packetization rate to the one used by the calling gateway if the one offered by the called gateway differs. In the following example, this means that the SBC replaces the packetization interval of 20 with 30 before it forwards the Alerting message to the calling gateway.



However, not all H.323 elements comply with the ITU recommendation. Since some network elements do modify the packetization rate in the dataType element, this behavior is now configurable.

When you enable media sample size preservation, the SBC allows the packetization rate to be modified and forwards on the modified dataType element to the calling gateway. In the following example, you can see that the SBC forwards the called gateway's Alerting with the packetization interval of 20 despite the fact that the calling gateway's Setup specified 30.



Note that the calling endpoint might or might not work with the modified dataType.

You can enable this feature for the global H.323 configuration so that it applies to all H.323 fastStart calls, or you can enable it on a per-H.323 interface (stack) basis. When you enable this feature for an individual H.323 interface (stack), the SBC performs media sample size preservation for calls egressing on that interface.

Media Sample Size Configuration

This section shows you how to configure media sample size preservation for the global H.323 configuration and for an individual H.323 interface (stack).

To enable media sample size preservation for the global H.323 configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ORACLE(h323)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **forwardFSAceptedDataType** with a plus sign in front of it. Then press Enter.

   ```
   ORACLE(h323)# options +forwardFSAcceptedDataType
   ```

   If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to the h323** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

5. Save and activate your configuration.

   To enable media sample size preservation for an individual H.323 interface:

6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

7. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

8. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ORACLE(h323)#
   ```

9. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stack)#
   ```

   If you are adding support for this feature to a pre-existing H.323 interface (stack), then you must select (using the ACLI **select** command) the one you want to edit.

10. **options**—Set the options parameter by typing **options**, a Space, the option name **forwardFSAceptedDataType** with a plus sign in front of it. Then press Enter.

```
ORACLE(h323-stack)# options +forwardFSAcceptedDataType
```

If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to the h323-stack** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

11. Save and activate your configuration.

H.323-TCS H.245 Support for H.264 and G722.1

The Oracle Communications Session Border Controller supports the H.264 video codec and the G722.1 audio codec. Especially useful for customer video product offerings in which the Oracle Communications Session Border Controller is deployed, this support further allows the Oracle Communications Session Border Controller to increase ease of use by supporting private addressing. Without this feature enabled (the Oracle Communications Session Border Controller's previous behavior), the Oracle Communications Session Border Controller required deployment for IANA registered IP addresses—despite the fact that IP VPNs allow for RFC 1918 private addressing.

H.323-TCS Generic Video Configuration

To enable this feature, you need to set up media profile configurations appropriately. Media profiles now allow you to set the configuration either as "generic video" or generic audio.

H.245 provides for defining new capabilities that are described as H.245 generic capabilities (GenericCapability), which the Oracle Communications Session Border Controller now supports using the H.245 GenericCapability structure. H.264 and G.722.1 are the first codecs the Oracle Communications Session Border Controller offers that use this mechanism.

To set a media profile for generic video support:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **media-profile** and press Enter.

```
ORACLE(session-router)# media-profile
ORACLE(media-profile)#
```

4. **name**—Set the name of the generic video media profile to genericVideo. There is no default for this parameter.
5. **media-type**—Set the media type to use for this media profile; for generic video, set this parameter to video.
6. **payload-type**—Set the payload type to use for the generic video media profile.

7. **transport**—Set the transport type to use for the generic video media profile.
8. Complete the rest of the media profile configuration as needed.
9. Save and activate your configuration.

The following is a sample of a generic video media profile configuration:

```
media-profile
        name                            genericVideo
        media-type                      video
        payload-type                    99
        transport                       RTP/AVP
        req-bandwidth                   0
        frames-per-packet               0
        parameters
        average-rate-limit              0
        sdp-rate-limit-headroom         0
        sdp-bandwidth                   disabled
```

H.323-TCS Generic Audio Configuration

To set a media profile for generic audio support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **media-profile** and press Enter.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. **name**—Set the name of the generic audio media profile to genericAudio. There is no default for this parameter.
5. **media-type**—Set the media type to use for this media profile; for generic video, set this parameter to audio.
6. **payload-type**—Enter the format in SDP m lines. No payload type number is assigned for newer, dynamic codecs. For RTP/AVP media-profile elements, this field should only be configured when there is a standard payload type number that corresponds to the encoding name. Otherwise, this field should be left blank. This field is used by the system to determine the encoding type when the SDP included with a session identifies the standard payload type on the em line, but does not include an a-rtpmap entry.
7. **transport**—Set the type of transport protocol to use for the generic audio media profile. The default value is **RTP/AVP**.
   - UPD | RTP/AVP
8. Complete the rest of the media profile configuration as needed.
9. Save and activate your configuration.

The following is a sample of a generic audio media profile configuration:

```
media-profile
        name                            genericAudio
        media-type                      audio
        payload-type                     104
        transport                        RTP/AVP
        req-bandwidth                    0
        frames-per-packet                0
        parameters
        average-rate-limit               0
        sdp-rate-limit-headroom          0
        sdp-bandwidth                    disabled
```

International Peering with IWF and H.323 Calls

When you do not enable this feature, H.323 calls can default to a National Q.931 Number Type and it is not possible to change it to an International number. This feature allows you to override that behavior by configuring the option **cpnType=X**, where X is an integer that maps to various Q.931 Number Types. When this option is set, Q.931 Number Type for both calling party and called party are updated to the configured value for all outing calls on the h323-stack.

The following is a list of possible **cpnType=X** option values for X:

- 0—Unknown public number
- 1—International public number
- 2—National public number
- 3—Specific public network number
- 4—Public subscriber number
- 5—Public abbreviated number
- 6—Private abbreviated number

You configure this feature as an option in the **h323-stack** configuration.

To configure the cpnType=X option for H323-H323 calls:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **h323-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# h323-config
   ORACLE(h323)#
   ```

4. Type **h323-stacks** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(h323)# h323-stack
ORACLE(h323-stack)#
```

5. Set the options parameter by typing **options**, a Space, the option name **cpnType=x** with a plus sign in front of it, and then press Enter.

```
ORACLE(h323-stack)# options +cpnType=x
```

   If you type **options** without the plus sign, you will overwrite any previously configured options. In order to append the new options to the h323-stack's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

Default OLC Behavior Changed in Upgrade

You can configure the Oracle Communications Session Border Controller to force media profiles in OLC messages when negotiating H.323 calls. This is the default behavior prior to Release S-C6.1.0.

In Release 6.1.0 and forward the default behavior of OLC is to inherit the values recieved in the signaling message from the remote endpoint.

To enable forced media profiles in OLC:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **h323-config** and press Enter.

```
ORACLE(session-router)# h323-config
ORACLE(h323-config)#
```

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **forceMediaProfileInOLC** with a plus sign in front of it, and then press Enter.

```
ORACLE(h323-config)# options +forceMediaProfileInOLC
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Options

The options parameter in the global H.323 and H.323 interface configurations allows you to establish the use of specific features; most of those features are customer specific.

You should exercise caution when you apply options because of the fact that many of them are for customer-specific applications. Consult with your Acme Packet systems engineering to find out if using a particular option would be an advantage to you.

Under no circumstance do we recommend that you configure options without Oracle consultation. There is the chance that you could set an option that might harm an otherwise sound configuration.

Some of the options described below are only applicable to IWF calls. However, you need to establish them in your H.323 configuration.

Global H.323 Options

The following table lists and describes global H.323 options for the Oracle Communications Session Border Controller (SBC).

> **Note:**
>
> Oracle recommends you contact your Oracle account representative before configuring these options.

Options	Description
NoDynamicMSD	SBC forcefully assumes the “primary” role for an outgoing call, and the "secondary" role for an incoming call.
AllowOLCWoMSD	SBC sends OLC before primary - secondary determination is complete. Causes the SBC to be noncompliant with the H.323 recommendation, which does not permit an OLC to be sent prior to MSD completion.
ModifyMediaInAck	SBC accepts and propagates changes to media presented in an OLC Ack. Applies only to Fast Start OLC/OLC Ack messages embedded in H.225 and Q.931 messages during call setup. Causes SBC to be noncompliant with the H.323 recommendation, which does not permit media characteristic to be specified in an OLC to be changed in an OLCAck.
MapG729	SBC maps H.245 G.729 to SDP G.729 with Annex B and vice versa. Applicable only to IWF calls.
ColonG729	SBC uses the : (colon) instead of the = (equal sign) in the media attribute line a=fmtp:18 annexb=yes/no when mapping H.245 G.729 or SDP G.729 with Annex B. Applicable only to IWF calls.
IwfLRQ	SBC sends an INVITE (with no SDP) to a redirect server in response to an incoming LRQ received on an H.323 interface. If a 3xx message with a redirected contact header is returned, the SBC will send an LCF in response to the LRQ. Otherwise, it will send an LRJ.
NoG729AnnexB	SDP received by the IWF with H.729 and no FMTP will be mapped to G.729 on the H.323 side of the call. Can also be set in the session agent options parameter.
sameT38Port	SBC does not allocates separate ports for audio and T.38. SBC will send the same audio port in the OLCAck that it sees in a request mode for T.38 and a new OLC for T.38.

Options	Description
pvtStats	SBC includes program value tree (PVT) statistics in the show h323d display that are a sum of the PVT statistics for all H.323 interfaces. Used for debugging purposes.
strayARQTimer	Required the syntax "strayARQTimer=x," where x is the number of seconds the system waits before tearing down an unsuccessful call in the case of stray ARQs.
forceMediaProfileInOLC	Reverts to older OLC Behavior.

H.323 Interface Options

The following table lists and describes the H.323 interface options.

> **Note:**
>
> Oracle recommends that you contact your Oracle account representative before configuring these options.

Option	Description
stackAliasWins	Oracle Communications Session Border Controller will replace the sourceAddress of the incoming Setup message with the terminal alias of the egress interface when copying the incoming sourceAddress to the outgoing Setup message.
uniqueRRQRASAddress	Oracle Communications Session Border Controller will generate unique rasAddress for each RRQ that it sends to a gatekeeper in response to an incoming RRQ received on an H.323 interface configured for process registration. The IP address will be the local-ip of the outgoing interface, so the port is the unique portion of the rasAddress.
nonV4AdditiveRRQ	Gatekeeper associated with the H.323 interface support additive registration even though it does not set the additiveRegistration field in the RRQ message. When sending in the additive mode, the H.323 interface only sends with the RRQ new terminal aliases that need to be registered. In non-additive mode, the interface sense all the terminal aliases that have been registered, plus the new aliases.
cachedTerimnalAlias	Oracle Communications Session Border Controller copies the terminal alias(es) of the registered endpoint to the asourceAddress field of the Setup message. Terminal alias(es) are changed after the system successfully processes an RRQ from the endpoint.
proxySrcInfo	Oracle Communications Session Border Controller copies the sourceInfo from the incoming Setup message to the outgoing Setup message. Otherwise, Oracle Communications Session Border Controller uses its own endpointType for the sourceInfo field.
noAliasinRCF	Oracle Communications Session Border Controller does not include any terminal alias in the RCF.
forceH245	Oracle Communications Session Border Controllerinitiates an H.245 connection after the call is connected. Otherwise, Oracle Communications Session Border Controller listens for an H.245 connection to be initiated by a remote endpoint.
useCPNInRAS	Oracle Communications Session Border Controller uses the calling party number (CPN) IE of the incoming call as the srcInfo of a RAS message sent in the outgoing call (such as an ARQ).

Option	Description
maintenanceProxy	Oracle Communications Session Border Controller registers interfaces on the enterprise side with a gatekeeper on the carrier side, and registers endpoints through the Oracle Communications Session Border Controller with a unique rasAddress. Interfaces on the enterprise side are associated with the carrier interfaces; you set this option on the carrier side.
convertPNToE164	Oracle Communications Session Border Controller converts the address type partyNumber to dialedDigits (E.164). Conversion applies to sourceAddress, destinationAddress, and destExtraCallInfo aliases in Setup messages.
useCalledPNAsDestInfo	Oracle Communications Session Border Controller uses the H.225 called party number IE as the destinationInfo in ARQ and LRQ requests. Since translation rules can be applied to the Called Party Number, the option enables digit normalization for RAS requests. When not used, Oracle Communications Session Border Controller derives the destinationInfo field in RAS requests from the DestnationAddress field of the incoming Setup.
waitForIncomingH245	On the incoming leg, the Oracle Communications Session Border Controller does not send out its h245Address, but waits for the calling endpoint to send its H245Address.Applies to the outgoing call led as well: The Oracle Communications Session Border Controller does not send out a Facility with startH245 reason and waits for the called endpoint to send its H245Address.
uniqueRRQSrcPort	Enables H.323 RAS Port Mapping. The Oracle Communications Session Border Controller uses the RAS port that it assigned in the rasAddress parameters of an RRQ message as the UDP source port of the outgoing RRQ. Because this feature is linked to the unique RRQ functionality, be aware of the following before you enable the feature: • Enabling H.323 RAS Port Mapping automatically enables the Oracle Communications Session Border Controller's unique RRQ functionality, eliminating the need for you to configure the latter as a separate option. • Enabling the unique RRQ functionality (by setting the uniqueRRQRASAddress option) does not automatically enable H.323 RAS Port Mapping.
srcCallSignallingPort	Enables use of the Q.931 port value for the port field in the sourceCallSignalAddress parameter in an H.225 Setup message. Useful for customers who configure a separate H.323 interface (stack) on the core side for each external IP-PBX.

H.323 Stack Monitoring

In releases prior to S-C6.2.0, the Oracle Communications Session Border Controller provides SNMP monitoring of H.323 session agents but not of the H.323 stacks themselves. The H.323 stack/interface configuration now provides a way for you to set alarm thresholds on a per-stack basis. When enabled, this alarm system ties into the max-calls value to send critical, major, or minor alarms when the number of calls approaches the threshold.

Each H.323 stack now has a threshold crossing alert (TCA) where you can set up three severity levels: critical, major, and minor. You can define one severity level or all three for each stack. To prevent the alarm from firing continuously as call volume through the stack varies, each severity level has an has a reset value below the TCA you set. In addition, each threshold value resets when:

- An alarm with a higher severity is triggered, or
- The built-in reset value for the threshold level is 1% less than the parameter value

RTN 1477

H.323 Stack Monitoring Configuration

This section shows you how to configure H.323 stack monitoring for one H.323 stack configuration. This example shows one instance of the alarm-threshold sub-configuration being established; remember that you can set three—critical, major, and minor. Simply repeat the configuration steps to add more severity levels.

To set up H.323 stack monitoring:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **h323** and press Enter to access the global H.323 configuration.

   ```
   ORACLE(session-router)# h323
   ORACLE(h323)#
   ```

4. **Type h323-stack and press Enter. If you are adding H.323 stack monitoring to an existing H.323 stack configuration, then remember you must select the stack you want to edit.**

   ```
   ORACLE(h323)# h323-stack
   ORACLE(h323-stack)#
   ```

5. Type **alarm-threshold** and press Enter to configure this feature.

   ```
   ORACLE(h323-stack)# alarm-threshold
   ORACLE(alarm-threshold)#
   ```

6. **severity**—Enter the type of severity level for the alarm you want to define. Choose from: **critical**, **major**, or **minor**. This value is required, and defaults to **minor**.
7. **value**—Enter the percentage of the number of calls defined in the **max-calls** parameter that triggers the alarm. For example, if you want to set a minor alarm to fire when the call rate through the stack reaches half the **max-calls** value, enter **50** (meaning 50%). The default value for this parameter is 0, which disables the alarm.

   Remember that if the number of calls falls to below 1% of the max-calls threshold you set, the clear trap fires.
8. Save your work. You can see the data related to this feature using the ACLI **display-alarms** and **show h323 stack stack-alarms** commands.

H.323 Automatic Features

This section describes H.323 features that are automatically enabled on your Oracle Communications Session Border Controller. You do not have to configure special parameters

to turn them on. Even though you do not have to turn these features on, this section describes what they do and how they work.

Alias Mapping

Alias mapping permits destination addresses to be modified by a gatekeeper.

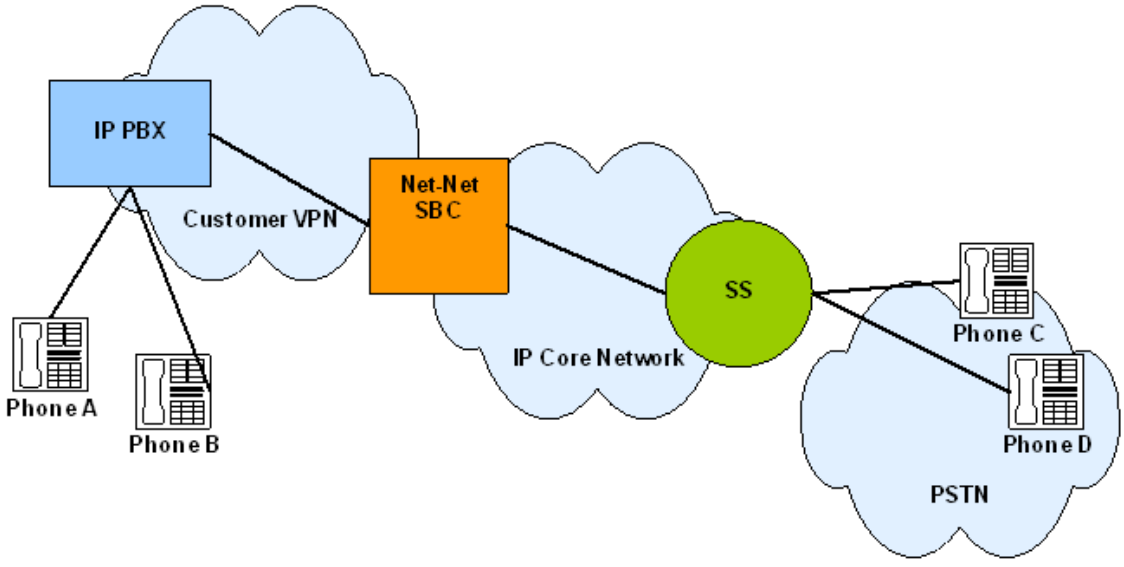
When sending an ARQ or an LRQ message to a gatekeeper, the Oracle Communications Session Border Controller sets the canMapAlias field in that message to true. This setting indicates that the Oracle Communications Session Border Controller accepts modified destination information from the gatekeeper. If the resulting ACF or LCF contains destinationInfo and/or destExtraCallInfo fields, then the Oracle Communications Session Border Controller copies that information respectively to the destinationAddress and destExtraCallInfo fields of the Setup message. In addition, if the destinationInfo is either type e164 or type partyNumber, the Oracle Communications Session Border Controller copies the information into the calledPartyNumber information element (IE) of the Setup message, replacing the existing calledPartyNumber IE.

You do not need to configure special parameters for this feature; it is enabled automatically.

Call Hold and Transfer

The SBC's H.323 call hold and transfer feature supports consultation in addition to call holder and transfer. This feature uses signaling procedures based on the ITU-T recommendations/ H.323 specification for what it calls third party initiated pause and rerouting.

The following diagram shows how the SBC is positioned to provide call hold and transfer support for H.323.



Call Hold and Transfer Basic Call

The following diagram show the signaling and media flows between the IP PBX and a softswitch. Note how the SBC is position to mediate flows between the two devices.

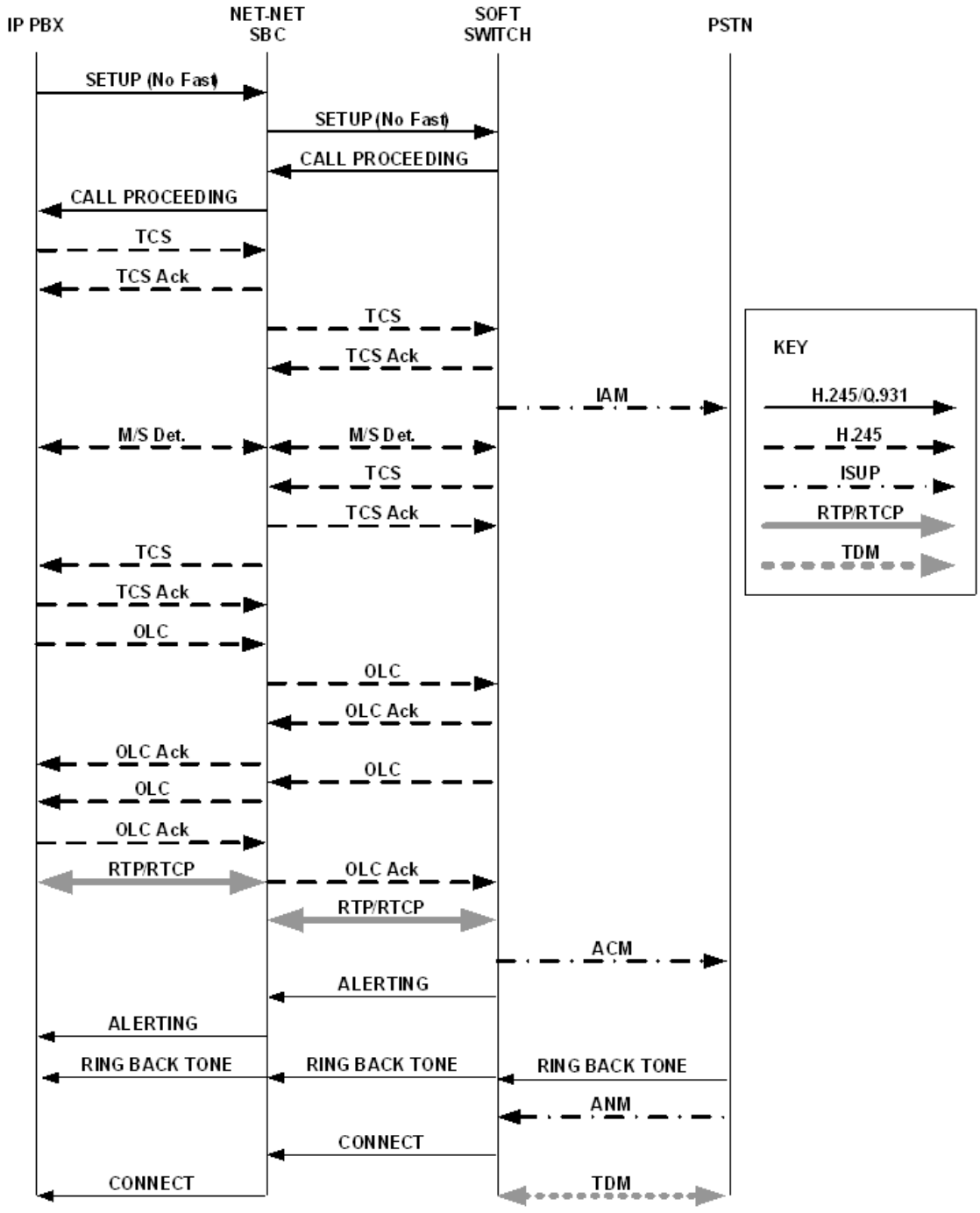
In the Call Proceeding messages forwarded to the IP PBX, the SBC uses a non-zero value to ensure that the IP PBX initiates an H.245 session. A progress indicator does not need to be

included if the H.245 address is present in any of the following message types: Alerting, Progress, or Connect.

After the SBC receives a Call Proceeding message from the softswitch that contains the H.245 address, the SBC sends another Call Proceeding with its own H.245 address.

In the following call flow, the softswitch generates message to the gateway. These messages are:

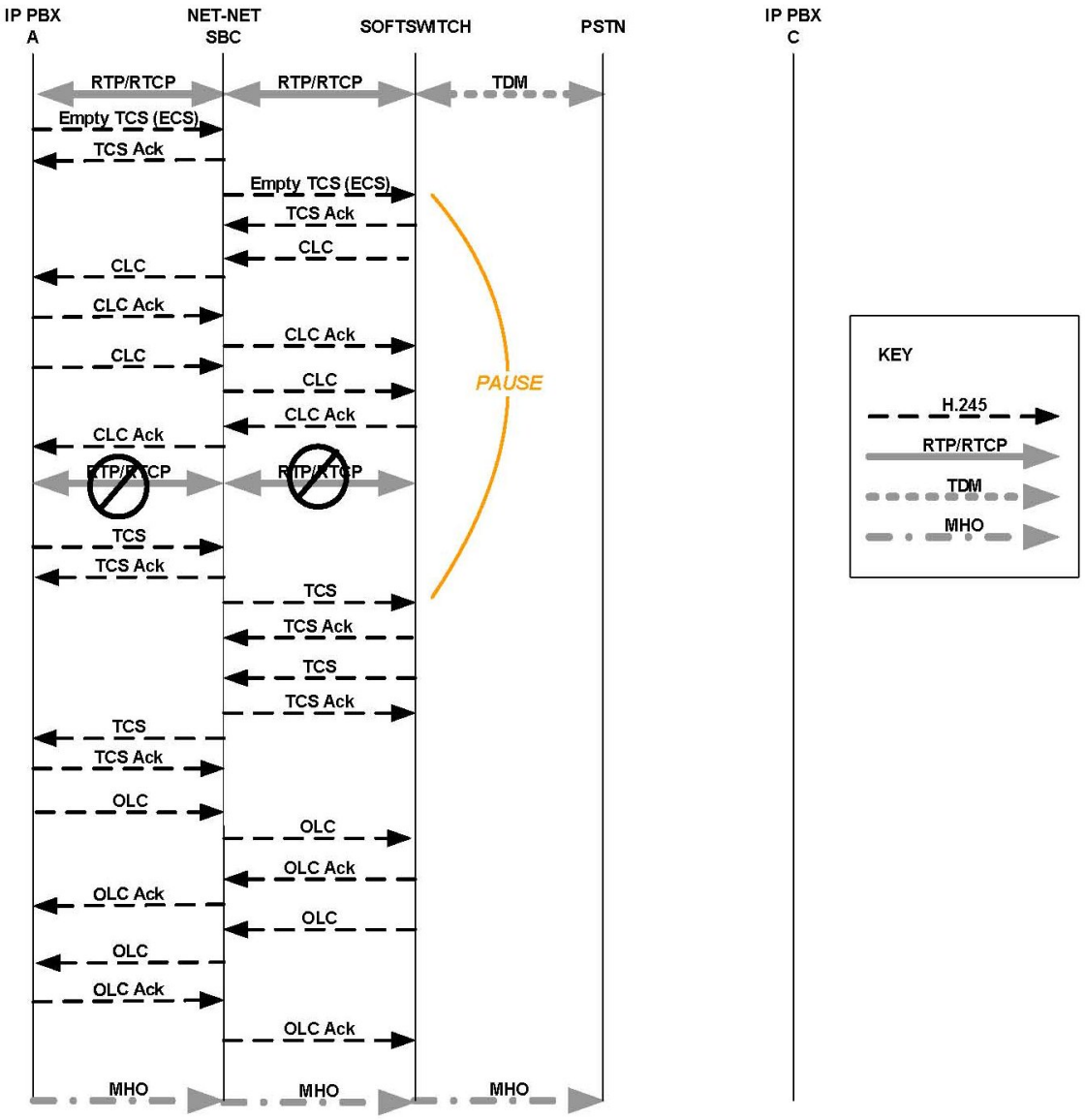
- Initial Address Message (IAM)
- Address Complete Message (ACM)
- Answer Message (ANM)



Call Hold and Transfer Music on Hold

The following diagram begins with the condition that IP PBX A is already connected with a gateway, with the SBC and the softswitch positioned between the two.

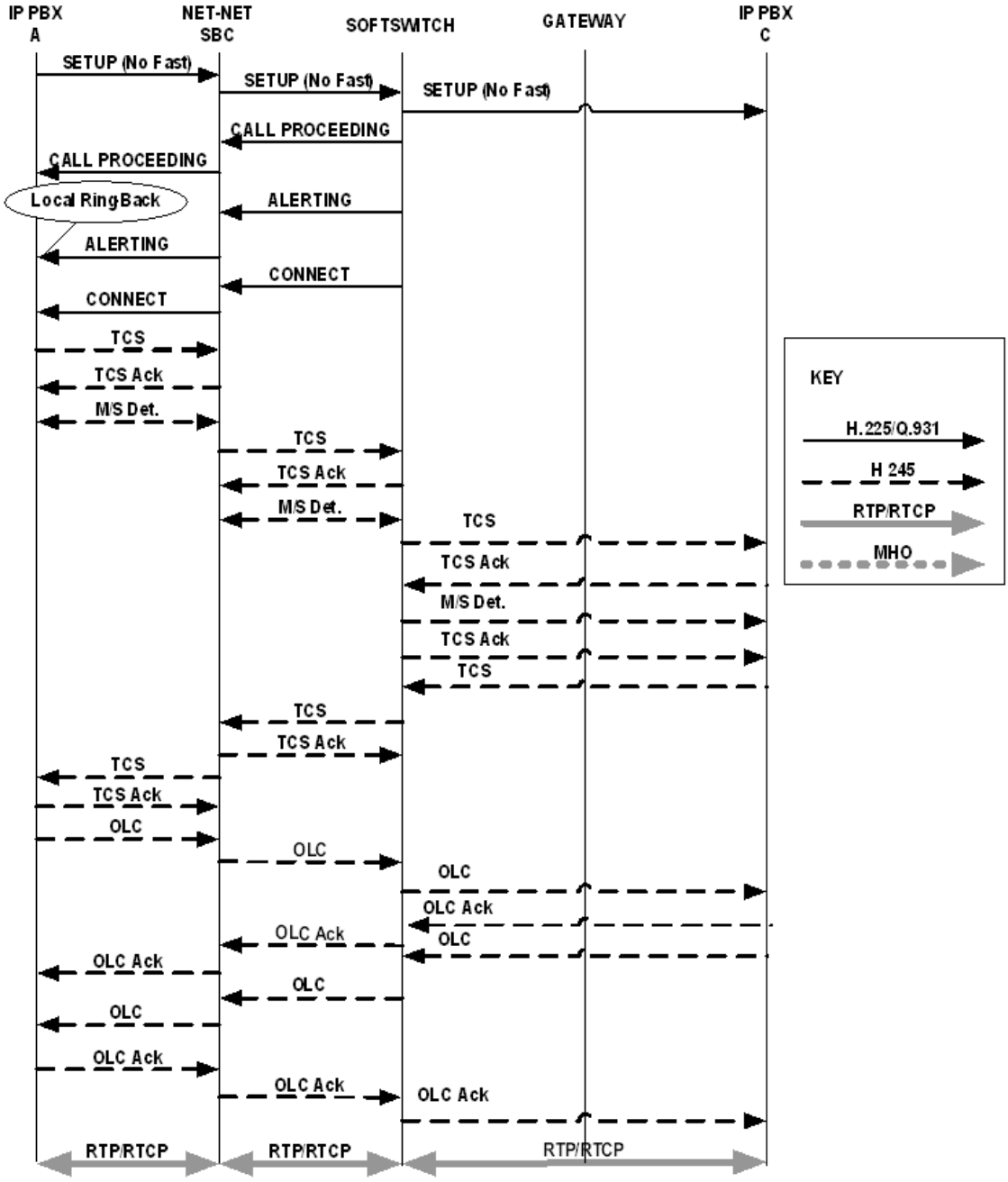
You can see in the call flow where the channels for transporting media are closed, and where the RTP/RTCP is stopped. This creates a pause for the call. With the SBC mediating the process, IP PBX A and the softswitch exchange TCS and OLC messages that allow music on hold (MHO) to flow between IP PBX A and the gateway.

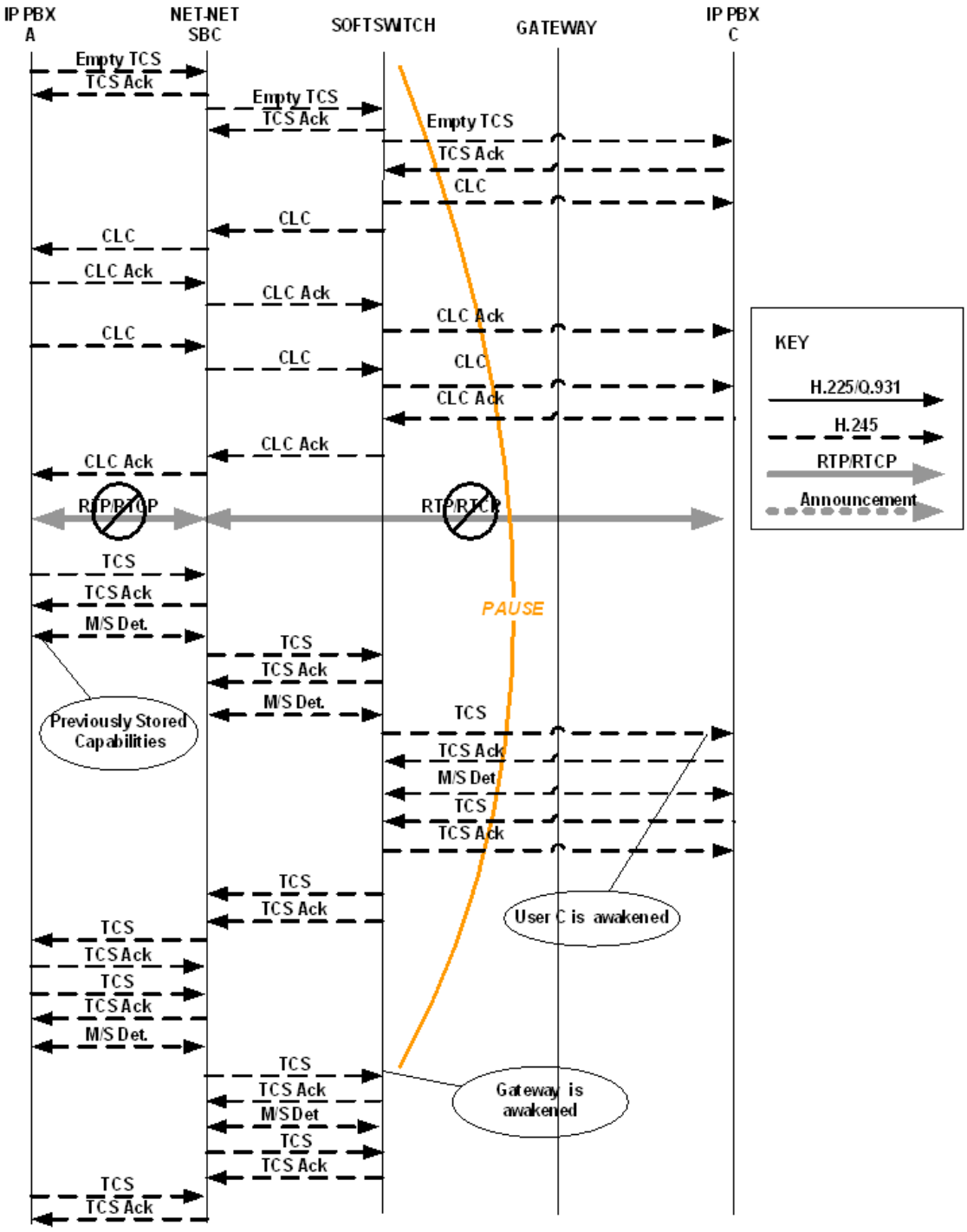


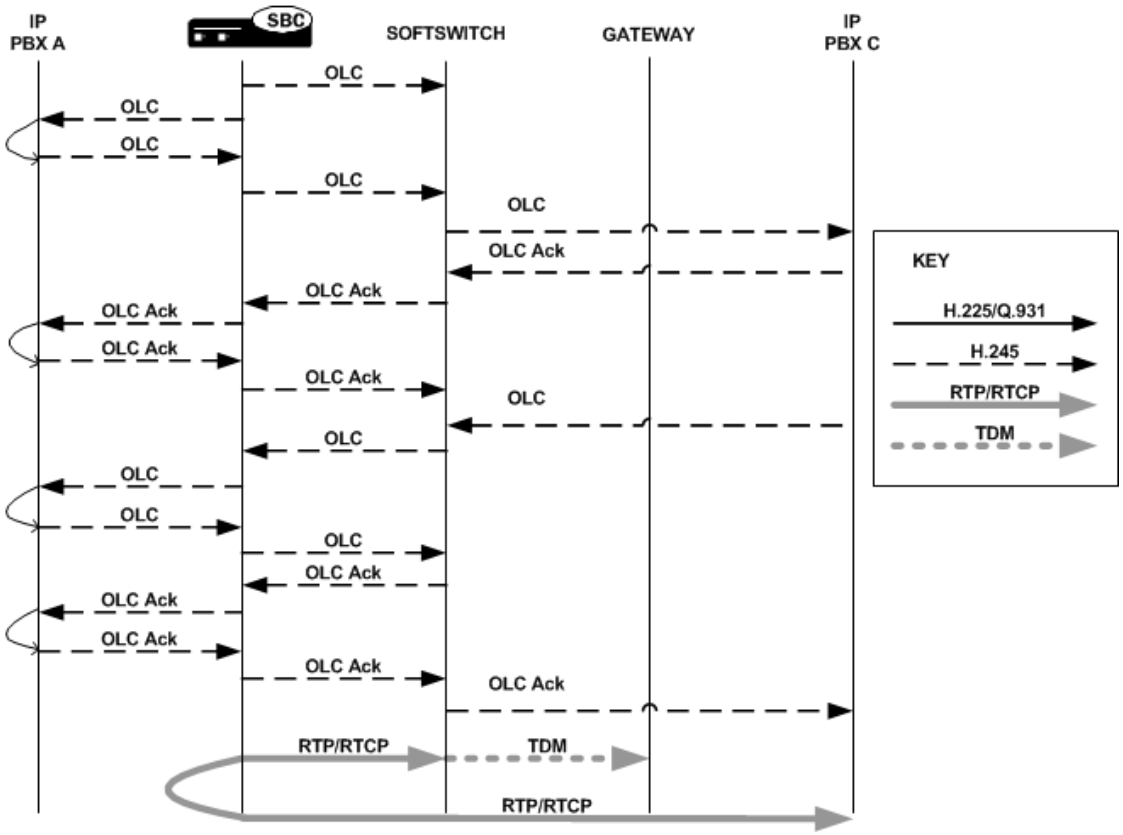
Call Hold and Transfer

The following diagram shows how call transfer works on the SBC for H.323. In this diagram, you can see:

- Where local ringback occurs
- Where the pause begins and ends
- Where users and gateways are awakened
- Where logical channels are opened and closed




IP PBX A
NET-NET SBC
SOFTSWITCH
GATEWAY
IP PBX C
Empty TCS
TCS Ack
Empty TCS
TCS Ack
Empty TCS
TCS Ack
CLC
CLC
CLC
CLC Ack
CLC Ack
CLC Ack
CLC
CLC
CLC
CLC Ack
CLC Ack
CLC Ack
RTP/RTCP
RTP/RTCP
TCS
TCS Ack
M/S Det.
PAUSE
TCS
TCS Ack
M/S Det.
Previously Stored Capabilities
TCS
TCS Ack
M/S Det
TCS
TCS Ack
TCS
TCS Ack
User C is awakened
TCS
TCS Ack
TCS
TCS Ack
M/S Det.
TCS
TCS Ack
M/S Det
TCS
TCS Ack
Gateway is awakened
TCS
TCS Ack
KEY
H.225/Q.931
H.245
RTP/RTCP
Announcement



Media Release for SS-FS Calls

When the Oracle Communications Session Border Controller routes a slow-start to fast-start call, it is possible for the same fast-start call to be routed back through the Oracle Communications Session Border Controller making for a hairpin flow. If it does becomes a hairpin flow, then the Oracle Communications Session Border Controller routes it to its destination as a fast-start to fast-start call. This can result in one-way media if:

- The destination of the hairpin call is in the same realm as the originating slow-start to fast-start call
- The realm reference in the first bullet item is configured to disable in-realm media management
- The called endpoint accepts the proposed fast-start logical channels

The enhancements to the Oracle Communications Session Border Controller's behavior described in this section show how the Oracle Communications Session Border Controller follows additional procedures when setting up a hairpin flow to avoid one-way media when media release occurs.

For H.323 calls, the Oracle Communications Session Border Controllerestablishes media using the H.245 procedures described in the H.245 ITU-T recommendation: control protocol for multimedia communication. It also uses the Fast Connect procedure defined in the H.323 ITU-T recommendation: packet-based multimedia communication systems.

The latter ITU-T recommendation allows a calling endpoint to send a Setup message that contains a fastStart element, a sequence of OLC structures that describe the calling endpoint's proposed forward/reverse logical channels. If the called endpoint accepts this proposal, then logical channels are established.

When the Oracle Communications Session Border Controller translates a call originating in slow-start to fast-start, it uses a Fast Connect procedure in the outgoing leg by sending an outgoing Setup that includes a fastStart element with one or more OLC structures. But when the Oracle Communications Session Border Controller constructs this message, it is unaware of whether the call will become hairpinned or if media release will occur. Because it does not yet have this information, the Oracle Communications Session Border Controller sets the Network Address and the TSAP identifier in the OLC structures to the ingress IP address and port of a corresponding media flow allocated for media traveling between the calling and called endpoints. So if the called endpoint accepts the fastStart the Oracle Communications Session Border Controller proposes, the called endpoint would send its media to the Oracle Communications Session Border Controller. After acceptance, the system starts H.245 procedures on the slow-start side of the call to set up logical channels on that side. Then the Oracle Communications Session Border Controller updates the IP address and port of the media flows using OLC and OLCAck messages received from the calling endpoint.

This procedure works well for endpoints that are not in the same realm, or that are in the same realm for which media management is disabled, because each endpoint must send its media through the Oracle Communications Session Border Controller. When the endpoints are in the same realm and when media management is enabled, however, the Oracle Communications Session Border Controller must perform additional steps for media release in slow-start to fast-start calls.

To support media release in slow-start to fast-start calls, the Oracle Communications Session Border Controller performs a hold-and-resume procedure on the fast-start side. After it establishes channels on the slow-start side and if it detects media release being enabled, the Oracle Communications Session Border Controller sends and empty TCS to the fast-start side to put that side on hold. Then the called endpoint closes all the logical channels it previously opened in the Fast Connect procedure and stops transmitting to them. And the Oracle Communications Session Border Controller also closes it logical channels. Once the channels are closed, the Oracle Communications Session Border Controller resumes the call by sending a new, restricted TCS to the fast-start side. The restricted TCS only contains the receive and transmit capabilities of the codecs types that the called endpoint accepted in the Fast Connect procedure, and it forces the called endpoint to re-open logical channels of the same codec types accepted in the Fast Connect procedure. Once it receives and OLC from the called endpoint, the Oracle Communications Session Border Controller sends on OLCAck with the Network Address and TSAP identifier for the logical channel from the calling endpoint. Then the Oracle Communications Session Border Controller re-opens logical channels (of the same codec types that it open in the Fast Connect procedure). If the called endpoint has not changed its Network Address and TSAP identifier for its logical channels, media is re-established after the Oracle Communications Session Border Controller and the called endpoint exit the hold state. The last steps is for the Oracle Communications Session Border Controller to re-sends the full TCS message from the calling to the called endpoint to inform the called endpoint of the full capabilities of the calling endpoint.

Dependencies

This feature depends on the following assumptions:

- The H.323 endpoint supports the third-party-initiated pause and re-routing feature.
- The H.323 endpoint does not change its Network Address and TSAP identifier when it re-opens the logical channels.
- The H.323 endpoint does not immediately tear down the call when there is not established logical channel in the call.

Hold-and-Resume Procedure

The hold-and-resume procedure has three states:

- Media Hold—Starts when the Oracle Communications Session Border Controller (SBC) sends the empty TCS to the called endpoint to put it on hold.
  When the SBC detects media release, it puts the called endpoint on hold. It can only do so if it has exchanged the TCS and TCSAck messages and completed primary-secondary determination with the calling endpoint.

  When the SBC receives a TCSAck in response to the empty TCS that it sent to the called endpoint, it closes the logical channels it opened as part of the Fast Connect procedure; the called endpoint likewise closes its logical channels. The two then exchange CLC and CLCAck messages, which signals the start of the Media Resume state.
- Media Resume—Starts when the SBC sends a restricted TCS to resume the call.
  The restricted TCS the SBC sends contains only the receive and transmit capabilities of the codec types previously accepted by the called endpoint in the Fast Connect procedure. This forces the called endpoint to re-open logical channels of the same codec type that were previously accepted in the Fast Connect procedure.

  After sending this TCS, the system is ready (as specified in the ITU-T recommendations) to take part on the primary-secondary determination (MSD) process. However, the called party and not the SBC initiates the MSD if it is required. The MSD is completed if necessary. Alternately, the called endpoint can start to re-open its logical channels. When it receives the first OLC from the called endpoint, the SBC also starts to re-open its logical channels.
- Media Complete—Starts when all the logical channels that the SBC re-opens are acknowledged by the called endpoint.

When it enters the Media Complete state, the SBC updates the called endpoint with the full capabilities of the calling endpoint by sending the full TCS.

H.323 and IWF Call Forwarding

This section describes the Oracle Communications Session Border Controller's H.323 and IWF Call Forwarding feature, which is supported for H.323 calls and for calls initiated in SIP that require interworking to H.323.

Previous Behavior

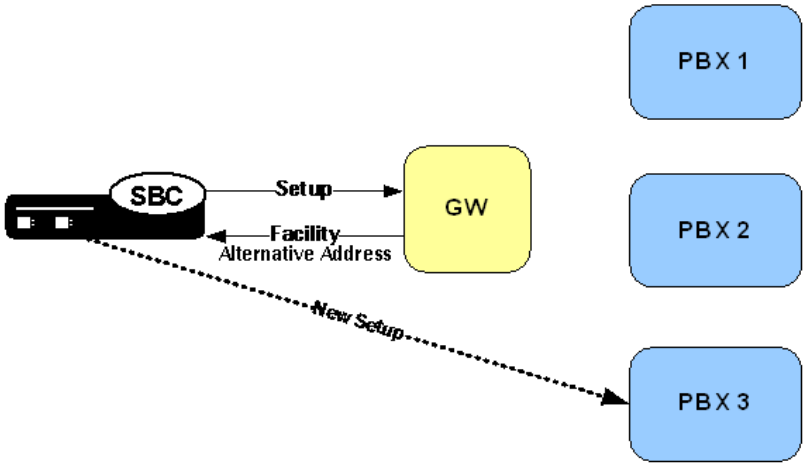
Prior to Release 4.1, the Oracle Communications Session Border Controller did not forward calls when the remote H.323 endpoint sent a Facility message with Call deflection as the reason and an alternate address for forwarding. Instead, it would either:

- Fail to release the initial call and initiate the forwarded call
- Drop the entire call when the remote endpoint for the call tore down the session

New Behavior

In the diagram below, you can see that the Oracle Communications Session Border Controller sends the initial Setup message to the gateway, and the gateway returns the Facility message with an alternate address for forwarding. Rather than engaging in its former behavior, the Oracle Communications Session Border Controller now releases the call with the gateway and sends a new Setup to the alternate address from the Facility message.

This new Setup up has no effect on the first call leg, which remains connected.

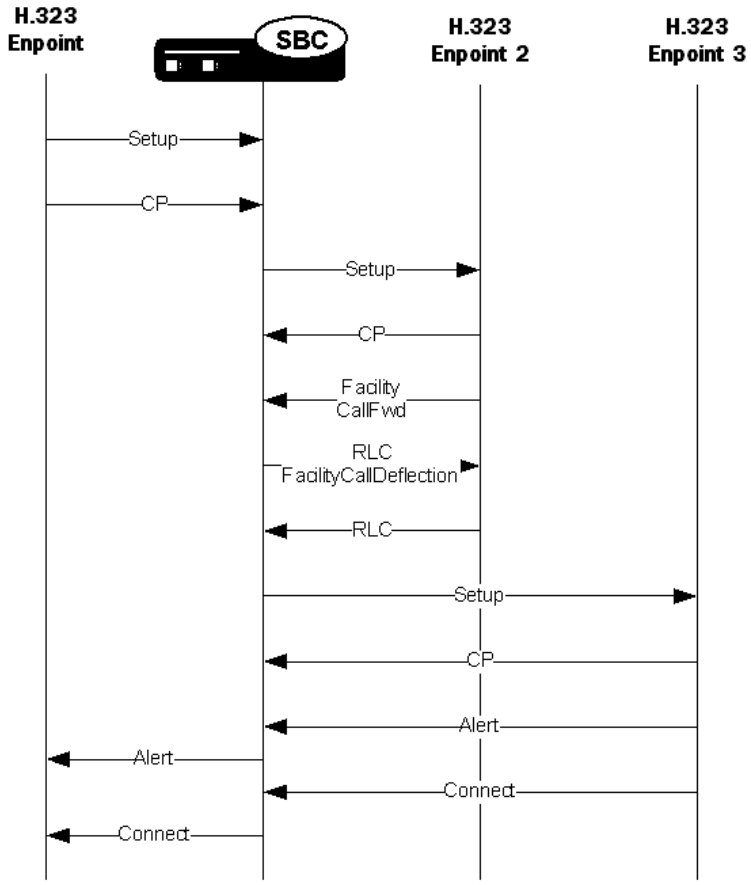


When it receives a Facility message with the reason CallForwarded, the Oracle Communications Session Border Controller looks for an alternate transport address in the Facility's alternativeAddress or alternativeAliasAddress element. The Oracle Communications Session Border Controller releases the egress call with the reason facilityCallDeflection. Then it takes one of two courses of action:

- If it does not find an alternative address, the Oracle Communications Session Border Controller releases the ingress call (with the reason facilityCallDeflection).
- If it finds an alternative address and the egress call has not been alerted or answered, the Oracle Communications Session Border Controller at this point tries to initiate a new egress call. The Oracle Communications Session Border Controller uses the alternative alias address to populate the calledPartyNumber information element (IE) and the destination address of the new Setup.

H.323 Sample Call Flow

The following diagram shows how the H.323 Call Forwarding feature works in a purely H.323 environment.



H.323 NOTIFY Support

To inform another call party of a certain event or communicate information to it, and H.323 entity might send a NOTIFY message. For example, a gateway might send a NOTIFY message to inform the calling party of a display name for a transferee. in previous releases, the Oracle Communications Session Border Controller did not process such a NOTIFY message, blocking the message from reaching its intended destination.

The Oracle Communications Session Border Controller supports the NOTIFY message so that it can pass through and reach its intended destination.

Caveats

The Oracle Communications Session Border Controller does not support interworking the NOTIFY message to a SIP message for calls that require interworking between H.323 and SIP; this support is for pure H.323 calls only.

H.323 H.239 Support for Video+Content

The Oracle Communications Session Border Controller supports multiple media streams for the same payload, generic capabilities, and H.239 generic messages. As a result, these additions broaden the Oracle Communications Session Border Controller's support for videoconferencing, and free you from have to configure media profiles for H.323 support.

> **Note:**
>
> These additions are supported for H.323-H.323 traffic only. These additions do not support SIP-H.323 interworking (IWF), so you still need to configure media profiles for that application.

Multiple Media Streams with the Same Payload

In releases prior to S-C6.2.0, the Oracle Communications Session Border Controller supports multiple audio-video-data streams only if those streams use different payload types. The Oracle Communications Session Border Controller's behavior is extended to provide this support as of Release S-C6.2.0. The Oracle Communications Session Border Controller identifies extendedVideoCapability used to establish an additional channel for H.239-compliant endpoints, an OLC that was formerly not supported.

Support for Generic Capabilities

This feature identifies the OIDs, shown in the table below, and uses the dynamicPayload type from the incoming OLC to generate its own OLC. You no longer need media profiles for genericAudio, genericVideo, and genericData.

Capability Name	Capability Class	Capability Identifier
H.283	Data protocol	{itu-t (0) recommendation (0) h (8) 283 generic-capabilities (1) 0}
G.722.1	Audio protocol	{itu-t (0) recommendation (0) g (7) 7221 generic-capabilities (1) 0}
G.722.1 Extension	Audio protocol	{itu-t (0) recommendation (0) g (7) 7221 generic-capabilities (1) extension (1) 0}
H.324	Data protocol	{itu-t (0) recommendation (0) h (8) 324 generic-capabilities (1) http (0)}
H.263	Video protocol	{itu-t (0) recommendation (0) h (8) 263 generic-capabilities (1) 0} Note: Use of this capability to signal H.263 "Profiles and Levels" per Annex X/ H.263 should always be accompanied in parallel by the signalling of the same modes in H263VideoCapability. This is necessary to ensure that systems which do not recognize the H.263 generic capabilities continue to interwork with newer systems.
H.224	Data protocol	{itu-t (0) recommendation (0) h (8) 224 generic-capabilities (1) 0}
G.722.2	Audio protocol	{itu-t (0) recommendation (0) g (7) 7222 generic-capabilities (1) 0}
G.726	Audio protocol	{itu-t (0) recommendation (0) g (7) 726 generic-capabilities (1) version2003 (0)}
H.241/H.264	Video protocol	{itu-t (0) recommendation (0) h (8) 241 specificVideoCodecCapabilities (0) h264 (0) generic-capabilities (1)}

Capability Name	Capability Class	Capability Identifier
H.241/H.264	Video protocol	{itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) iPpacketization(0) RFC3984NonInterleaved(1)}
H.241/H.264	Video protocol	{itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) iPpacketization(0) RFC3984Interleaved(2)}

Support for H.239 Generic Messages

This section describes the Oracle Communications Session Border Controller's support for H.239 Generic Messages.

Generic Message	Description
Generic Request	• flowControlReleaseRequest—Used when a device wants to add a channel toward an MCU that has sent multipointConference, or if the device wants to increase a channel bit rate when the channel is flow-controlled. The message has the channelId, which is the logicalChannelNumber of the channel. The Oracle Communications Session Border Controller proxies this message, replacing the channelId with the logicalChannelNumber of its channel. • presentationTokenRequest—Request by the sender to acquire the indicated token. The message has the channelId, which is the logicalChannelNumber of the channel. The Oracle Communications Session Border Controller proxies this message, replacing the channelId with the logicalChannelNumber of its channel.
Generic Response	• flowControlReleaseResponse—Sent in response to the flowControlReleaseRequest, either acknowledging or rejecting the request. The "acknowledge" response indicates the far-end device intends to make a best-effort attempt to comply with the request. The exact bit rate requested may not be allocated. The reject response indicates that the far-end device does not intend to comply with the request. The response contains the channelId that was sent in the request. While proxying the response, the Oracle Communications Session Border Controller will replace the channelId with the channelId it received in the request. • presentationTokenResponse—Sent in response to the presentationTokenRequest. The response will either confirm or reject the assignment of the indicated token to the sender of the presentationTokenRequest. The response contains the channelId that was received in the request. While proxying the response, the Oracle Communications Session Border Controller will replace the channelId with the channelId it received in the request.
Generic Command	• presentationTokenRelease—Sent by the device holding the token in order to relinquish the token. The message has the channelId, which is the logicalChannelNumber of the channel. The Oracle Communications Session Border Controller proxies this message, replacing the channelId with the logicalChannelNumber of its channel.

Generic Message	Description
Generic Indication	• presentationTokenIndicateOwner—Indicates who owns the token. The message has the channelId, which is the logicalChannelNumber of the channel. The Oracle Communications Session Border Controller proxies this message, replacing the channelId with the logicalChannelNumber of its channel.

Support for Miscellaneous Indication

An endpoint sends a miscellaneous indication to send (logicalChannelActive) or stop (logicalChannelInactive) live video streams. The message has a channelId, which is the channel's logicalChannelNumber. The Oracle Communications Session Border Controller proxies this message, replacing the channelId with the logicalChannelNumber of its own channel.

Video Conferencing Support for Polycom Terminals

The SBC includes support for Polycom video conferencing that implements messaging properly and presents proper addressing information for session billing, as described below.

The SBC supports operations in a video conferencing environment with Polycom H323 terminals and a Polycom MCU (Multipoint Conferencing Unit) by relaying H.239/H.245. The SBC implements the following messages appropriately:

- Miscellaneous command message with subtype such as multiPointModeCommand, cancelMultipointModeCommand
- Conference Indication message with subtype such as terminalNumberAssign, terminalYouAreSeeing

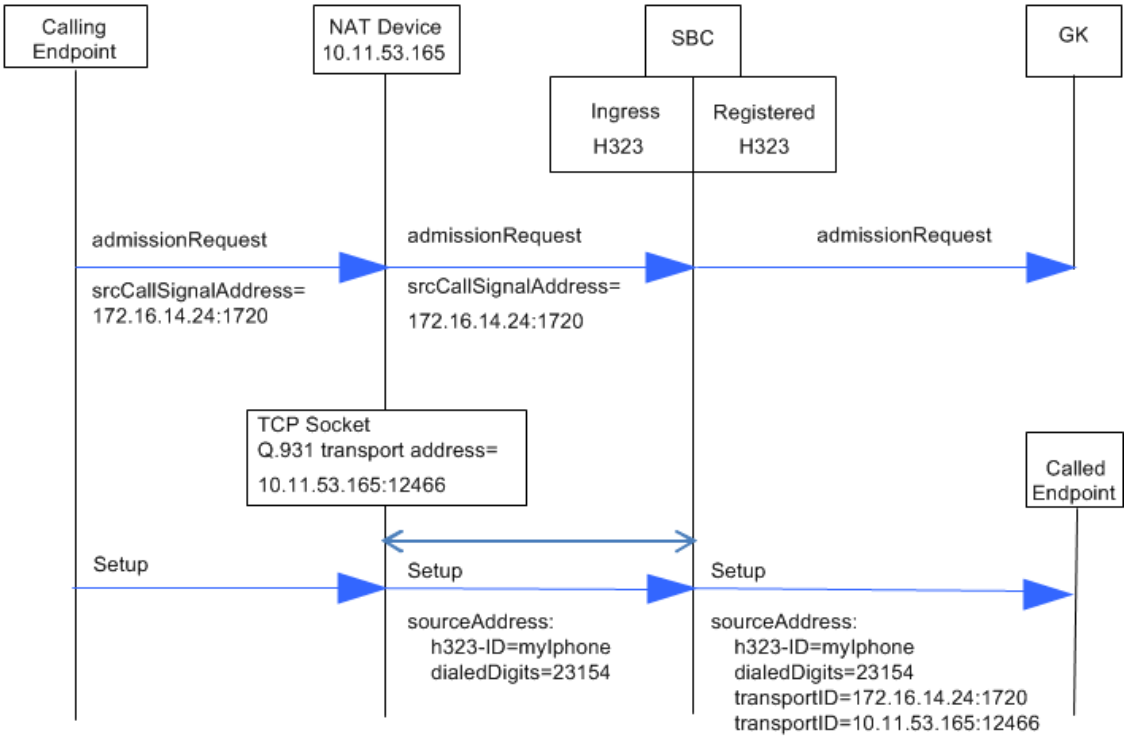
The SBC can also resolve a video conference billing issue caused by a NAT device. NAT devices change the sourceCallSignalAddress in a Setup message to the IP address of the NAT device. Deployments that rely on this address in CDRs to bill for the service need this address to be that of the original station behind the NAT. The user sets the **addSrcCallSignalAddr** option in the **h323-stack** that receives the incoming Setup to cause the SBC to present the desired address.

The srcCallSignalAddress field in the admissionRequest message received by the SBC often contains the transportAddress of the calling endpoint. This field is optional for the admissionRequest message. If configured with the option and presented with this field, the SBC saves the address for later use in outgoing Setup messages. If the srcCallSignalAddress field in the admissionRequest message is not present, or is equal to 0.0.0.0, the SBC does not save it.

Having stored this address, the SBC uses it following the steps below. Note that the Setup message in this scenario is preceded by the admission request and confirmation messages from the gatekeeper for the call.

1. Add the srcCallSignalAddress, saved from the admissionRequest message, into the sourceAddress field of the outgoing Setup message. The SBC adds this address in the format of an AliasAddress of type transportID after any other AliasAddresses that are normally included from the incoming Setup sourceAddress field.
2. Add the remote transport address of the incoming Q.931 TCP connection, into the sourceAddress field of the outgoing Setup message. The SBC adds this address in the format of an AliasAddress of type transportID, after the alias that was added in step 1.

The call flow diagram below depicts the SBC adding addresses in accordance with this feature.



Use the syntax below to set this option on the h323-stack receiving the incoming Setup.

```
(h323-stack)# options +addSrcCallSignalAddr
```

The user can find this process confirmed in the log.h232d log file.

ACLI Signaling Mode Configuration Examples

The following ACLI displays provide examples of the Signaling Modes of Operation described earlier in this chapter.

Configuration Fields and Values for B2BGW Signaling

This example provides is a sample for the Back-to-Back Gateway Signaling mode of operation.

```
h323-config
        state                           enabled
        log-level                       INFO
        response-tmo                    4
        connect-tmo                     32
h323-stack
        name                            zone1
        state                           enabled
        isgateway                       enabled
        realm-id                        zone1realm
        assoc-stack                     zone2
        local-ip                        x.x.x.x (IP address of VGW-A)
```

```
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=private
        ras-port                        1719
        auto-gk-discovery               enabled
        multicast                       224.0.1.41:1718
        gatekeeper                      x.x.x.x (IP address of GkZone1)
        gk-identifier                   gk-zone1.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone2
        state                           enabled
        isgateway                       enabled
        realm-id                        DomainCrealm
        assoc-stack                     zone1
        local-ip                        x.x.x.x(IP address of VGW-C)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=acme01
        ras-port                        1719
        auto-gk-discovery               enabled
        multicast                       224.0.1.41:1718
        gatekeeper                      x.x.x.x(IP address of GkZONE2)
        gk-identifier                   gk-zone2.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone3
        state                           enabled
        isgateway                       enabled
        realm-id                        zone3realm
```

```
        assoc-stack                     zone4
        local-ip                        x.x.x.x(IP address of VGW-B)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=private
        ras-port                        1719
        auto-gk-discovery               enabled
        multicast                       224.0.1.41:1718
        gatekeeper                      x.x.x.x(IP address of GkZone3)
        gk-identifier                   gk-zone3.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone4
        state                           enabled
        isgateway                       enabled
        realm-id                        DomainCrealm
        assoc-stack                     zone3
        local-ip                        x.x.x.x(IP address of VGW-D)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=private
        ras-port                        1719
        auto-gk-discovery               enabled
        multicast                       224.0.1.41:1718
        gatekeeper                      x.x.x.x(IP address of GkZone4)
        gk-identifier                   gk-zone4.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
```

Back-to-Back Gatekeeper Proxy and Gateway

This example provides is a sample for the Back-to-Back Gateway Proxy and Gateway mode of operation.

```
h323-config
        state                           enabled
        log-level                       INFO
        response-tmo                    4
        connect-tmo                     32
h323-stack
        name                            zone1
        state                           enabled
        isgateway                       disabled
        realm-id                        zone1realm
        assoc-stack                     zone2
        local-ip                        x.x.x.x(IP address of VGW-A/GK-A)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=private
        ras-port                        1719
        auto-gk-discovery               disabled
        multicast                       0.0.0.0:0
        gatekeeper                      x.x.x.x(IP address of GkZone1)
        gk-identifier                   gk-zone1.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone2
        state                           enabled
        isgateway                       disabled
        realm-id                        DomainCrealm
        assoc-stack                     zone1
        local-ip                        x.x.x.x(IP address of VGW-C/GK-C)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=acme01
        ras-port                        1719
        auto-gk-discovery               disabled
        multicast                       0.0.0.0:0
```

```
        gatekeeper                      x.x.x.x(IP address of GkZONE2)
        gk-identifier                   gk-zone2.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone3
        state                           enabled
        isgateway                       disabled
        realm-id                        zone3realm
        assoc-stack                     zone4
        local-ip                        x.x.x.x(IP address of VGW-B/GK-B)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=private
        ras-port                        1719
        auto-gk-discovery               disabled
        multicast                       0.0.0.0:0
        gatekeeper                      x.x.x.x(IP address of GkZone3)
        gk-identifier                   gk-zone3.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone4
        state                           enabled
        isgateway                       disabled
        realm-id                        DomainCrealm
        assoc-stack                     zone3
        local-ip                        x.x.x.x(IP address of VGW-D/GK-D)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
        h323-ID=private
        ras-port                        1719
```

```
        auto-gk-discovery           disabled
        multicast                   0.0.0.0:0
        gatekeeper                  x.x.x.x(IP address of GkZone4)
        gk-identifier               gk-zone4.example.com
        alternate-transport
        q931-port                   1720
        q931-max-calls              200
        h245-tunneling              enabled
        fs-in-first-msg             disabled
        call-start-fast             disabled
        call-start-slow             disabled
        media-profiles
        process-registration        disabled
        anonymous-connection        disabled
        proxy-mode
        filename
```

Interworking Gatekeeper-Gateway

This example provides is a sample for the Interworking Gatekeeper-Gateway mode of operation.

```
h323-config
        state                       enabled
        log-level                   INFO
        response-tmo                4
        connect-tmo                 32
h323-stack
        name                        zone1
        state                       enabled
        isgateway                   disabled
        realm-id                    zone1realm
        assoc-stack                 zone2
        local-ip                    x.x.x.x(IP address of VGW-A/GK-A)
        max-calls                   200
        max-channels                10
        registration-ttl            0
        terminal-alias
                                    h323-ID=private
        ras-port                    1719
        auto-gk-discovery           disabled
        multicast                   0.0.0.0:0
        gatekeeper                  x.x.x.x(IP address of GkZone1)
        gk-identifier               gk-zone1.example.com
        q931-port                   1720
        alternate-transport
        q931-max-calls              200
        h245-tunneling              enabled
        fs-in-first-msg             disabled
        call-start-fast             disabled
        call-start-slow             disabled
        media-profiles
        process-registration        disabled
        anonymous-connection        disabled
        proxy-mode
```

```
        filename
h323-stack
        name                            zone2
        state                           enabled
        isgateway                       enabled
        realm-id                        DomainCrealm
        assoc-stack                     zone1
        local-ip                        x.x.x.x(IP address of VGW-C)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=acme01
        ras-port                        1719
        auto-gk-discovery               enabled
        multicast                       0.0.0.0:0
        gatekeeper                      0.0.0.0:0
        gk-identifier                   gk-zone2.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone3
        state                           enabled
        isgateway                       disabled
        realm-id                        zone3realm
        assoc-stack                     zone4
        local-ip                        x.x.x.x(IP address of VGW-B/GK-B)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
        h323-ID=private
        ras-port                        1719
        auto-gk-discovery               disabled
        multicast                       0.0.0.0:0
        gatekeeper                      x.x.x.x(IP address of GkZone3)
        gk-identifier                   gk-zone3.example.com
        q931-port                       1720
        alternate-transport
        q931-max-calls                  200
        h245-tunneling                  enabled
        fs-in-first-msg                 disabled
        call-start-fast                 disabled
        call-start-slow                 disabled
        media-profiles
        process-registration            disabled
```

```
        anonymous-connection            disabled
        proxy-mode
        filename
h323-stack
        name                            zone4
        state                           enabled
        isgateway                       enabled
        realm-id                        DomainCrealm
        assoc-stack                     zone3
        local-ip                        x.x.x.x(IP address of VGW-D)
        max-calls                       200
        max-channels                    10
        registration-ttl                0
        terminal-alias
                                        h323-ID=private
        ras-port                        1719
        auto-gk-discovery               disabled
        multicast                       0.0.0.0:0
        gatekeeper                      x.x.x.x(IP address of GkZone4)
        gk-identifier                   gk-zone4.example.com
```

Additional Information

This section contains detailed tables to use as a reference when you are learning about H.323 features or when you are configuring them.

About Payload Types

You set the payload type when you are configuring a media profile to support Slow Start to Fast Start Translation.

When you configure media profiles, you might need set the payload type to identify the format in the SDP m lines. For RTP/AVP, the default transport method of a media profile configuration, this will be the RTP payload type number. Newer codecs have dynamic payload types, which means that they do not have an assigned payload type number.

When you use RTP/AVP as the transport method, you should only set the payload type when there is a standard payload type number for the encoding name; otherwise, leave the payload type blank.

The Oracle Communications Session Border Controller uses the payload type value to determine the encoding type when SDP identifies the standard payload type in the m line, but does not include an a=rtpmap entry. These are two equivalent SDPs:

c=IN IP4 192.0.2.4

```
m=audio 0 RTP/AVP 0
```

c=IN IP4 192.0.2.4

```
m=audio 0 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

The first does not include the RTP map entry, but uses the standard payload type of 0. If the Oracle Communications Session Border Controller receives an SDP like the first, it uses the payload type 0 to locate the corresponding media profiles configuration. When an a=rtpmap is present, the Oracle Communications Session Border Controller uses the encoding name in the a=rtpmap line to find the media profile configuration and does not consider the payload type number.

Payload Types for Standard Audio and Visual Encodings

The following is a table of standard audio and visual payload encodings defined in H. Schulzrinne, GND Fokus, RTP Profile for Audio and Visual Conferences with Minimal Control, RFC 1890, and in the *RTP Parameters* document in IANA's Directory of Generally Assigned Numbers.

Payload Type	Encoding Name	Audio (A)/Visual (V)	Clock Rate (Hz)
0	PCMU	A	8000
1	1016	A	8000
2	G721	A	8000
3	GSM	A	8000
4	G723	A	8000
5	DVI4	A	8000
6	DVI4	A	16000
7	LPC	A	8000
8	PCMA	A	8000
9	G722	A	8000
10	L16	A	44100
11	L16	A	44100
12	QCELP	A	8000
13	reserved	A	N/A
14	MPA	A	90000
15	G728	A	8000
16	DVI4	A	11025
17	DVI4	A	22050
18	G729	A	8000
19	reserved	A	N/A
20	unassigned	A	N/A
21	unassigned	A	N/A
22	unassigned	A	N/A
23	unassigned	A	N/A
dyn	GSM-HR	A	8000
dyn	GSM-EFR	A	8000
dyn	L8	A	var.
dyn	RED	A	N/A
dyn	VDVI	A	var.
24	unassigned	V	N/A
25	CelB	V	90000
26	JPEG	V	90000
27	unassigned	V	N/A
28	nv	V	90000
29	unassigned	V	N/A

Payload Type	Encoding Name	Audio (A)/Visual (V)	Clock Rate (Hz)
30	unassigned	V	N/A
31	H261	V	90000
32	MPV	V	90000
33	MP2T	AV	90000
34	H263	V	90000
35-71	unassigned	?	N/A
72-76	reserved for RTCP conflict avoidance	N/A	N/A
77-95	unassigned	?	N/A
96-127	dynamic	?	N/A
dyn	BT656	V	90000
dyn	H263-1998	V	90000
dyn	MP1S	V	90000
dyn	MP2P	V	90000
dyn	BMPEG	V	90000

About RAS Message Treatment

When you enabled the H.323 Registration Proxy, the Oracle Communications Session Border Controller modifies and deletes certain fields as outlined in the table below. The Oracle Communications Session Border Controller forwards any fields that are not listed in this table without modifying or deleting them.

> **Note:**
>
> Although the Oracle Communications Session Border Controller forwards a field, it does not always support the feature related to that field.

Field Name	Message	Deleted	Modified	Value Used in Modification
alternateEndpoints	RRQ, URQ, ACF	X	N/A	N/A
alternateGatekeeper	RCF, URQ	X	N/A	N/A
altGKInfo	RRJ, URJ, DRJ	X	N/A	N/A
alternateTransportAddresses	RRQ, ARQ, ACF	X	N/A	N/A
callModel	ARQ	N/A	X	direct
N/A	ACF	N/A	X	gatekeeperRouted
callSignalAddress	RRQ	N/A	X	Mapped virtual CSA allocated by the system for registering the endpoint.
N/A	RCF, ARJ	N/A	X	CSA of gatekeeper stack

Field Name	Message	Deleted	Modified	Value Used in Modification
N/A	URQ	N/A	X	If URQ is from an endpoint, endpoint's mapped virtual CSA. If URQ is from a gatekeeper, real CSA of endpoint.
destCallSignalAddress	ARQ, ACF	X	N/A	N/A
destinationInfo.transportID	ARQ, ACF	X	N/A	N/A
destExtraCallInfo.trasnportID	ARQ, ACF	X	N/A	N/A
discoveryComplete	RRQ	N/A	X	TRUE
endpointAlias.trasnportID	URQ	X	N/A	N/A
endpointAliasPattern.Wwildcard.transportID	URQ	N/A	N/A	N/A
featureServerAlias.trasnportID	RCF	X	N/A	N/A
gatekeeperIdentifier	RRQ	N/A	X	Gatekeeper identifier of the gateway stack, either configured in the H.323 gateway stack or discovered dynamically.
maintainConnection	RRQ, RCF	N/A	X	FALSE
mutlipleCall	RRQ, RCF		X	FALSE
preGrantedARQ.alternateTransportAddresses	RCF	X	N/A	N/A
preGrantedARQ.useSpecifiedTransport	RCF	X	N/A	N/A
rasAddress	RRQ	N/A	X	Mapped virtual RAS address allocated by the system for registering endpoint
remoteExtentsionAddress.trasnportID	ARQ, ACF	X	N/A	N/A
srcCallSignalAddress	ARQ	X	N/A	N/A
srcInfo.transportID	ARQ	X	N/A	N/A
supportedH248Packages	RRQ	X	N/A	N/A
supportsAltGK	RRQ	X	N/A	N/A
supportedPrefixes.prefic.transportID	RCF, URQ	X	N/A	N/A
terminalAlias.transportID	RRQ	X	N/A	N/A
terminalAliasPattern.wilcard.transportID	RRQ	X	N/A	N/A
willRespondToIIRR	RCF, ACF	X	N/A	N/A
willSupplyUUIEs	RRQ, ARQ		N/A	N/A

Field Name	Message	Deleted	Modified	Value Used in Modification
uuiesRequested	ACF	N/A	X	FALSE
setup			X	FALSE
callProceeding			X	FALSE
connect			X	FALSE
alerting			X	FALSE
information			X	FALSE
releaseComplete			X	FALSE
facility			X	FALSE
progress			X	FALSE
empty			X	FALSE
...,			X	FALSE
status			X	FALSE
statusInquiry			X	FALSE
setupAcknowledge				
notify				

7

IWF Services

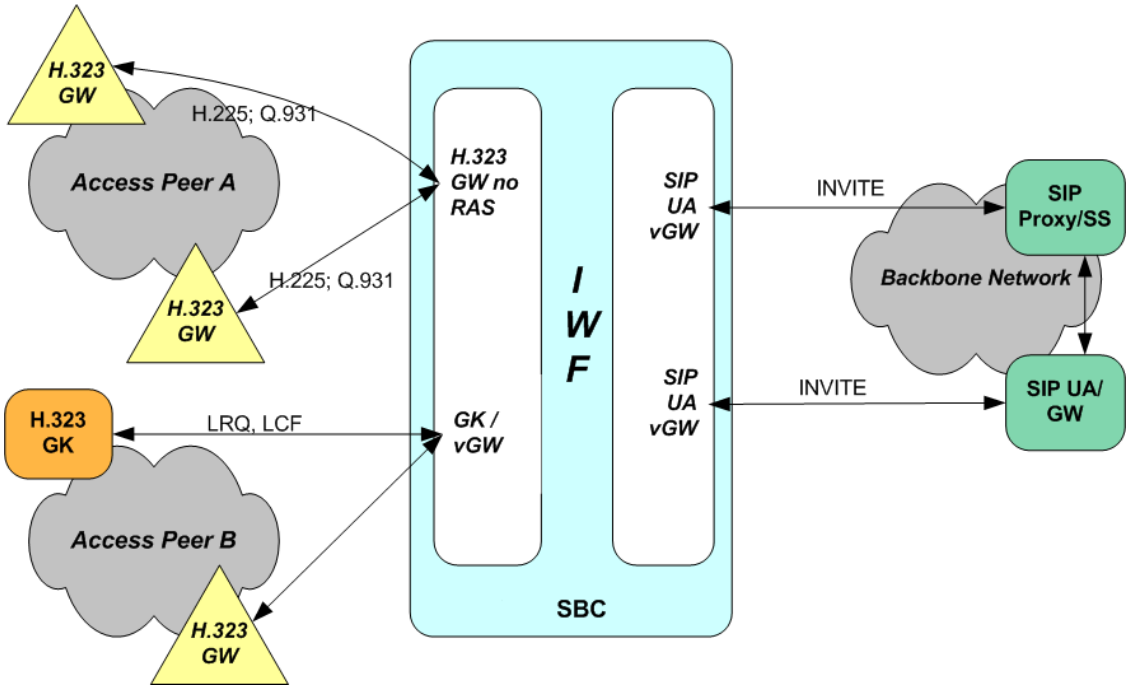
Using the Oracle Communications Session Border Controller's interworking function (IWF), you can interconnect SIP networks with H.323 networks. Considering the large amount of H.323 deployments already in place and the continuing emergence of SIP in new VoIP deployments, the IWF provides a much-needed solution. SIP providers can maintain a single-protocol backbone while exchanging VoIP sessions with H.323 providers.

The H.323 Signaling Services section contains information about the H.323 signaling modes of operation that the Oracle Communications Session Border Controller supports. The following H.323 signaling modes of operation can be used when you use the Oracle Communications Session Border Controller's IWF in an access or a peering solution.

- Back-to-back gateway signaling
- Interworking gatekeeper/gateway

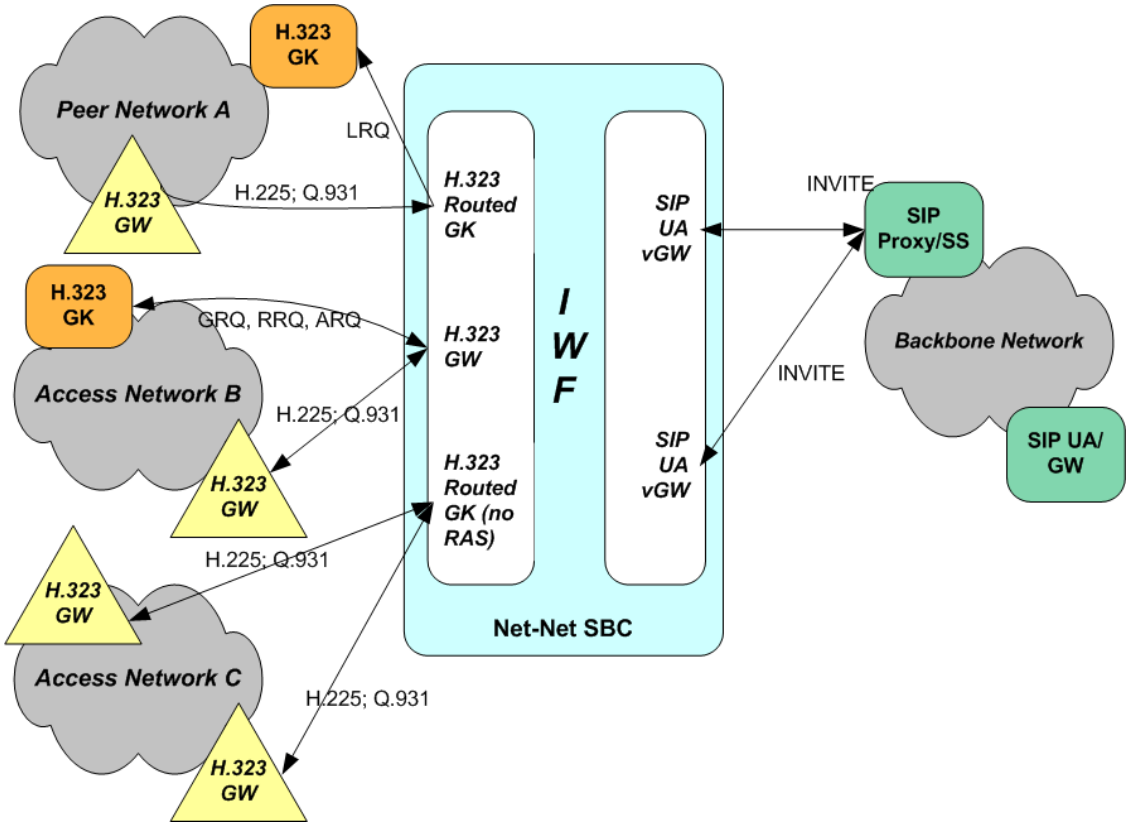
Access Network Application

You can configure your Oracle Communications Session Border Controller so that it provides an access solution for your network. The access solution allows SIP-based hosted communications platforms to be extended to enterprise-based H.323 systems. In the figure below, you can see different types of H.323 signaling modes being interworked with SIP. On the H.323 side, the SBC can appear to be a gatekeeper or a gateway, depending on how you configure the H.323 interface. On the SIP side, the SBC can appear to be a SIP UA or behave as a virtual gateway.



Networking Peering Application

In the IWF network peering solution, you can see the same network elements at work. However, the H.323 side of this IWF application shows the use of a gatekeeper controlled gateway for Peer Network B. Because this is a peering solution, the SIP side of the SBC communicates with the SIP proxy or softswitch in the backbone network rather than with the SIP UA or SIP gateway.



SIP and H.323

The Oracle Communications Session Border Controller supports interworking between SIP and H.323 for H.323 Slow Start and Fast Start calls. In addition to describing IWF sessions when initiated from the H.323 side and from the SIP side (with sample call flows), this section provides information you will need when you configure SIP and H.323.

SIP H.323 Negotiation H.323 Fast Start

The Oracle Communications Session Border Controller can perform protocol translations for SIP and H.323 Fast Start, where media capabilities are sent with the Setup request for an H.323 session.

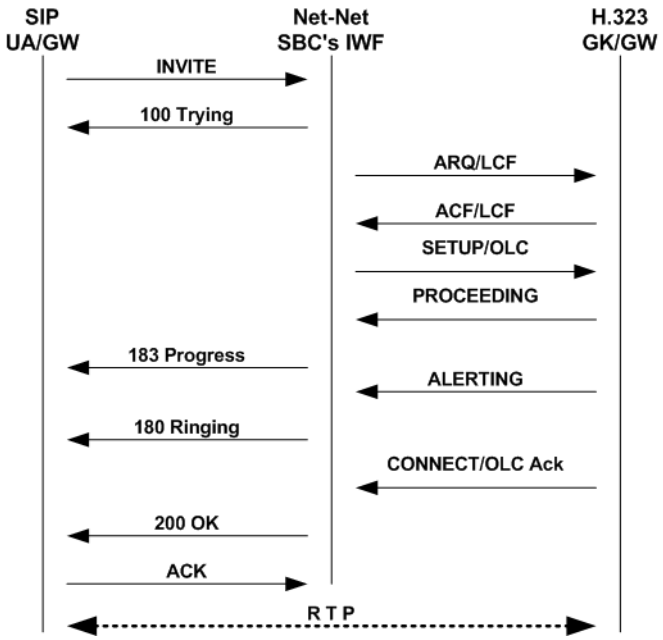
This section's call flow diagrams show how SIP and H.323 messages flow between SIP and H.323 devices, with the Oracle Communications Session Border Controller positioned between the two entities so it can perform translations. The following two sample scenarios with Fast Start appear in the diagrams below, although other scenarios are possible:

- Calls originating in SIP being translated to H.323 Fast Start
- Calls originating in H.323 Fast Start translated to SIP

SIP to Fast Start H.323

In the following diagram below, a SIP endpoint (such as a UA or a SIP Gateway) initiates a session by sending an INVITE message destined for an H.323 endpoint (a GK or GW). Between these entities, the system is positioned to perform interworking. The Oracle Communications Session Border Controller recognizes that the INVITE message is destined for an H.323 device, and returns a 100 Trying message to the SIP endpoint as it attempts to negotiate the H.323 side of the session. This negotiation starts when the Oracle Communications Session Border Controller initiates the RAS process with the H.323 endpoint by sending either an ARQ or an LRQ, allowing the Oracle Communications Session Border Controller to determine if the H.323 endpoint will accept the session.

Once the H.323 endpoint responds with an ACF or LCF, the Oracle Communications Session Border Controller reissues the SIP INVITE on the H.323 side as an H.225 Setup, which is sent with the OLC. Then the H.323 endpoint responds with Proceeding and Alerting messages (which correspond respectively to SIP 183 Progress and 180 Ringing messages). At that point, the H.323 endpoint sends a Connect message that includes the OpenLogicalChannel message (OLC), announcing the logical channel for media flows has been set up. The Oracle Communications Session Border Controller converts the H.323 OLC to a SIP 200 OK. After receiving the 200 OK, the SIP endpoint sends an ACK, confirming that the session has been established. Because there is no H.323 equivalent for the SIP ACK, the Oracle Communications Session Border Controller does not generate a corresponding message on the H.323 side. At this point, the session is fully established and RTP flows between the endpoints.

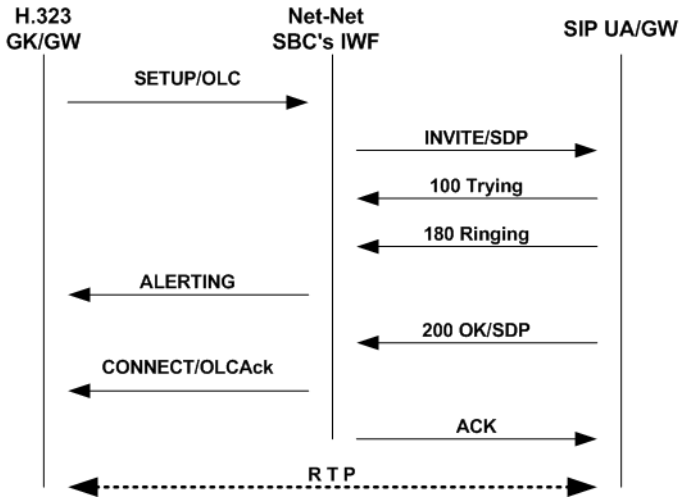


H.323 Fast Start to SIP

In the diagram below, an H.323 endpoint (a GK or GW) initiates a session by sending a Setup request destined for a SIP endpoint (such as a UA or a SIP Gateway). Between these entities,

the Oracle Communications Session Border Controller is positioned to perform interworking. The H.323 endpoint has completed the RAS process prior to sending the SETUP message.

The Oracle Communications Session Border Controller receives the Setup message and then sends a SIP INVITE on the SIP side. The SIP endpoint responds with a 100 Trying; the Oracle Communications Session Border Controller does not resend this message on the H.323 side. Next, the SIP endpoint issues a 180 Ringing message, which the Oracle Communications Session Border Controller reissues to the H.323 endpoint as an Alerting message. The SIP endpoint then sends a 200 OK, retransmitted by the Oracle Communications Session Border Controller as a Connect message that includes an OLC. Once the Oracle Communications Session Border Controller sends an ACK to the SIP endpoint, RTP flows between the endpoints.



SIP H.323 Negotiation H.323 Slow Start

The Oracle Communications Session Border Controller can also perform protocol translations for SIP and H.323 Slow Start, where—unlike the cases with Fast Start described above—media information is not sent with the Setup request for an H.323 session. For H.323 Slow Start, media is negotiated after the session is established.

This section's call flow diagrams show how SIP and H.323 messages flow between SIP UA/GW and an H.323 GK/GW, with the Oracle Communications Session Border Controller positioned between the two entities so it can perform translations. Two sample scenarios with Slow Start appear in the diagrams below:

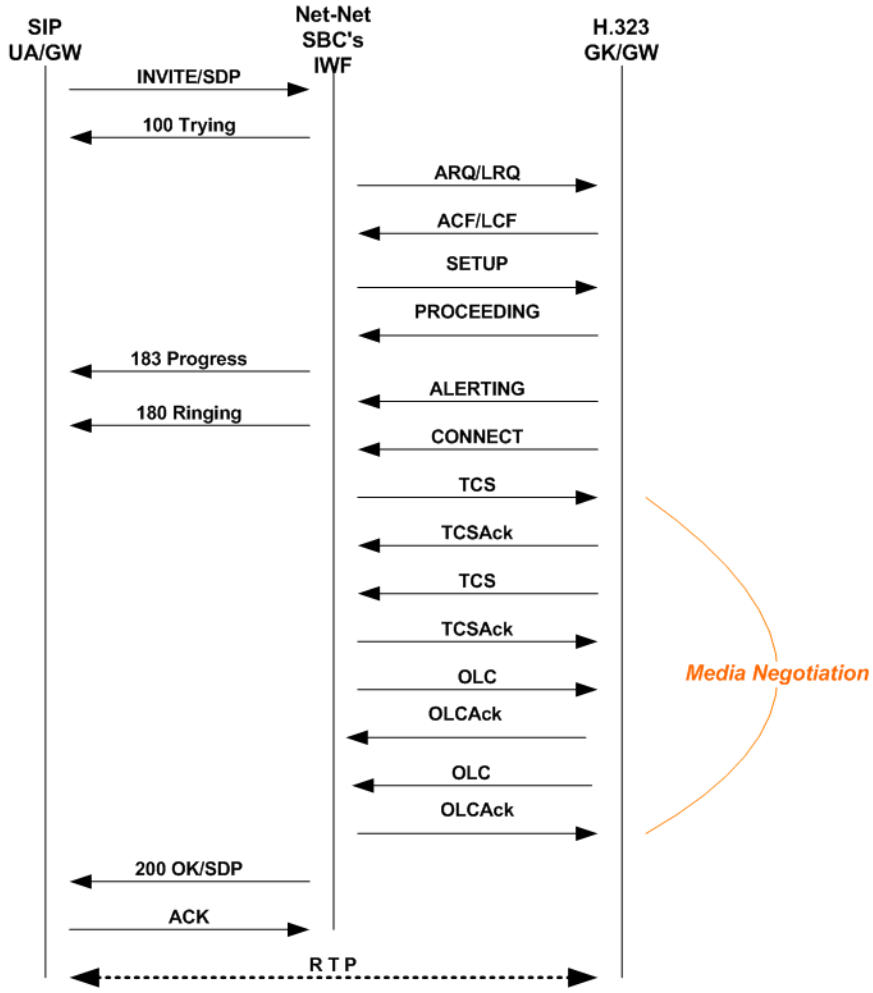
- SIP being interworked to Slow Start H.323
- Slow Start H.323 being interworked to SIP

H.323 SIP to Slow Start

In the following diagram below, a SIP endpoint (such as a UA or a SIP Gateway) initiates a session by sending an INVITE request destined for an H.323 Slow Start endpoint (a GK or GW). Between these entities, the Oracle Communications Session Border Controller is positioned to perform interworking.

The call flow for this type of translation works fundamentally the same way that the translation does for SIP to Fast Start H.323, with the exception of how the media is established. Media is negotiated through the exchange of TCS and OLC messages after the H.323 Connect and SIP

180 Ringing messages have been sent. The first TCS message is sent from the Oracle Communications Session Border Controller to the H.323 endpoint, and it contains information about media capabilities in SDP. The H.323 endpoint accepts and acknowledges this information with a TCS Ack message. Then the H.323 endpoint sends a second TCS, carrying information about the Gateway's capabilities, that the Oracle Communications Session Border Controller accepts and acknowledges. The H.323 endpoint and the Oracle Communications Session Border Controller then exchange OLC and OLC Ack messages that establish the operating mode and Gateway capability. Finally, the Oracle Communications Session Border Controller completes the 200 OK/ACK sequence on the SIP side, and RTP flows between the two endpoints.



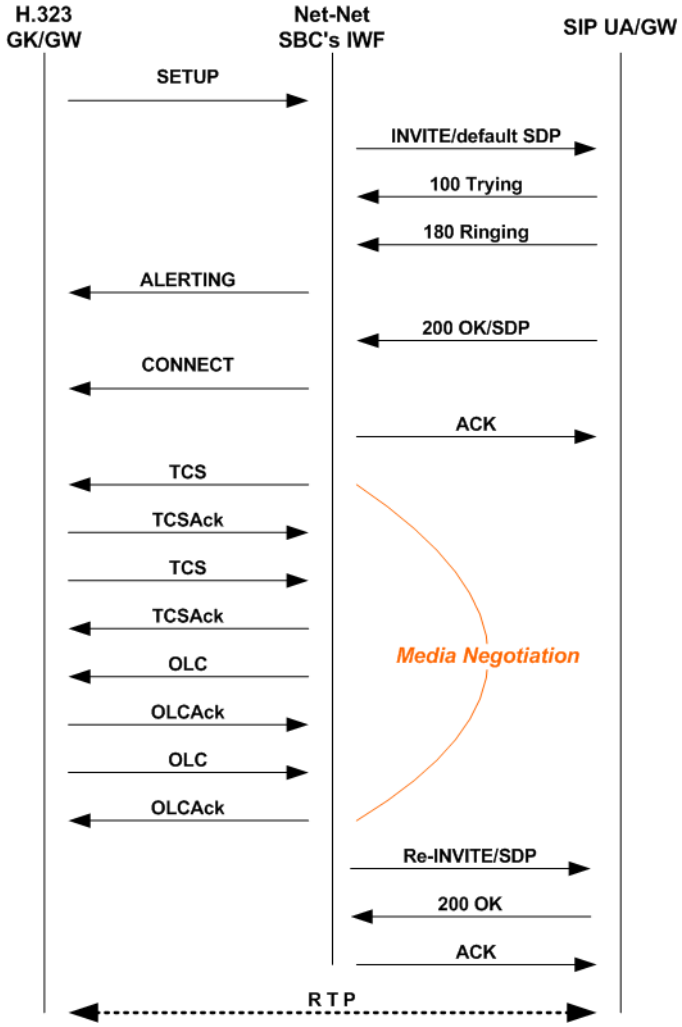
H.323 Slow Start to SIP

In the following diagram below, an H.323 endpoint (GW or GK) initiates a session by sending a Setup request destined for a SIP endpoint (such as a UA or a SIP Gateway). Between these entities, the Oracle Communications Session Border Controller is positioned to perform interworking. The H.323 endpoint has completed the RAS process prior to sending the SETUP message.

The call flow for this type of translation works fundamentally the same way that the translation does for H.323 Fast Start to SIP, with the exception of how the media is established. When the Oracle Communications Session Border Controller receives an H.323 message destined for a SIP endpoint, it sends a SIP INVITE message that includes default SDP to that SIP endpoint. The default SDP is constructed using information in the media profiles listed for the IWF

configuration; if necessary, this media information is amended later in the sequence. Once the call is set up, the Oracle Communications Session Border Controller negotiates media with the H.323 endpoint through a series of TCS/TCS Ack and OLC/OLC Ack messages that establish the operating mode and Gateway capability.

When the Oracle Communications Session Border Controller completes media negotiation with the H.323 endpoint, it issues a re-INVITE to the SIP endpoint that contains the updated information needed for media transmission. In response, the SIP endpoint sends a 200 OK message that the Oracle Communications Session Border Controller answers with an ACK. Then RTP can flow between the two endpoints.



Status and Codec Mapping

The Oracle Communications Session Border Controller maps SIP and H.323 status codes as described in this section. Status and codec mapping do not require configuration; they occur transparently.

IWF Termination from H.323

When a call that requires the IWF terminates from the H.323 side, the Oracle Communications Session Border Controller uses the mapping scheme in the following table to determine the appropriate SIP status.

H.323 Disconnect Reason	SIP Status
No Bandwidth	480 Temporarily Unavailable
Gatekeeper Resource	404 Not Found
Unreachable Destination	404 Not Found
Destination Rejection	603 Decline
Invalid Revision	505 Version Not Supported
No Permission	401 Unauthorized
Unreachable Gatekeeper	503 Service Unavailable
Gateway Resource	480 Temporarily Unavailable
Bad Format Request	400 Bad Request
Adaptive Busy	486 Busy Here
In Conference	486 Busy Here
Undefined Reason	500 Internal Server Error
Facility Call Deflection	486 Busy Here
Security Denied	401 Unauthorized
Called Party Not Registered	404 Not Found
Caller Not Registered	401 Unauthorized

IWF Termination During H.323 RAS

When a call that requires the IWF terminates from the H.323 side during RAS and generates an error, the Oracle Communications Session Border Controller uses the mapping scheme in the following table to determine the appropriate SIP status.

H.323 RAS Error	SIP Status
Called Party Not Registered	404 Not Found
Invalid Permission	401 Unauthorized
Request Denied	503 Service Unavailable
Undefined	500 Internal Server Error
Caller Not Registered	401 Unauthorized
Route Call To Gatekeeper	305 User Proxy
Invalid Endpoint ID	500 Internal Server Error
Resource Unavailable	503 Service Unavailable
Security Denial	401 Unauthorized
QoS Control Not Supported	501 Not Implemented
Incomplete Address	484 Address Incomplete
Route Call to SCN	302 Moved Temporarily
Aliases Inconsistent	485 Ambiguous
Not Currently Registered	401 Unauthorized

IWF RAS Registration Failure Code Mapping

For calls that require interworking between H.323 and SIP, the Oracle Communications Session Border Controller supports IWF response code mapping. This feature enables the Oracle Communications Session Border Controller to support configurable SIP response codes for IWF calls that fail during RAS, when the Oracle Communications Session Border Controller has been unable to register with a gatekeeper; this allows a wider range of more accurate response codes to be communicated.

When this feature is not enabled, the Oracle Communications Session Border Controller generates a 404 Not Found when a SIP-to-H.323 call fails as a result of the stack's failure to register with a gatekeeper.

When the condition noted above takes place, the response code can be any of the ones listed in this table. The code values listed in the table are used to specify the code to which you want to map.

Code	Description
403	Forbidden
406	Not Acceptable
408	Request Timeout
410	Gone
420	Bad Extension
480	Temporarily Unavailable
486	Busy Here
487	Request Terminated
500	Server Internal Error
503	Service Unavailable
504	Server Time-out
600	Busy Everywhere
603	Decline

To enable IWF RAS registration failure code mapping:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ORACLE(h323)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**iwfRegFailCode=X**), and then press Enter. X is the SIP response code that you want to use; the table above lists the supported response codes that are supported.

   ```
   ORACLE(h323)# options +iwfRegFailCode=503
   ```

   If you type **options iwfRegFailCode=X**, you will overwrite any previously configured options. In order to append the option to the options list, you must prepend the new option with a plus sign as shown in the previous example.

IWF Termination from SIP

When a call that requires the IWF terminates from the SIP side, the Oracle Communications Session Border Controller uses the mapping scheme in the following table to determine the appropriate H.323 Release Complete Reason code.

SIP Status	H.323 Release Complete Reason
300 Multiple Choices	Undefined Reason
401 Unauthorized	Security Denied
402 Payment Required	Undefined Reason
403 Forbidden	No Permission
404 Not Found	Unreachable Destination
405 Method Not Allowed	Undefined Reason
406 Not Acceptable	Undefined Reason
407 Proxy Authentication Required	Security Denied
408 Request Timeout	Adaptive Busy
409 Conflict	Undefined Reason
410 Gone	Unreachable Destination
411 Length Required	Undefined Reason
414 Request-URI Too Large	Bad Format Address
415 Unsupported Media Type	Undefined Reason
420 Bad Extension	Bad Format Address
480 Temporarily Unavailable	Adaptive Busy
481 Call/Transaction Does Not Exist	Undefined Reason
482 Loop Detected	Undefined Reason
483 Too Many Hops	Undefined Reason
484 Address Incomplete	Bad Format Address
485 Ambiguous	Undefined Reason
486 Busy Here	In Conference
487 Request Terminated	Undefined Reason
488 Not Acceptable Here	Undefined Reason
500 Internal Server Error	Undefined Reason
501 Not Implemented	Undefined Reason
502 Bad Gateway	Gateway Resource
503 Service Unavailable	Gateway Resource
504 Gateway Timeout	Adaptive Busy
505 Version Not Supported	Invalid Revision
600 Busy Everywhere	Adaptive Busy
603 Decline	Destination Rejection
604 Does Not Exist Anywhere	Unreachable Destination
606 Not Acceptable	Undefined Reason

Q.850 Cause to H.323 Release Complete Reason

When a call that requires the IWF terminates from the H.323 side and no H.323 Release Complete Reason is specified, the Oracle Communications Session Border Controller maps the Q.850 cause to an H.323 Release Complete Reason using the mapping scheme in the following table. This new H.323 status is then mapped to a SIP status as described in the IWF Termination from SIP table.

Q.850 Cause	H.323 Release Complete Reason
No Route To Destination	Unreachable Destination
Normal Call Clearing	Destination Rejection
User Busy	In Conference
Subscriber Absent	Called Party Not Registered
Invalid Number Format	Bad Format Address
Normal Unspecified	Undefined Reason
No Circuit/Channel Available	No Bandwidth
Network Out Of Order	Unreachable Gatekeeper
Temporary Failure	Adaptive Busy
Switching Equipment Congestion	Gateway Resource
Resource Unavailable	Gatekeeper Resource
Incompatible Destination	Invalid Revision
Interworking Unspecified	No Permission

Codec Mapping

The Oracle Communications Session Border Controller uses the following mapping scheme when converting media specifications between H.245 (used in H.323) and SDP (used in SIP).

Media coming into the Oracle Communications Session Border Controller one way exits the system in the corresponding way as specified in the following table. For example, media coming into the Oracle Communications Session Border Controller as H.245 type g711Ulaw64k exits the system as media type PCMU.

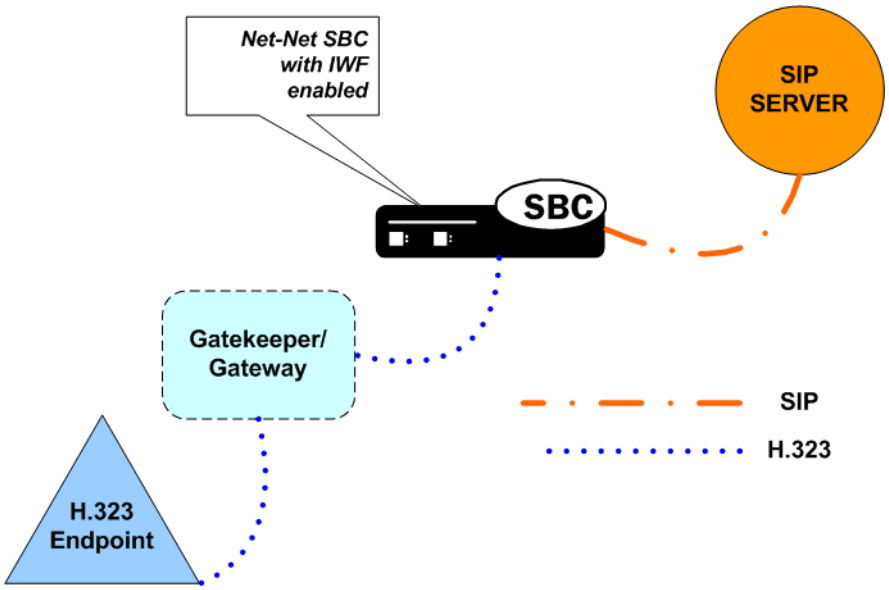
H.245 Type	SDP Media Type
g711Ulaw64k	PCMU
g711Ulaw56k	PCMU
g711Alaw64k	PCMA
g711Alaw56k	PCMA
g726	G726-32
g7231	G723
g722	G722
g728	G728
g729wAnnexB	G729
g729	G729 fmtp:18 annexb=no
h261VideoCapability	H261
h263VideoCapability	H263

IWF Service Enhancements

This section describes the Oracle Communications Session Border Controller features that are supported for when the Oracle Communications Session Border Controller performs interworking between SIP and H.323. Enabling these enhancements only requires that you set up a fully functional SIP configuration, a fully functional H.323 configuration, and that you enable IWF on your Oracle Communications Session Border Controller . You do not have to set any special configuration because these enhancements happen automatically.

SIP Redirect—H.323 LRQ Management

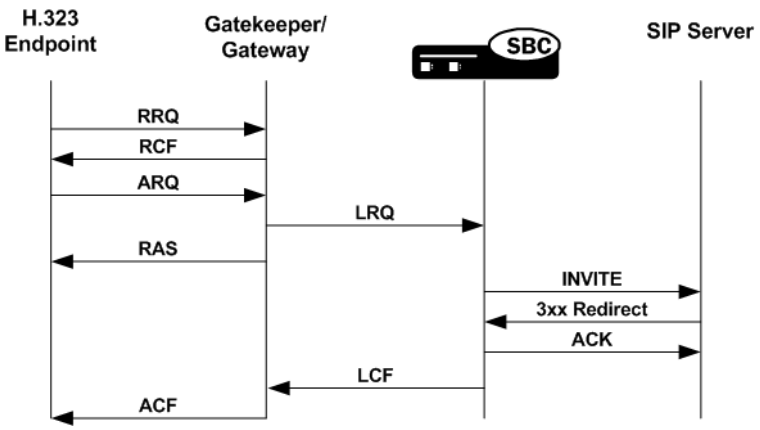
When it needs to interact with a SIP Redirect server, the SBC can interpret the SIP messages and manage them on the H.323 side of the session. For IWF sessions, the SBC handles SIP Redirect and H.323 LRQ messages.



Redirect—LRQ Management Sample 1

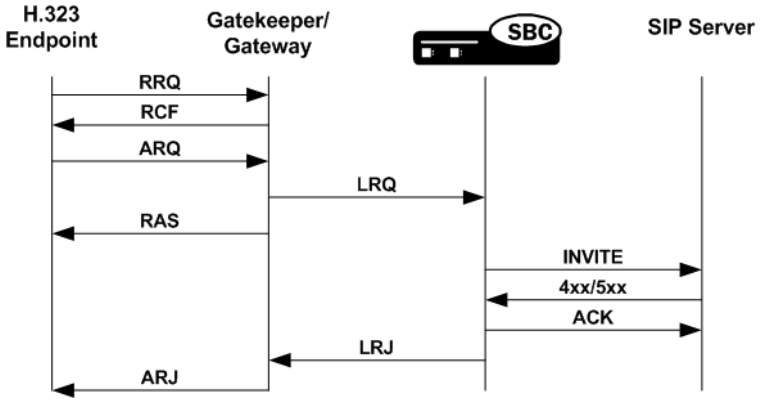
This section presents three possible scenarios for SIP Redirect-H.323 LRQ management.

The following diagram shows an established session that uses SIP Redirect—H.323 LRQ management. Here, the Oracle Communications Session Border Controller sends an INVITE to a SIP Redirect Server that responds with a 3xx Redirection message. The Oracle Communications Session Border Controller then sends the gatekeeper/gateway an LCF message that causes an ACF message to be sent to the H.323 endpoint.



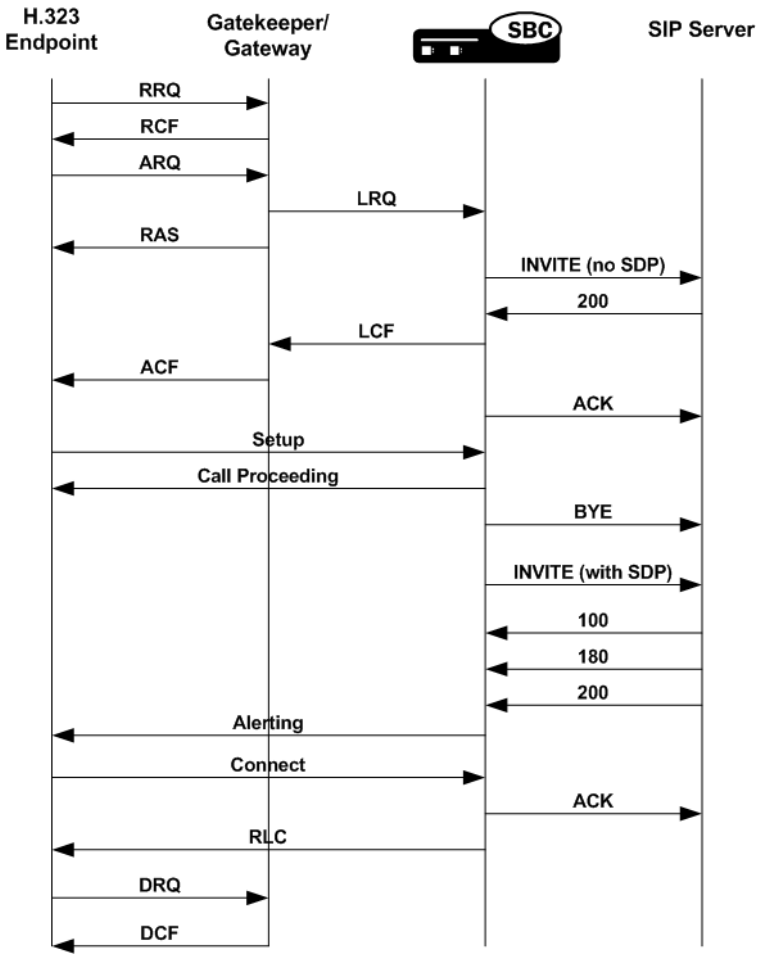
Redirect—LRQ Management Sample 2

The following diagram shows how the Oracle Communications Session Border Controller handles the exchange when the SIP Redirect server declares either that there is an error or that there is no such user. These SIP messages come from either the 4xx Request Failure or 5xx Server Failure series. In the example below, the SIP Redirect server returns a 401 Unauthorized message, which the Oracle Communications Session Border Controller interworks and communicates to the H.323 gatekeeper as an LRJ. Then the H.323 gatekeeper/ gateway issues an ARJ to the H.323 endpoint.



Redirect—LRQ Management Sample 3

In this call flow, the SIP server issues a 2xx Successful message that is not supposed to be sent (because a 3xx, 4xx, or 5xx message should be sent in response to the Oracle Communications Session Border Controller 's INVITE). The Oracle Communications Session Border Controller sends a BYE message to the SIP Redirect Server, but it tries to initiate the session again, this time successfully. The final sample call flow shown rarely occurs.



SIP INFO and DTMF UII Management

The Oracle Communications Session Border Controller supports DTMF for that require the IWF, enabling features such as keypress, alphanumeric, and hookflash. Because tones are not transmitted as audio, they must pass as out-of-band signaling information, meaning that the Oracle Communications Session Border Controller needs to convert an H.245 UII (User Input Indication) into SIP.

Depending on the capability of the H.323 endpoint, the Oracle Communications Session Border Controller sends either an alphanumeric or DTMF signal in the H.245 UII. The Oracle Communications Session Border Controller sends nothing if the endpoint does not support an alphanumeric or DTMF signal. The SIP INFO message will have a content type of application/ dtmf-relay, and the message body will be in the form Signal=*\r\nDuration=250\r\n. If the duration is absent in the SIP INFO or the UII received on the H.323 side is alphanumeric, the Oracle Communications Session Border Controller uses the a 250 millisecond default value.

Mid-Session Media Change

Mid-session media change happens during a call that requires the IWF when the type of media being sent while a session is in progress changes. For example, a fax transmission might require mid-session media change; besides fax, other applications of this feature are possible. To support the transmission of a T.38 fax sent over an IWF session, some media channels must be opened and others closed. In addition, the Oracle Communications Session Border

Controller can accommodate a request for media change from, for example, audio to an image type for T.38 fax.

Because the media requirements are driven by endpoints and Gateways, you do not have to configure the Oracle Communications Session Border Controller s mid-session media change support.

Enhanced Support for FAX Calls

The Oracle Communications Session Border Controller now supports T.38 fax calls in networks containing elements that do not comply with the ITU-T H.323 Annex D recommendation for how to replace an existing audio stream with a T.38 fax stream. This support applies to signaling that requires interworking between SIP and H.323.

In the standard call model following the ITU-T recommendation, the endpoint detecting the fax tone sends an H.245 RequestMode message to its peer with a T.38 data mode. The receiving endpoint returns a RequestMode Ack by way of acknowledgement, triggering the sending endpoint to close its audio channel and open a T.38 fax channel. The receiving endpoint closes and opens the same channels on its end. T.38 fax streams flow upon the acknowledgement of all relevant channels.

However, certain endpoints close their logical channel before sending the H.245 RequestMode message for T.38, leaving the Oracle Communications Session Border Controller with its audio channel still open and without having attempted to open a T.38 fax channel. To overcome this issue, the Oracle Communications Session Border Controller now checks whether or not audio channels have been closed whenever it receives an H.245 RequestMode message for T.38. If it finds a closed audio channel, the Oracle Communications Session Border Controller checks for the presence of a matching outgoing audio channel. A match causes the Oracle Communications Session Border Controller to close the audio channel and continue with the procedure for converting to T.38 fax.

Removing the T.38 Codec from an H.245 TCS

For SIP-H.323 IWF sessions, H.323 automatically inserts the T.38 FAX codec in the H.246 TCS message. You can stop this insertion using the **remove-t38** parameter in the H.323 global configuration.

Early Media

For call that require the IWF, the Oracle Communications Session Border Controller supports a cut-through for early media for calls that originate in SIP or H.323.

For a session originating in SIP, the provisional message will contain the SDP information if a Fast Start OLC was received in the Call Proceeding, Alerting, or Progress messages. The same SDP will be sent in the SIP 200 OK.

For a session that starts in H.323, the Oracle Communications Session Border Controller translates the SDP it receives in SIP messages (either a 180 or a 183) into the appropriate H.323 Fast Start elements: Alerting or Progress. If the Alerting or Progress messages contain Fast Start elements, the Progress Indicator Q.931 information element (IE) will also be included in the message with Progress Descriptor 8, indicating that in-band information or an appropriate pattern is now available. This causes the call party to enable end-to-end early media in the reverse direction in accordance with H.323 v4.

In addition, the Oracle Communications Session Border Controller allows early media to flow in the forward direction for a call that requires the IWF starting in H.323 that is being translated to SIP. This happens after the Oracle Communications Session Border Controller has received

provisional response with SDP and has sent Alerting or Progress message with Fast Start to the calling party. Similarly, early media in the forward direction is enabled for a call that requires the IWF starting in SIP and being translated to H.323. This happens after the Oracle Communications Session Border Controller received Alerting or Progress messages with Fast Start and maps the Alerting or Progress to SIP 180 or 183 provisional response with the SDP answer.

Display Name Mapping

The Oracle Communications Session Border Controller displays the full name and number of the calling party (for features such as Caller ID) when it handles calls that require the IWF. The Oracle Communications Session Border Controller takes the display name in the From field of the SIP INVITE and maps it to the display IE so that it can show the full name of the calling party.

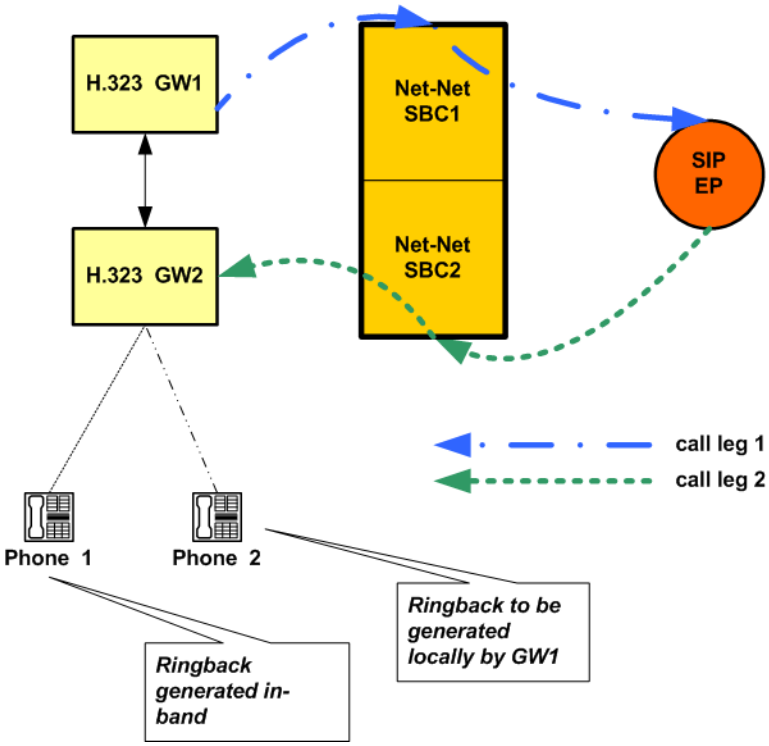
IWF Ringback Support

When interworking SIP and H.323 to a gateway, PSTN gateway, or other endpoint, the Oracle Communications Session Border Controller uses the mappings shown in the table below. The absence or presence of SDP in the SIP provisional message determines whether the tones are generated in-band or locally.

For each of the mappings listed in the following table, this section provides a sample call flow.

SIP Message	H.323 Message
No Message	CallProceeding
No Message	Progress without PI
183 with SDP	Progress with PI
180 w/o SDP	Alert without PI
180 with SDP	Alert with PI

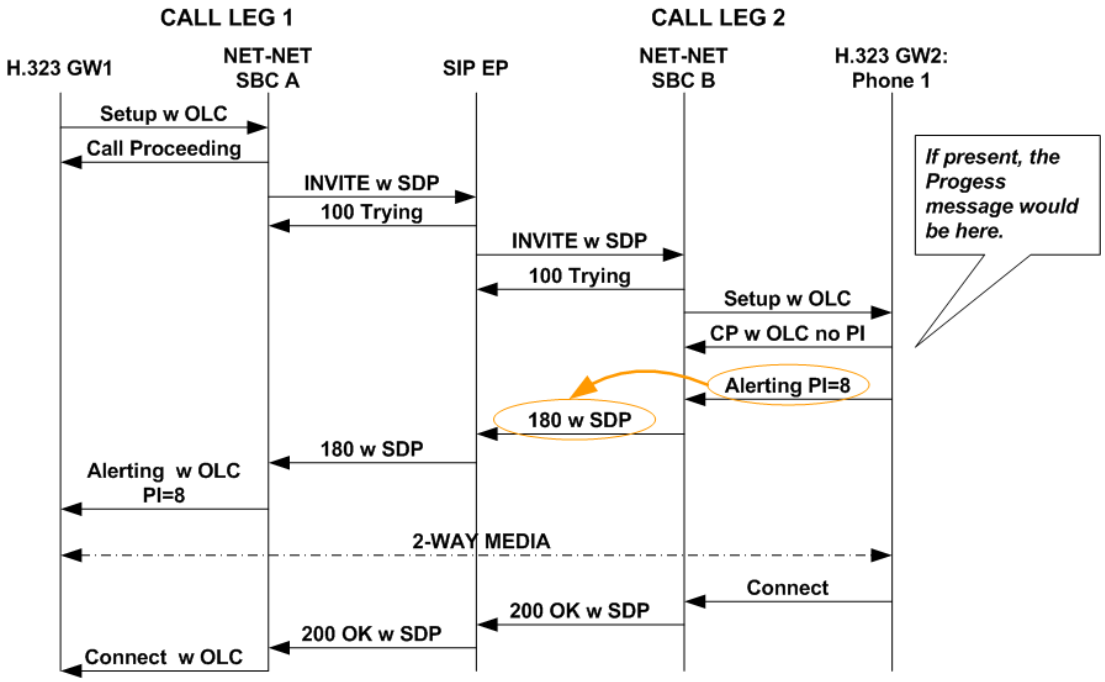
In the following diagram, a call that requires the IWF passes through the Oracle Communications Session Border Controller twice, creating two call legs. The call originates from H.323 GW1 and terminates in Phone 1 or Phone 2.



Sample 1 In-band Ringback without Progress Message

This sample flow shows how the Oracle Communications Session Border Controller handles a call that requires the IWF where there is no progress message. In this call flow, there is a progress indicator of eight (8), meaning that ringback is in-band.

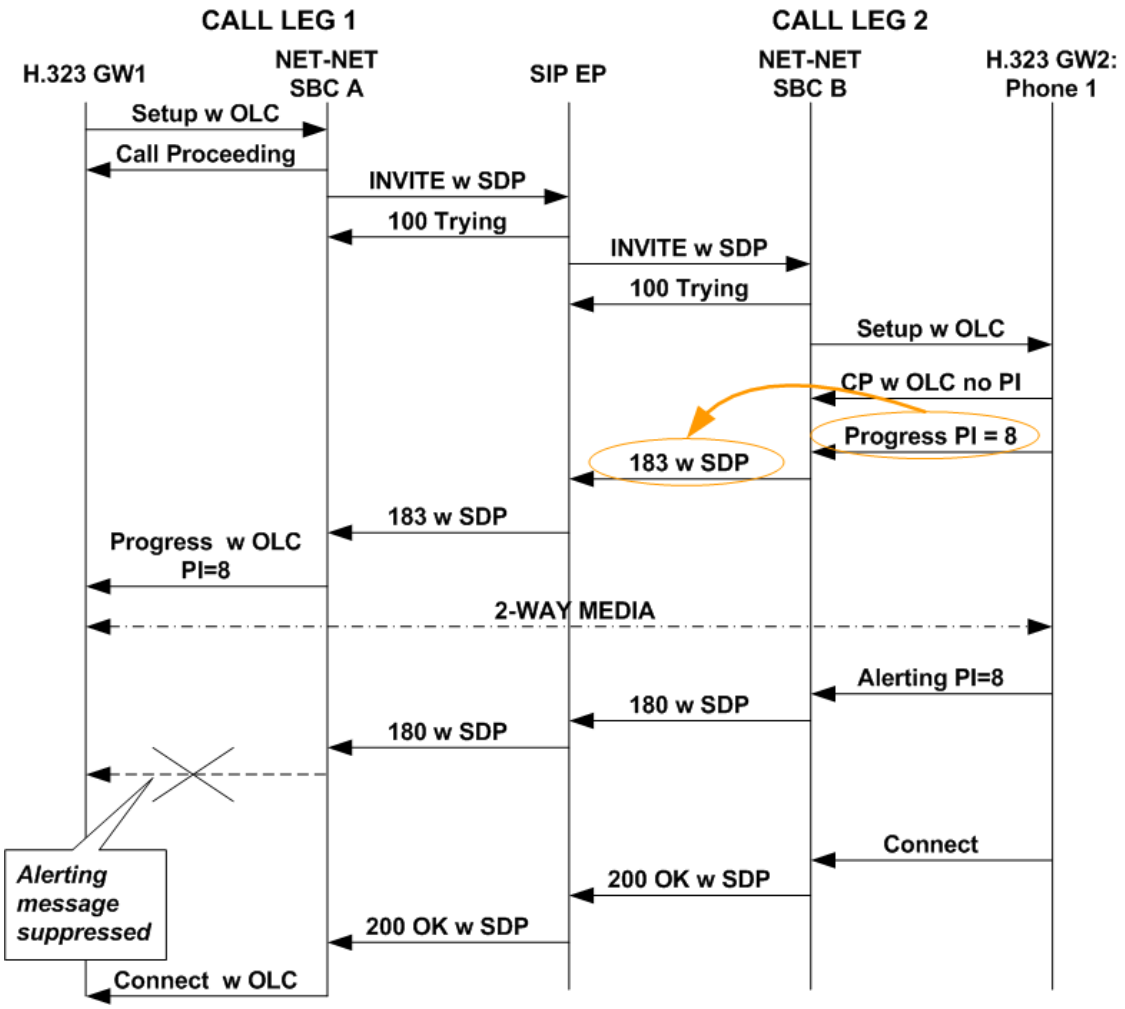
In this diagram, you can see that the Oracle Communications Session Border Controller maps the progress indicator included in the Alerting message sent from Phone 1 through H.323 GW2 to a SIP 180 message with SDP. When the Progress message appears, it contains the progress indicator rather than the Alerting message containing it.



Sample 2 In-band Ringback with Progress Message

This sample flow shows how the Oracle Communications Session Border Controller handles a call that requires the IWF where there is a progress message. In this call flow, there is a progress indicator of eight (8), meaning that ringback is in-band.

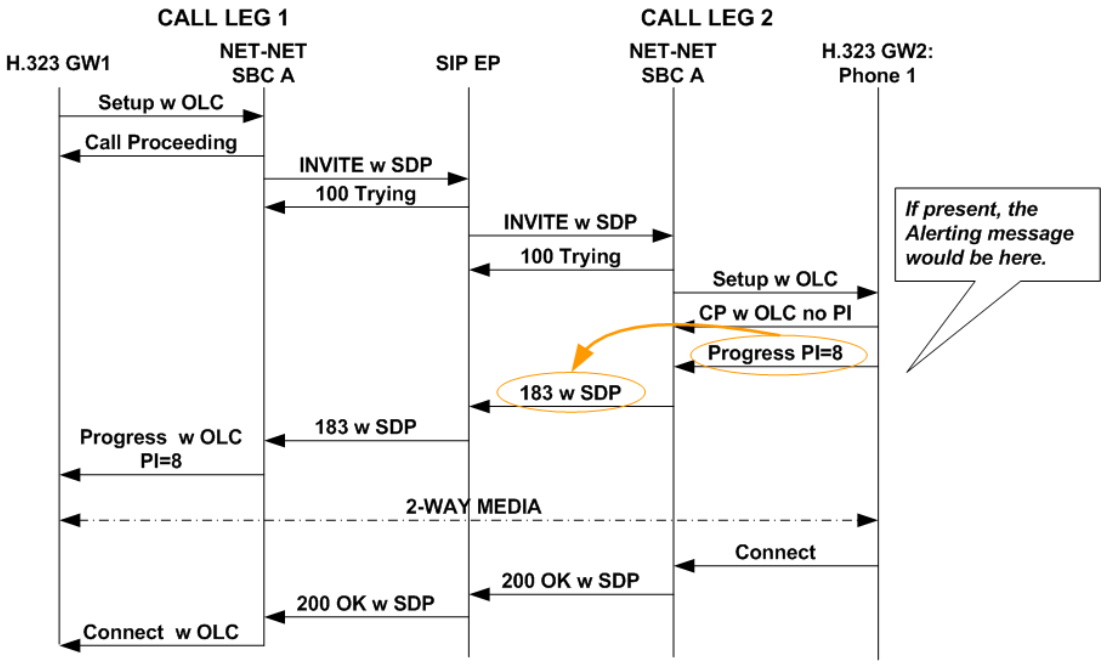
For this call flow, you can see again that the Oracle Communications Session Border Controller maps the progress indicator included in the alerting message sent from Phone 1 through H.323 GW2 to a SIP 180 message with SDP. Note that now the Progress message contains the progress indicator.



Sample 3 In-band Ringback without Alerting Message

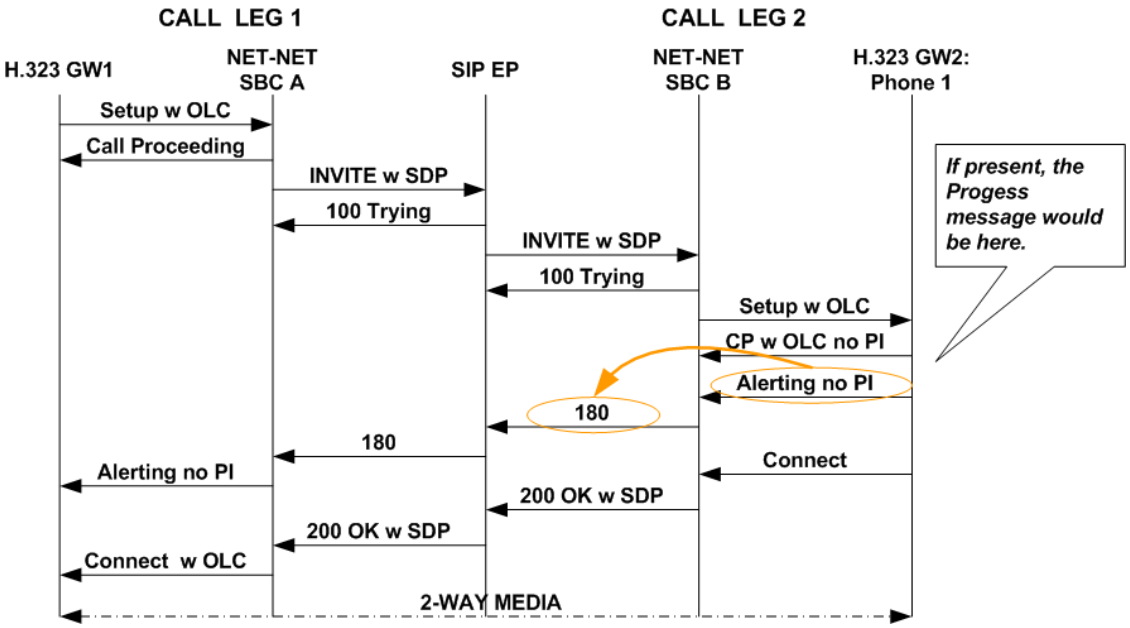
This sample flow shows how the Oracle Communications Session Border Controller handles a call that requires the IWF where there is no progress message. In this call flow, there is a progress indicator of eight (8), meaning that ringback is in-band.

In this diagram, you can see that the Oracle Communications Session Border Controller maps the progress indicator included in the Progress message sent from Phone 1 through H.323 GW2 to a SIP 180 message with SDP. When the Alerting message appears, it contains the progress indicator rather than the Progress message containing it.



Sample 4 Out-of-band Ringback without Progress Message

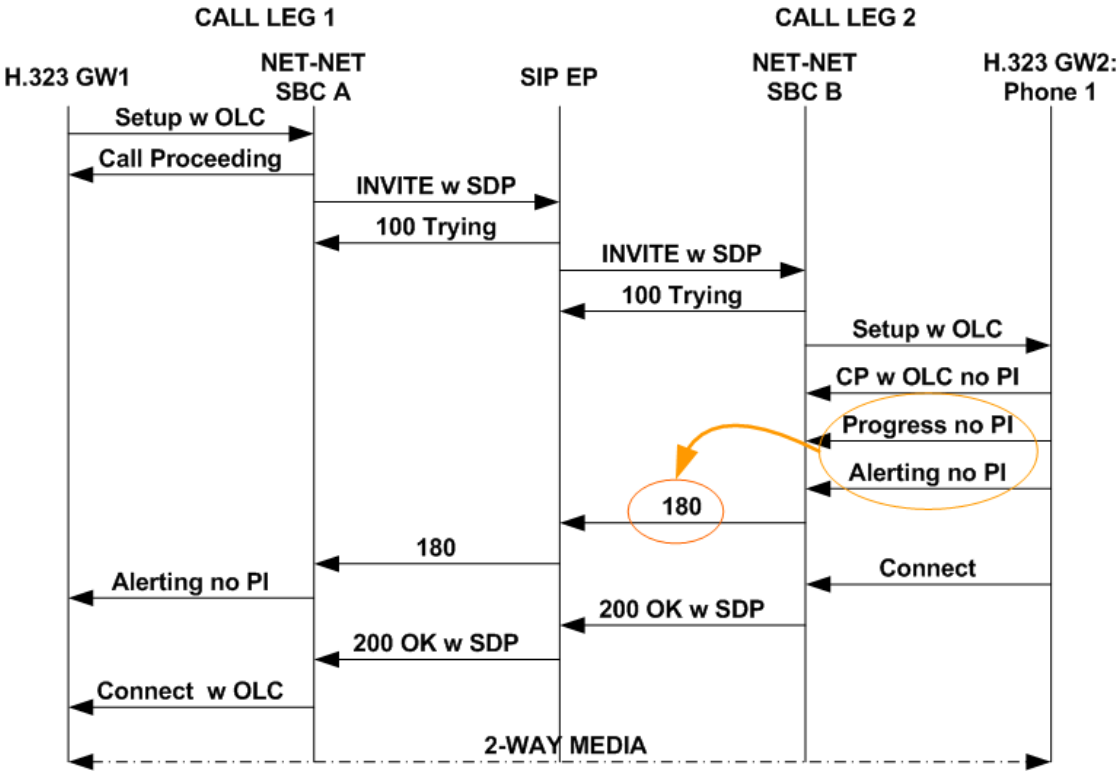
When there is no progress indicator included in the Alerting message, then there is out-of-band ringback. The system maps the Alerting message to a SIP 180, but it it does not include SDP in the SIP 180. This call flow shows that there is no Progress message and that media cannot be set up until after H.323 Connect and SIP messages are sent.



Sample Flow 5 Out-of-band Ringback with Progress Message

When there is no progress indicator included in either the Alerting or Progress messages, then there is out-of-band ringback. The system maps the Alerting message to a SIP 180, but it does

not include SDP in the SIP 180. This call flow shows includes the Progress message; still, media cannot be set up until after H.323 Connect and SIP messages are sent.

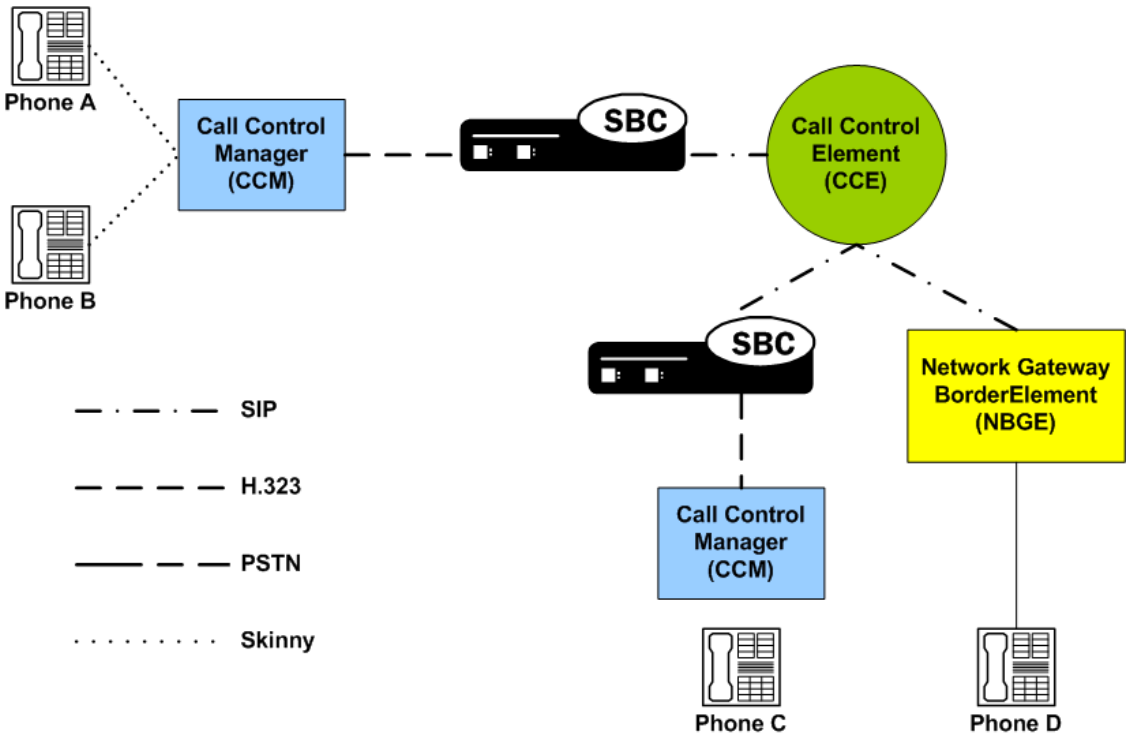


H.323 Endpoint-Originated Call Hold and Transfer

When calls that require the IWF originating in H.323, the SBC supports call hold, transfer, and conference for the H.323 call leg. The call hold and transfer feature uses signaling procedures based on the ITU-T recommendations/H.323 specification for third party initiated pause and rerouting.

You do not have to configure the SBC 's call hold and transfer feature.

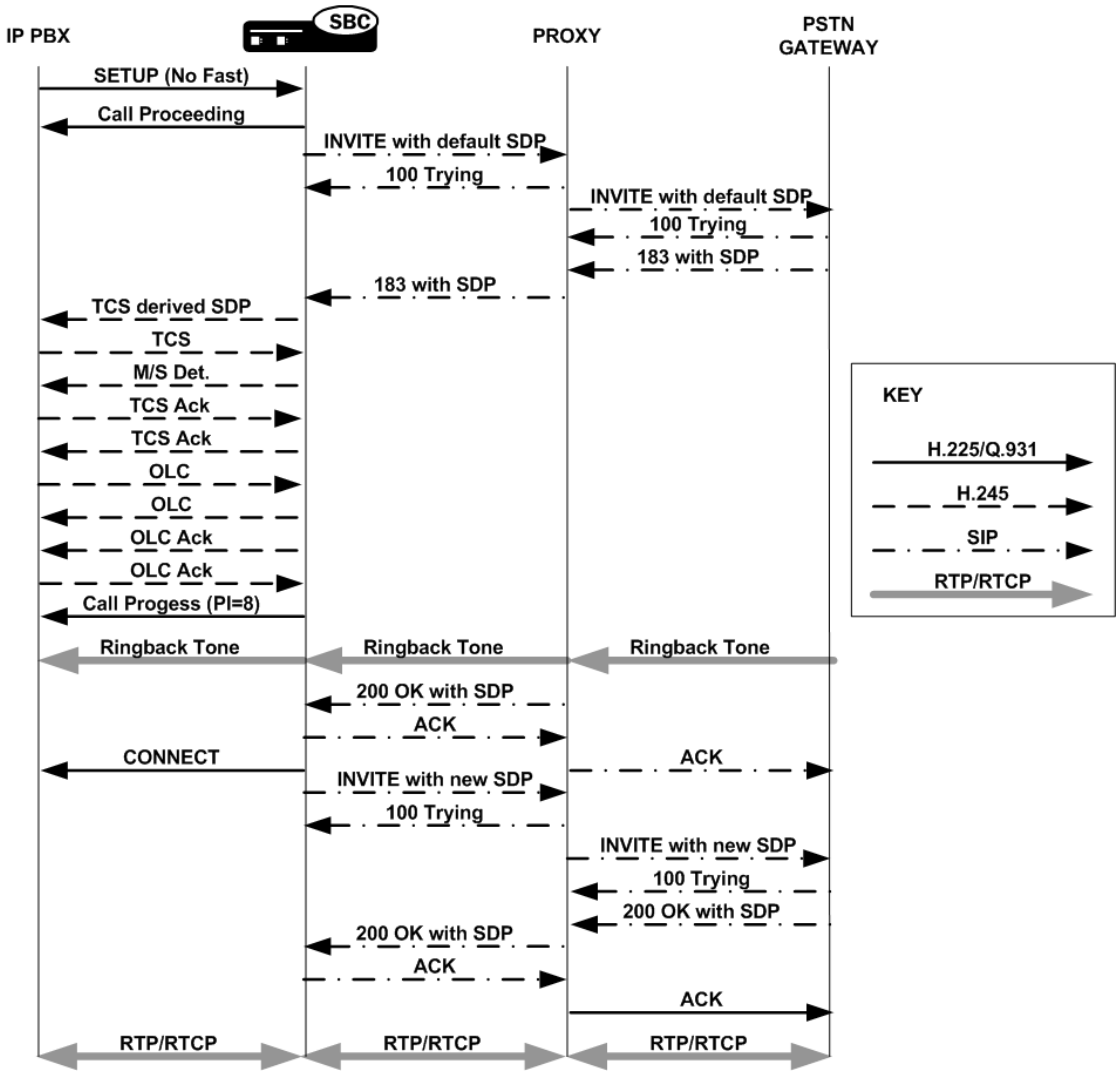
The following diagram shows how the SBC provides call hold and transfer support for IWF sessions that originate in H.323. As you review this section's call flow diagrams, you might want to refer back to the following logical diagram directly below to review the network elements involved, and what protocols they use.



Basic Call

In the following sample basic call, IP PBX A sends an H.323 Slow Starts message ultimately destined for the PSTN through the Oracle Communications Session Border Controller . The Oracle Communications Session Border Controller performs translation to SIP and inserts default information about media. Once the PSTN gateway responds with a 183 containing SDP, the Oracle Communications Session Border Controller sends that information to IP PBX A. Then the Oracle Communications Session Border Controller and the IP PBX exchange TCS- and OLC-related messages, and they negotiate primary-secondary determination. The Oracle Communications Session Border Controller also sends IP PBX A a Call Progress message with a progress indicator of 8.

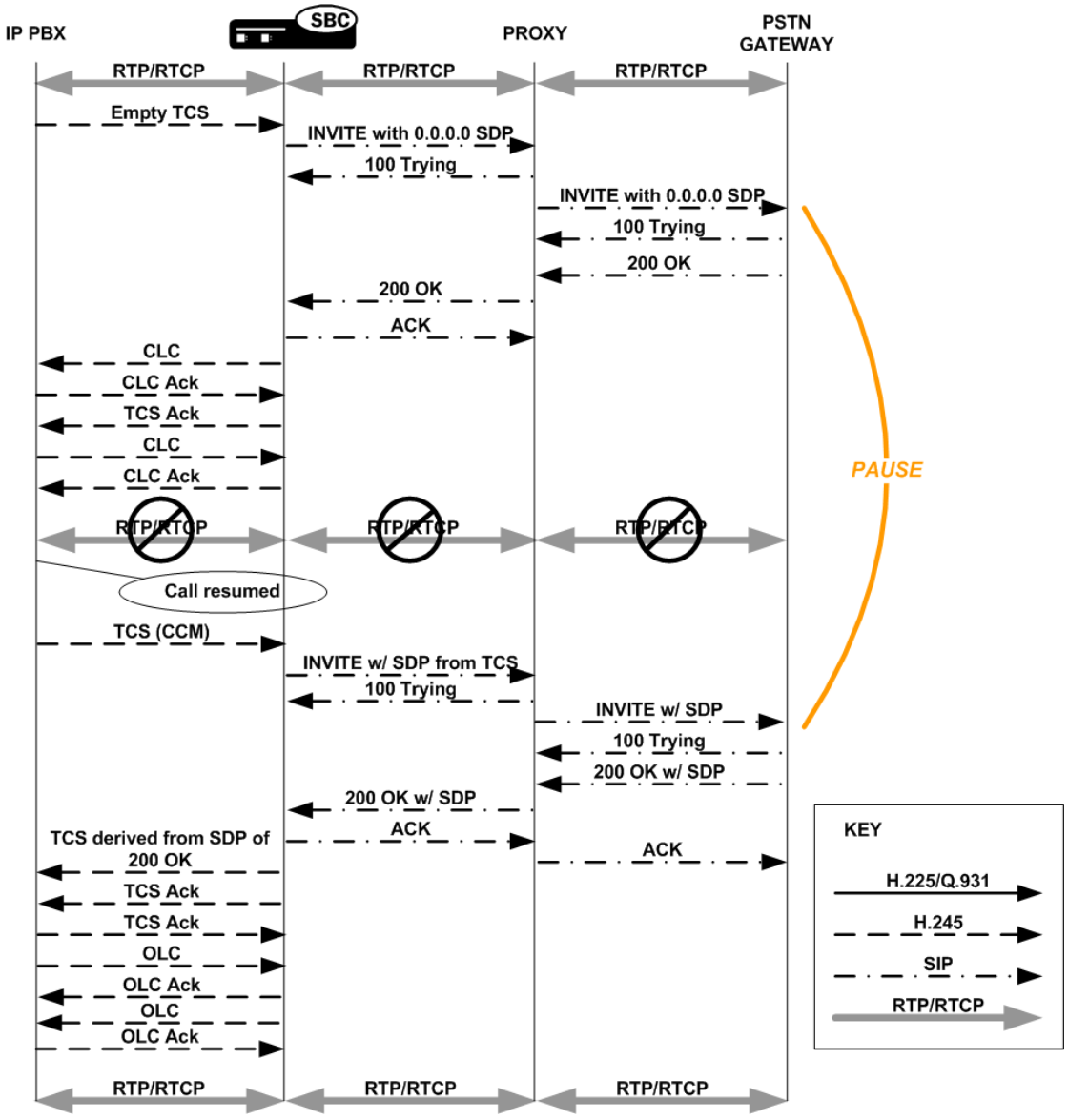
After the ring-back tone, the proxy sends a 200 OK message with SDP to the system. The Oracle Communications Session Border Controller sends a Connect message to the IP PBX A, and then it sends another SIP INVITE to the proxy that contains amended SDP (if that information about media is different from the default). After 200 OK and ACK messages are exchanged, media (RTP/RTCP) flow takes place.



Hold

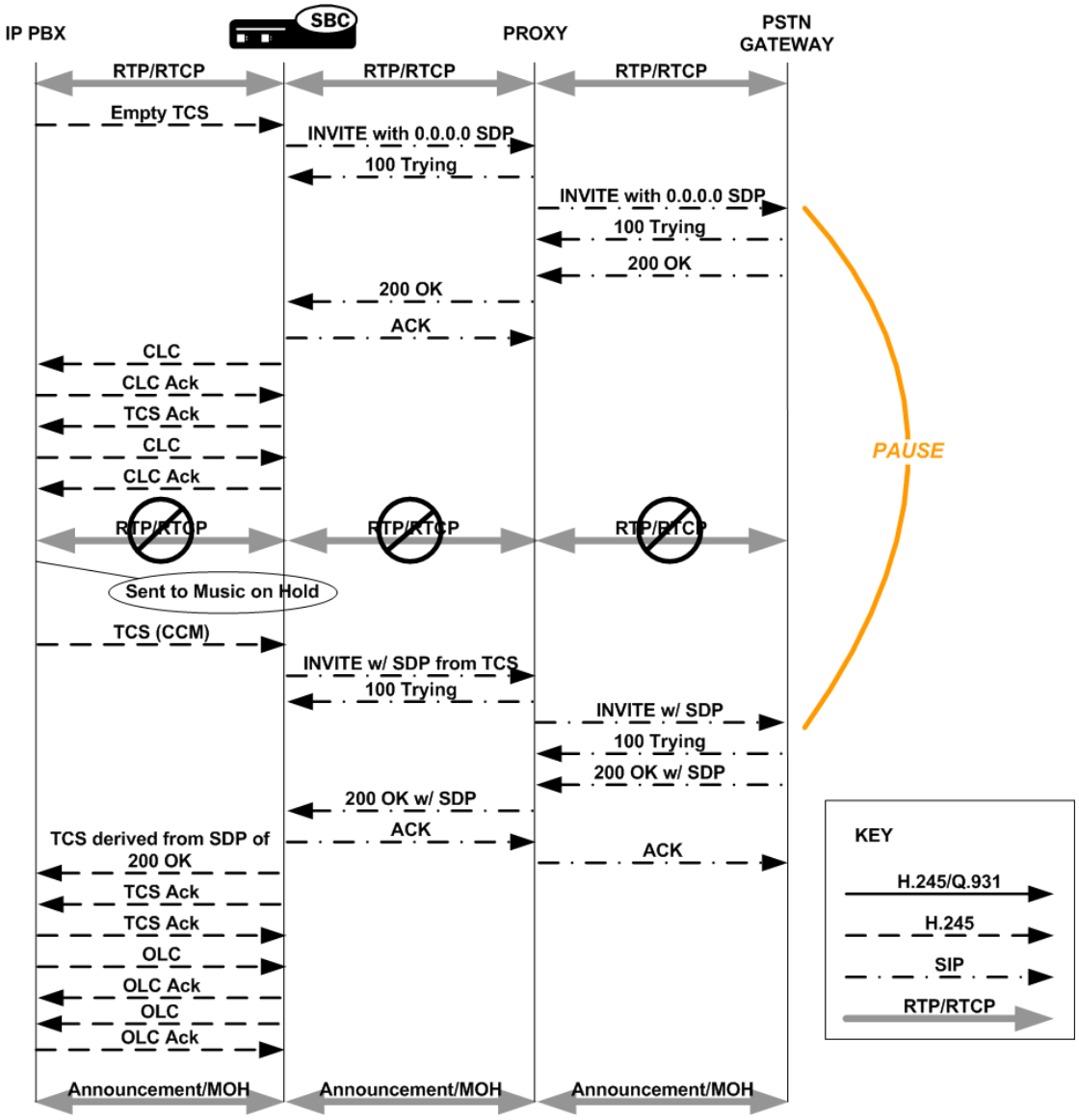
This sample call flow assumes that the IWF call is established and that the RTP/RTCP flow is already in progress. The hold button is pushed, and IP PBX A sends an empty TCS to the Oracle Communications Session Border Controller . The Oracle Communications Session Border Controller puts the called party on hold by sending an INVITE message with 0.0.0.0 SDP to the SIP side of the call. Using 0.0.0.0 as the media address effectively stops the media flow. This INVITE is acknowledged, and the Oracle Communications Session Border Controller closes the channels on the H.323 side, halting the RTP/RTCP flow.

When the caller on the H.323 side takes the call off hold, it resumes with a TCS that the Oracle Communications Session Border Controller receives and then translates on the SIP side as an INVITE with SDP. After that INVITE is acknowledged and received, the Oracle Communications Session Border Controller opens logical channels on the H.323 side and RTP/RTCP flows resume.



Music On Hold

This scenario is similar to the hold feature enabled for calls that require the IWF, except that after the RTP/RTCP flow between the H.323 and SIP sides stops, the call is sent to music on hold. Before the announcement or music plays, the SBC sets up the necessary support for media to be exchanged.



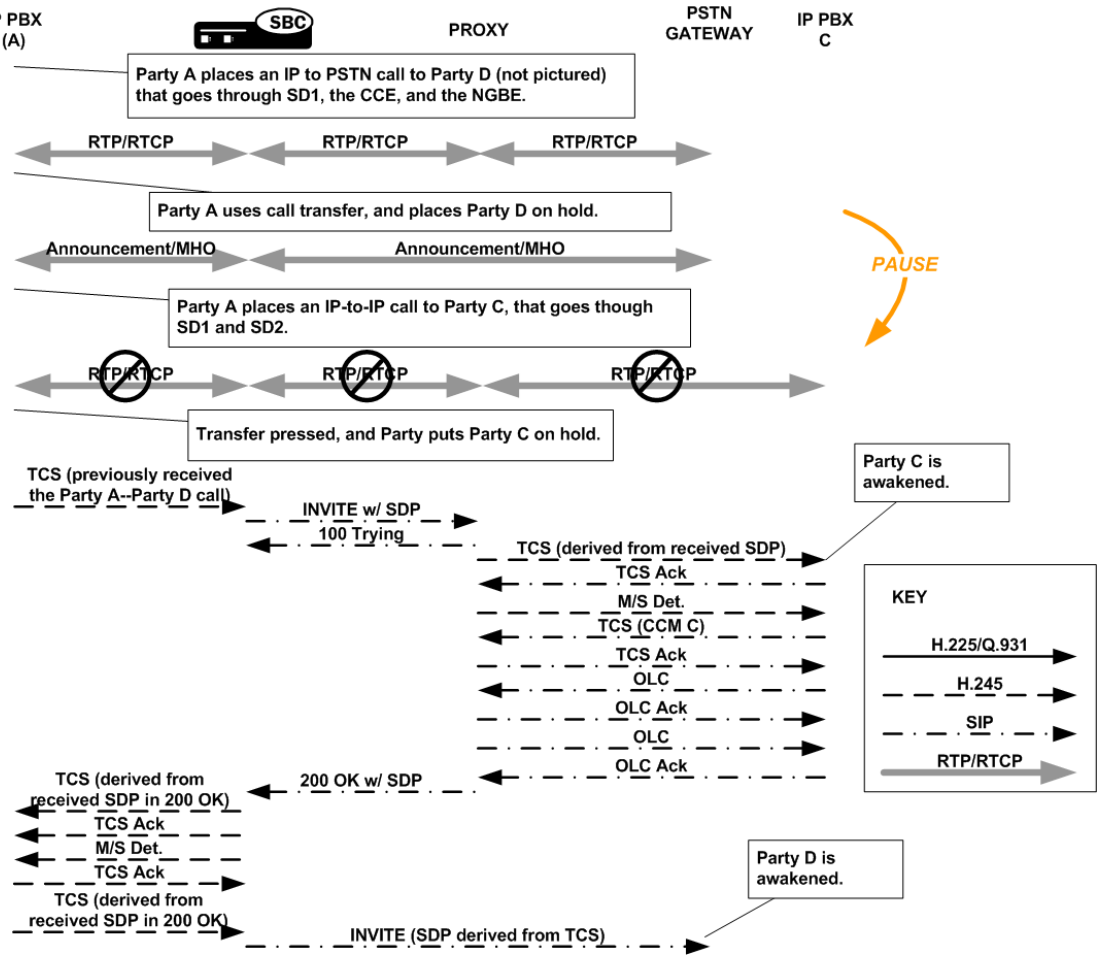
Transfer

The call flow described in this section recalls the diagram at the top of the H.323 Endpoint-Originated Call Hold and Transfer section, where endpoints A, B, and C are H.323 devices and endpoint D is a SIP device. When you follow the signaling and media flows, note that there are two Oracle Communications Session Border Controller s in the call transfer and two sets of SIP/H.323 translations that take place. The first Oracle Communications Session Border Controller translates H.323 to SIP, and the second performs the same operations with the protocols reversed.

In the scenario pictured, Party A is on a call with Party D, but wants to transfer Party C to Party D. Party A places Party D on hold, and then makes the call to Party C. Party A then puts Party C on hold, pressing the transfer button. You can see that Oracle Communications Session Border Controller 1 receives a TCS from the IP PBX, which is then translated to SIP. Oracle Communications Session Border Controller 2 receives it, performs the required protocol translations, and then opens a session with Party C via another IP PBX. Once this session is up and Party D is awakened, channels are established for media exchange.

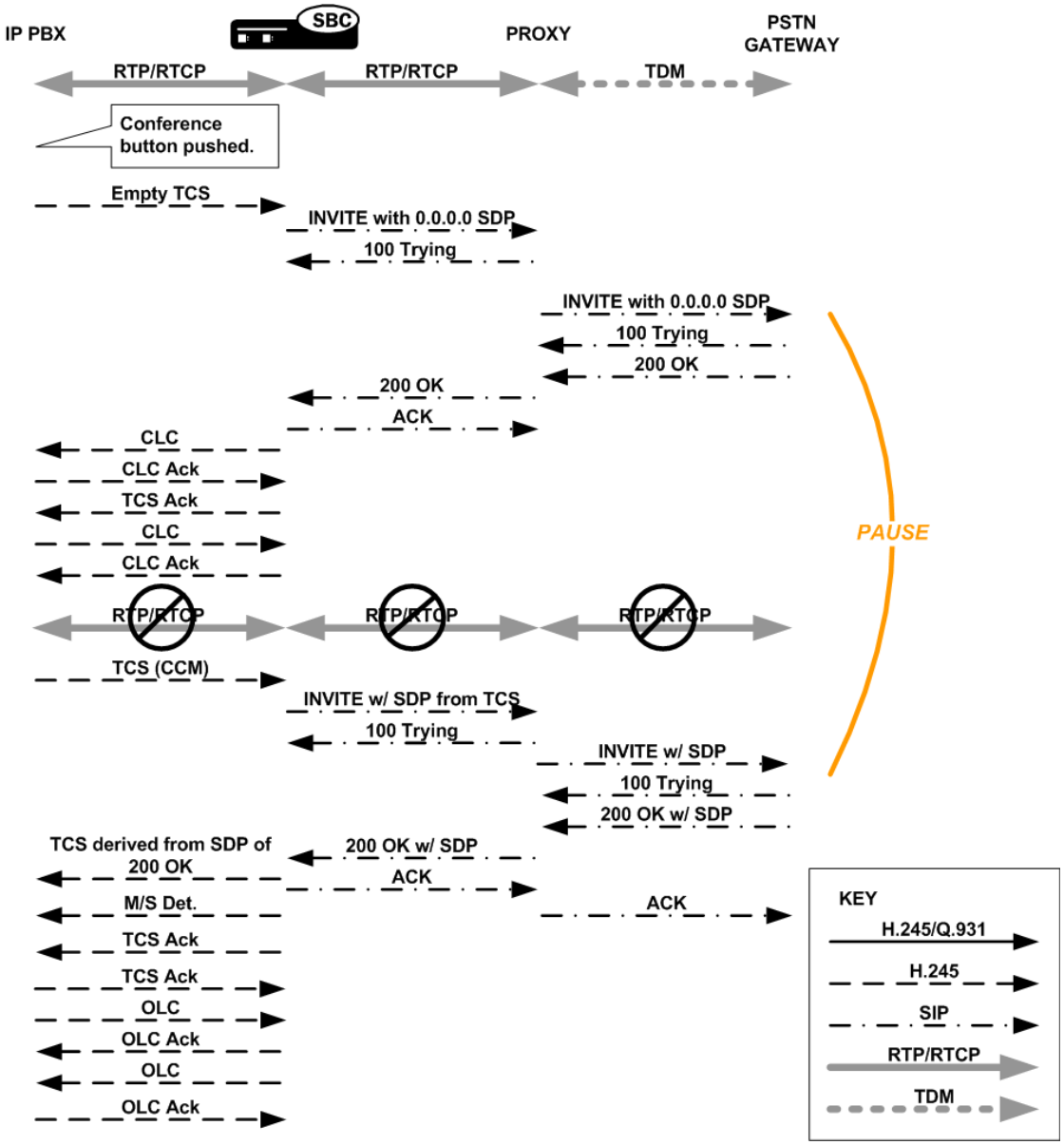
In order to redirect the media so that it flows between Party C and Party D, the Oracle Communications Session Border Controller 1 and IP PBX C exchange OLC and OLC Ack messages that contain address information for Party C and for Party D. Address information for both parties is contained in the OLC Ack messages that the Oracle Communications Session Border Controller exchanges with the IP PBX. IP PBX A does not move forward with the call until it has the necessary address information.

Even though Party A's participation in the call stops early in this scenario, the IP PBX with which it is associated keeps the signaling sessions with the Oracle Communications Session Border Controller alive to manage the transfer.



Conference

To conference a call that requires the IWF that starts in H.323, the SBC uses a scenario much like the one used for holding a call that requires the IWF. Here again, the INVITE with 0.0.0.0 as the media address and the closing of logical channels stops the flow of RTP/RTCP. After signaling and SDP/media information are re-established, RTP/RTCP for the conference flows.



IWF Call Forwarding

This section describes the Oracle Communications Session Border Controller ’s IWF Call Forwarding feature, which is supported for calls initiated in SIP that require interworking to H.323.

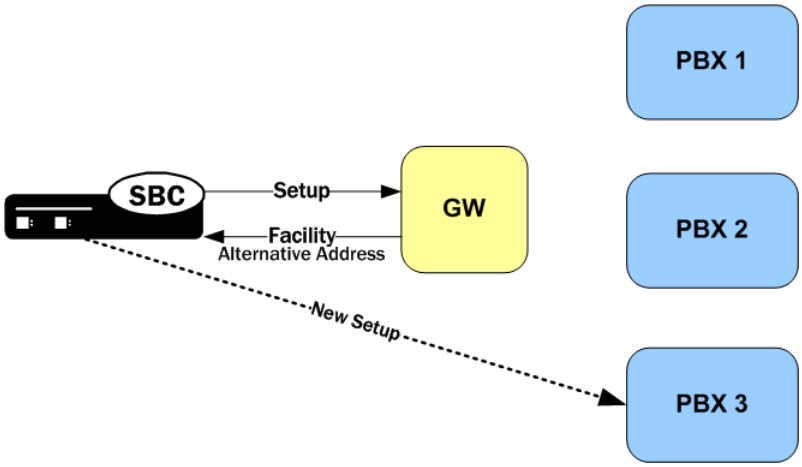
Prior to the implementation of this feature, the Oracle Communications Session Border Controller did not forward calls when the remote H.323 endpoint sent a Facility message with Call deflection as the reason and an alternate address for forwarding. Instead, it would either:

- Fail to release the initial call and initiate the forwarded call
- Drop the entire call when the remote endpoint for the call tore down the session

New Behavior

In the diagram below, you can see that the Oracle Communications Session Border Controller sends the initial Setup message to the gateway, and the gateway returns the Facility message with an alternate address for forwarding. Rather than engaging in its former behavior, the Oracle Communications Session Border Controller now releases the call with the gateway and sends a new Setup to the alternate address from the Facility message.

This new Setup up has no effect on the first call leg, which remains connected.



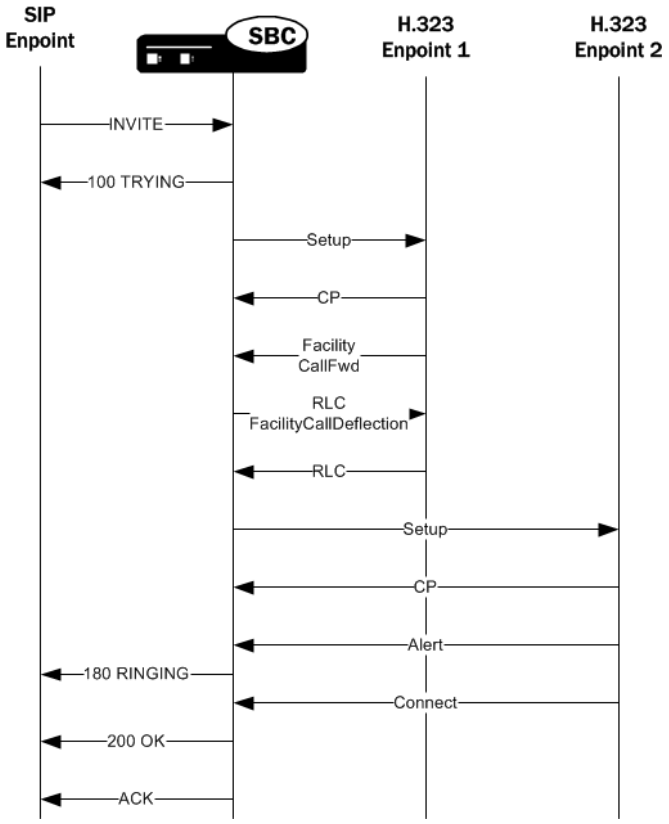
When it receives a Facility message with the reason CallForwarded, the Oracle Communications Session Border Controller looks for an alternate transport address in the Facility's alternativeAddress or alternativeAliasAddress element. The Oracle Communications Session Border Controller releases the egress call with the reason facilityCallDeflection. Then it takes one of two courses of action:

- If it does not find an alternative address, the Oracle Communications Session Border Controller releases the ingress call (with 486 BUSY HERE for a call being interworked from SIP to H.323).

If it finds an alternative address and the egress call has not been alerted or answered, the Oracle Communications Session Border Controller at this point tries to initiate a new egress call. The Oracle Communications Session Border Controller uses the alternative alias address to populate the calledPartyNumber information element (IE) and the destination address of the new Setup.

H.323 Sample Call Flow

The following diagram shows how the H.323 Call Forwarding feature works in a purely H.323 environment.



Media Release for H.323 SS-FS Calls for IWF

When the Oracle Communications Session Border Controller routes a slow-start to fast-start call, it is possible for the same fast-start call to be routed back through the Oracle Communications Session Border Controller making for a hairpin flow. If it does becomes a hairpin flow, then the Oracle Communications Session Border Controller routes it to its destination as a fast-start to fast-start call. This can result in one-way media if:

- The destination of the hairpin call is in the same realm as the originating slow-start to fast-start call
- The realm reference in the first bullet item is configured to disable in-realm media management
- The called endpoint accepts the proposed fast-start logical channels

The enhancements to the Oracle Communications Session Border Controller 's behavior described in this section show how the Oracle Communications Session Border Controller follows additional procedures when setting up a hairpin flow to avoid one-way media when media release occurs.

H.323

For H.323 calls, the Oracle Communications Session Border Controller establishes media using the H.245 procedures described in the H.245 ITU-T recommendation: control protocol for multimedia communication. It also uses the Fast Connect procedure defined in the H.323 ITU-T recommendation: packet-based multimedia communication systems.

The latter ITU-T recommendation allows a calling endpoint to send a Setup message that contains a fastStart element, a sequence of OLC structures that describe the calling endpoint's proposed forward/reverse logical channels. If the called endpoint accepts this proposal, then logical channels are established.

When the Oracle Communications Session Border Controller translates a call originating in slow-start to fast-start, it uses a Fast Connect procedure in the outgoing leg by sending an outgoing Setup that includes a fastStart element with one or more OLC structures. But when the Oracle Communications Session Border Controller constructs this message, it is unaware of whether the call will become hairpinned or if media release will occur. Because it does not yet have this information, the Oracle Communications Session Border Controller sets the Network Address and the TSAP identifier in the OLC structures to the ingress IP address and port of a corresponding media flow allocated for media traveling between the calling and called endpoints. So if the called endpoint accepts the fastStart the Oracle Communications Session Border Controller proposes, the called endpoint would send its media to the Oracle Communications Session Border Controller . After acceptance, the system starts H.245 procedures on the slow-start side of the call to set up logical channels on that side. Then the Oracle Communications Session Border Controller updates the IP address and port of the media flows using OLC and OLCAck messages received from the calling endpoint.

This procedure works well for endpoints that are not in the same realm, or that are in the same realm for which media management is disabled, because each endpoint must send its media through the Oracle Communications Session Border Controller . When the endpoints are in the same realm and when media management is enabled, however, the Oracle Communications Session Border Controller must perform additional steps for media release in slow-start to fast-start calls.

To support media release in slow-start to fast-start calls, the Oracle Communications Session Border Controller performs a hold-and-resume procedure on the fast-start side. After it establishes channels on the slow-start side and if it detects media release being enabled, the Oracle Communications Session Border Controller sends and empty TCS to the fast-start side to put that side on hold. Then the called endpoint closes all the logical channels it previously opened in the Fast Connect procedure and stops transmitting to them. And the Oracle Communications Session Border Controller also closes it logical channels. Once the channels are closed, the Oracle Communications Session Border Controller resumes the call by sending a new, restricted TCS to the fast-start side. The restricted TCS only contains the receive and transmit capabilities of the codecs types that the called endpoint accepted in the Fast Connect procedure, and it forces the called endpoint to re-open logical channels of the same codec types accepted in the Fast Connect procedure. Once it receives and OLC from the called endpoint, the Oracle Communications Session Border Controller sends on OLCAck with the Network Address and TSAP identifier for the logical channel from the calling endpoint. Then the Oracle Communications Session Border Controller re-opens logical channels (of the same codec types that it open in the Fast Connect procedure). If the called endpoint has not changed its Network Address and TSAP identifier for its logical channels, media is re-established after the Oracle Communications Session Border Controller and the called endpoint exit the hold state. The last steps is for the Oracle Communications Session Border Controller to re-sends the full TCS message from the calling to the called endpoint to inform the called endpoint of the full capabilities of the calling endpoint.

Hold-and-Resume Procedure

The hold-and-resume procedure has three states:

- Media Hold—Starts when the Oracle Communications Session Border Controller sends the empty TCS to the called endpoint to put it on hold.

When it detects media release, the Oracle Communications Session Border Controller puts the called endpoint on hold. It can only do so if it has exchanged the TCS-TCSAck messages and completed primary-secondary determination with the calling endpoint.

When the Oracle Communications Session Border Controller receives a TCSAck in response to the empty TCS that it sent to the called endpoint, it closes the logical channels it opened as part of the Fast Connect procedure; the called endpoint likewise closes its logical channels. The two then exchange CLC and CLCAck messages, which signals the start of the Media Resume state.

- Media Resume—Starts when the Oracle Communications Session Border Controller sends a restricted TCS to resume the call.
  The restricted TCS the Oracle Communications Session Border Controller sends contains only the receive and transmit capabilities of the codec types previously accepted by the called endpoint in the Fast Connect procedure. This forces the called endpoint to re-open logical channels of the same codec type that were previously accepted in the Fast Connect procedure.

  After sending this TCS, the Oracle Communications Session Border Controller is ready (as specified in the ITU-T recommendations) to take part on the primary-secondary determination (MSD) process. However, the called party and not the Oracle Communications Session Border Controller initiates the MSD if it is required. The MSD is completed if necessary. Alternately, the called endpoint can start to re-open its logical channels. When it receives the first OLC from the called endpoint, the Oracle Communications Session Border Controller also starts to re-open its logical channels.
- Media Complete—Starts when all the logical channels that the Oracle Communications Session Border Controller re-opens are acknowledged by the called endpoint.
  When it enters the Media Complete state, the Oracle Communications Session Border Controller updates the called endpoint with the full capabilities of the calling endpoint by sending the full TCS.

Additional IWF Steps

For calls originating in slow-start H.323 that require interworking to SIP, the Oracle Communications Session Border Controller also takes addition steps for media release in hairpinned flows that the Oracle Communications Session Border Controller routes as SIP to fast-start H.323.

For such a call, after the Oracle Communications Session Border Controller has established logical channels on the slow-start H.323 side of the call, it sends a reINVITE on the SIP side. This reINVITE has an updated session description to correct the media connection information. The the Oracle Communications Session Border Controller performs the hold-and-resume procedure on the fast-start side of the call. This procedure re-establishes the logical channels between the Oracle Communications Session Border Controller and the called endpoint, avoiding the one-way media problem.

When you are configuring H.323 globally on your Oracle Communications Session Border Controller , you might choose to set the noReInvite option. This option stops the Oracle Communications Session Border Controller from sending a reINVITE after the logical channels are established on the slow-start H.323 side of the call. Instead, the Oracle Communications Session Border Controller ’s H.323 task communicates internally with its own SIP task a SIP UPDATE message that corrects the SDP; then the SIP task updates media flow destinations. But the Oracle Communications Session Border Controller does not send the UPDATE to the next hop, which can result in the one-way media problem if the call is hairpinned and media release occurs. For such cases, the default behavior for the noReInvite option is overridden. When the Oracle Communications Session Border Controller detects media release in an

H.323-SIP call, it forwards the UPDATE to the next hop even when you enable the noReInvite option.

Dependencies

This feature depends on:

- The H.323 endpoint supports the third-party-initiated pause and re-routing feature.
- The H.323 endpoint does not change its Network Address and TSAP identifier when it re-opens the logical channels.
- The H.323 endpoint does not immediately tear down the call when there is not established logical channel in the call.
- The fact that the SIP endpoint supports the UPDATE message if the noReInvite option is enabled.

Before You Configure

The Oracle Communications Session Border Controller 's IWF requires that there be complete configurations for both SIP and for H.323. These two sets of configurations function together when the interworking is configured and enabled.

You enable the Oracle Communications Session Border Controller 's interworking capability when you set the IWF configuration's state parameter to enabled, and all required H.323 and SIP configurations are established. This means that all of the following configurations must be established:

- A full SIP configuration, including SIP interfaces, SIP ports, SIP-NATs (if needed), and SIP features
- A full H.323 configuration, including H.323 global and H.323 interface configurations
- Local policy and local policy attributes (the IWF will not work without these configurations)
- Media profiles
- Session agents and, if needed, session agent groups

H.323 Configuration

You must have a complete configuration to support H.323 traffic on your Oracle Communications Session Border Controller , including any required support for H.323 Fast Start or Slow Start.

In the H.323 interface configuration, you are able to configure interfaces that enable communication between H.323 devices (for audio, video, and/or data conferencing sessions).

If you know that your Oracle Communications Session Border Controller will be handling traffic originating in Slow Start H.323, you must establish the appropriate list of media profiles in the IWF configuration. Handling Slow Start traffic also requires that you establish appropriate local policy (and local policy attribute) configurations, but configuring session agents and session agent groups is optional.

SIP Configuration

SIP functionality must also be configured on your Oracle Communications Session Border Controller that will perform IWF translations. You must use appropriate local policy (and local

policy attribute) configurations, but configuring session agents and session agent groups is optional. If you use session agents, then you must also configure the information you need for media profiles.

The Role of Local Policy

You must configure local policies (and local policy attributes, if necessary) in order for translations between SIP and H.323 to take place. These local policies determine what protocol is used on the egress side of a session. Local policy and local policy attribute configurations make routing decisions for the session that are based on the next hop parameter that you set. The next hop can be any of the following:

- IPv4 address of a specific endpoint
- Hostname or IPv4 address of a session agent
- Name of a session agent group

You can use the application protocol parameter in the local policy attributes configuration as a way to signal the Oracle Communications Session Border Controller to interwork the protocol of an ingress message into a different protocol as it makes its way to its egress destination (or next hop).

For example, if you set the application protocol parameter to SIP, then an inbound H.323 message will be interworked to SIP as it is sent to the next hop. An inbound SIP message would travel to the next hop unaffected. Likewise, if you set the application protocol parameter to H.323, then an incoming SIP message will be interworked to H.323 before the Oracle Communications Session Border Controller forwards it to the next hop destination.

The following example shows a configured local policy and its attributes used for IWF traffic.

```
local-policy
        from-address
                                        *
        to-address
                                        444
        source-realm                    *
        state                           enabled
        last-modified-date              2004-04-20 17:43:13
        policy-attribute
                next-hop                        sag:sag_internal
                realm                           internal
                replace-uri                     disabled
                carrier
                start-time                      0000
                end-time                        2400
                days-of-week                    U-S
                cost                            0
                app-protocol                    SIP
                state                           enabled
                media-profiles
```

Local Policy in an IWF Session Initiated with H.323

In a session where the Oracle Communications Session Border Controller is interworking H.323 to SIP, it internally forwards the session on for interworking when:

- The next hop in the local policy is configured as a SIP session agent

- The next hop in the local policy is configured as a SIP session agent group
- The next hop in the local policy is not configured as a session agent or session agent group, and the application protocol parameter is set to SIP in the local policy attributes configuration.

Local Policy in an IWF Session Initiated with SIP

In a session where the Oracle Communications Session Border Controller is interworking SIP to H.323, it internally forwards the session on for interworking when:

- The next hop in the local policy is configured as an H.323 session agent
- The next hop in the local policy is configured as an H.323 session agent group
- The next hop in the local policy is not configured as a session agent or session agent group, and the application protocol parameter is set to H.323 in the local policy attributes configuration
  In this case the local policy should also define the egress realm, which you can set in the realm parameter of the local policy attributes configuration.

SIP-H.323 interworking with Dynamic Payload Types

The SIP and H.323 Protocols use Internet multimedia signaling over IP, and both use the Real-Time Transport Protocol (RTP) for transferring realtime audio/video data. The interworking function (IWF) provides a means of converting translation and signaling protocols and session descriptions between SIP and H.323. However, SIP and H.323 provide different mechanisms when exchanging payload types for media during IWF calls. Therefore, the International Telecomunications Union (ITU) modified the ITU H.245 recommendations in H.245 v16 to include a new “Dynamic Payload Type Replacement” capability that resolves this payload type conflict. This new capability provides a way for an H.323 endpoint to specify the payload type of a media stream for which the endpoint is willing to receive through the OLCacknowledgment (OLC-ACK) message in an audio/video call flow.

The Oracle Communications Session Border Controller supports this new “Dynamic Payload Type Replacement” capability by ensuring interworking of SIP and H.323 when audio/video call flows use dynamic payload types. The Oracle Communications Session Border Controller checks for the presence of this capability in the incoming TCS request. If it finds this capability in the TCS request, it sends an Open Logic Channel Acknowledgement (OLC-ACK) response with the payload type it is willing to receive.

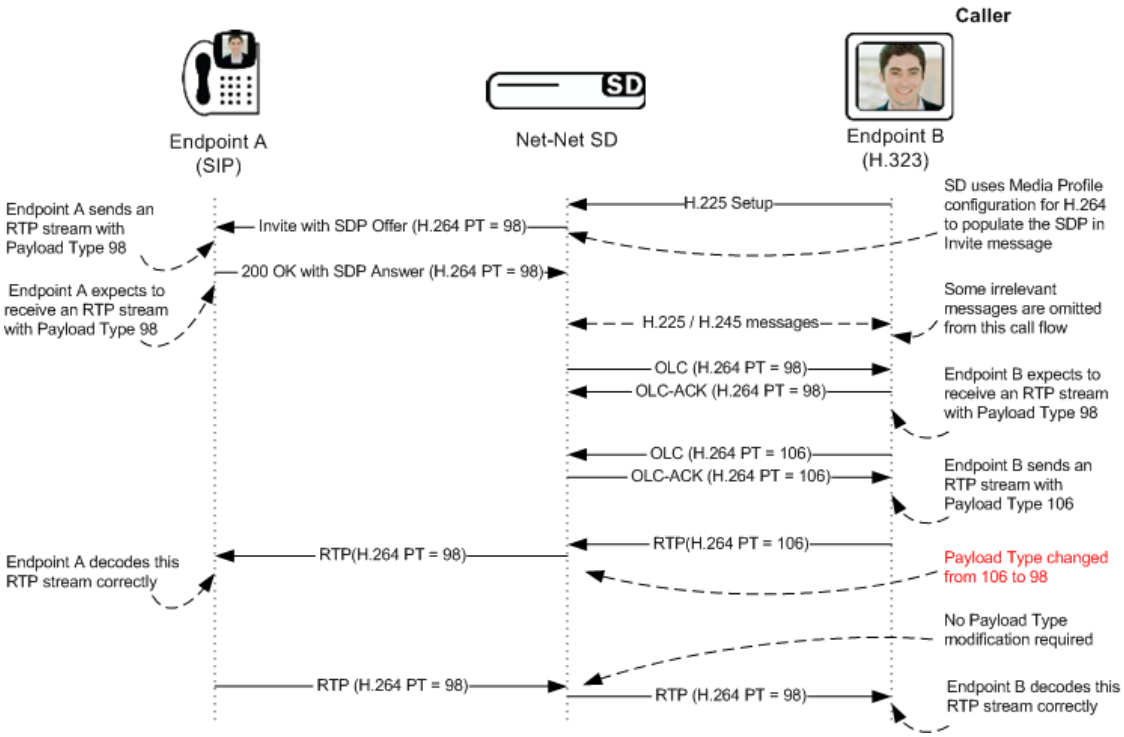
> **Note:**
>
> The Oracle Communications Session Border Controller always returns an OLC-ACK with a dynamic payload type value that it received in the incoming Session Description Protocol (SDP) from the SIP endpoint.

For devices that don't support the H.245 v16 recommendations, the Terminal Capability Set (TCS) request from the H.323 endpoint does not have the "Dynamic Payload Type Replacement" capability present. Therefore, the Oracle Communications Session Border Controller rewrites the payload type within the RTP packets when these packets traverse the Oracle Communications Session Border Controller. When devices in a session negotiate different payload types between SIP and H.323 packets, the RTP streams that they receive, always have the expected payload type in the RTP header.

> **Note:**
>
> The Oracle Communications Session Border Controller always maps the payload type on the RTP stream received from the H.323 endpoint, and sends it to the SIP endpoint for both audio and video. The Oracle Communications Session Border Controller does not support mapping of payload types in audio streams with 2833 DTMF packets.

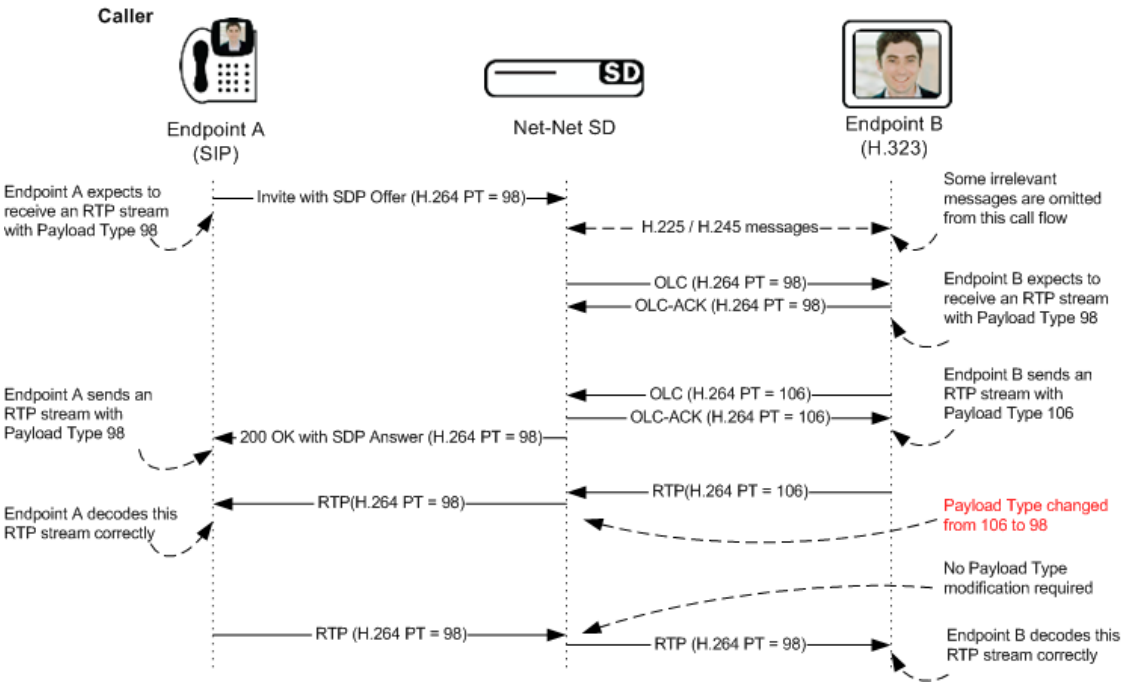
Figure 1a and 1b below shows the call flow from an H.323 Endpoint B to a SIP Endpoint A, and from a SIP Endpoint A to an H.323 Endpoint B, respectively. These illustrations show the negotiation of different dynamic payload types for the video streams but the Codec negotiated is the same. The Oracle Communications Session Border Controller dynamically replaces the payload type in the RTP header of the video stream received from the H.323 endpoint.

Figure 1a Endpoint B calling Endpoint A (H.323 endpoint does not have “Dynamic Payload Type Replacement” Capability)



The H.323 Endpoint B is not H.245 v16 compliant, and hence payload type replacement needs to be done in the RTP packets.

Figure 1b Endpoint A calling Endpoint B (H.323 endpoint does not have Dynamic Payload Type Replacement Capability)

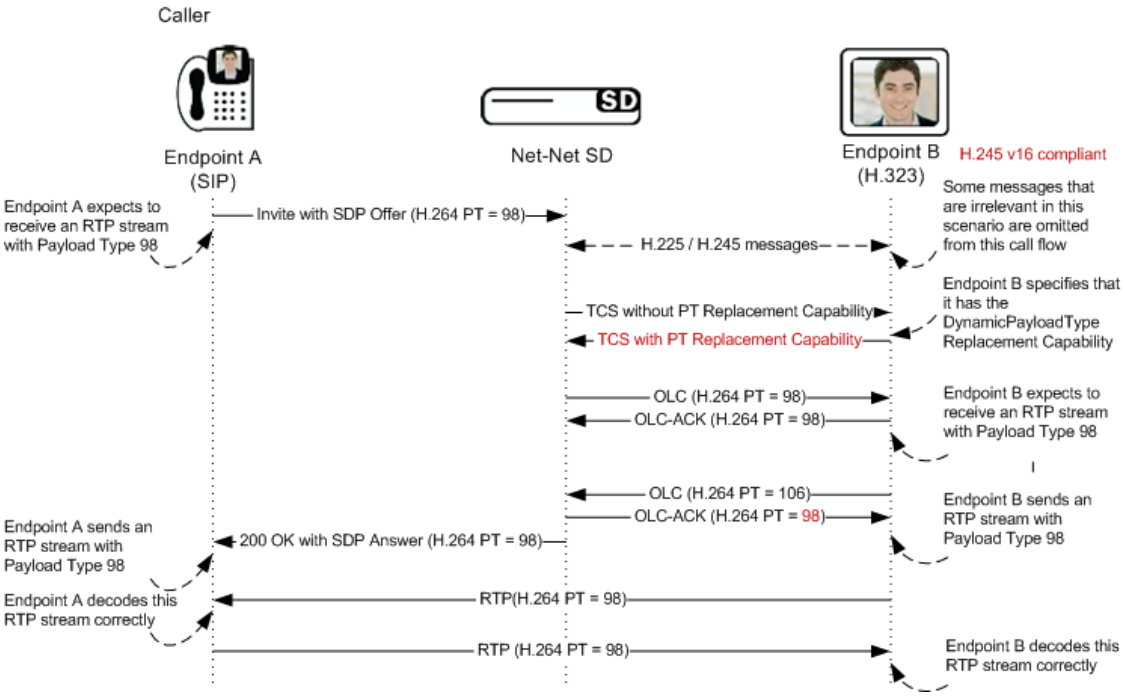


The H.323 Endpoint B is not H.245 v16 compliant, and therefore payload type replacement needs to be done in the RTP packets.

There is no concept of H.245 compliance for the SIP Endpoint A.

Figure 2a shows the call flow of SIP Endpoint A calling an H.323 Endpoint B using slow start. The Oracle Communications Session Delivery Manager modifies the dynamic payload type in the OLC-ACK based on payload type received in the incoming SDP OFFER in the "INVITE" message.

Figure 2a Endpoint A calling Endpoint B (H.323 endpoint has TCS with Dynamic Payload Type Replacement Capability)

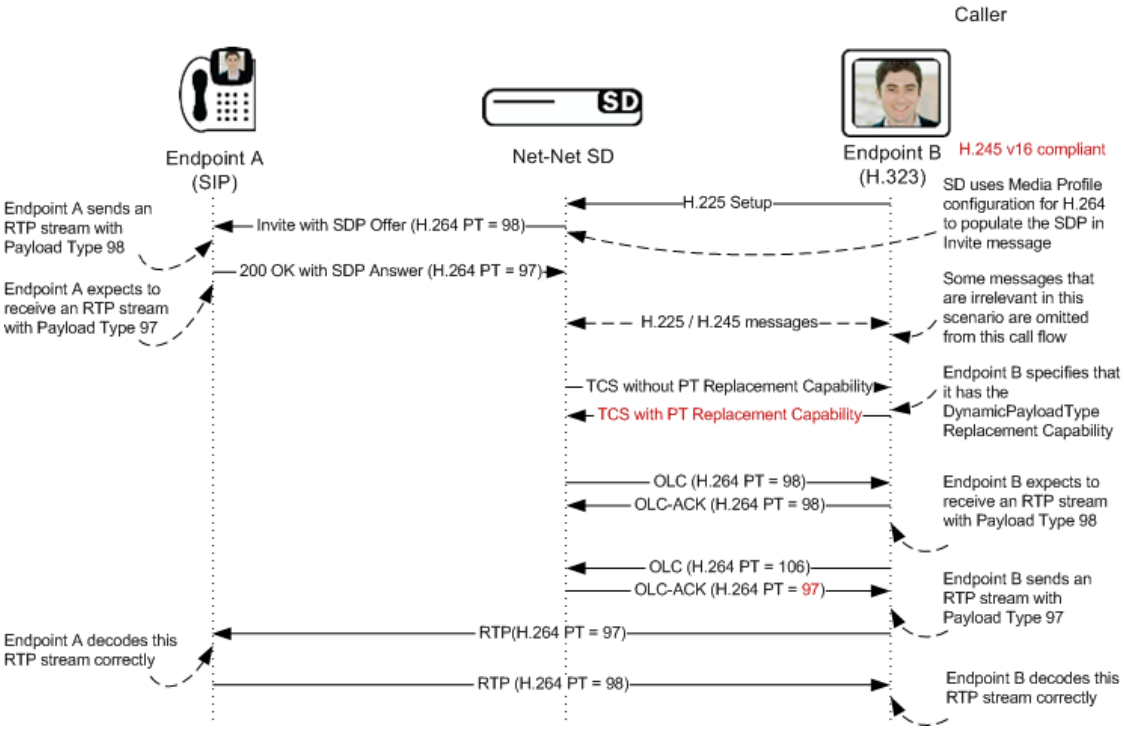


The H.323 Endpoint B is H.245 v16 compliant.

There is no concept of H.245 compliance for the SIP Endpoint A.

Figure 2b shows the call flow an H.323 Endpoint B using slow start, calling a SIP Endpoint A. The Oracle Communications Session Delivery Manager modifies the dynamic payload type in the OLC-ACK based on payload type received in the incoming SDP ANSWER in the "200 OK" message.

Figure 2b Endpoint B calling Endpoint A (H.323 endpoint here has TCS without “Dynamic Payload Type Replacement” Capability)



Configuring Interworking

If you have already completed the steps outlined in this chapter’s IWF Service Enhancements section, then enabling the IWF is a simple process. This section shows you how to enable the IWF, and how to enable certain features that you can use to supplement basic IWF functionality.

An IWF configuration might appear like this in the ACLI:

```
iwf-config
        state                           enabled
        media-profiles
                                        PCMU
                                        telephone-event
        logging                         disabled
```

Configure IWF

To enable the Inter-working Function (IWF) on your Oracle Communications Session Border Controller :

1. Access the **iwf-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# iwf-config
ORACLE(iwf-config)#
```

2. **state**—Enable this parameter if you want to translate SIP and H.323 sessions on your Oracle Communications Session Border Controller . The default value is **disabled**. Valid values are:
   - enabled | disabled
3. **media-profiles**—Enter the name of the media profiles you want to use for IWF translations. This name is either the name of an SDP codec (such as PCMU), or it can be telephone-event if you are configuring your system for DTMF support.

   If you want to use more than one media profile for SIP/H.323 translations, enter the names in quotation marks with a space between each one.

```
ORACLE(iwf-config)# media-profiles "PCMU telephone-event"
```

4. **logging**—Enable this parameter if you want the Oracle Communications Session Border Controller to log SIP messages that are related to the IWF. The default value is **disabled**. Valid values are:
   - enabled | disabled
5. **forward source call**—Type **enable** if you want to the Oracle Communications Session Border Controller to add the `h225SourceCallSignalAddress` header to the egress SIP INVITE.
6. Type **done** to save your configuration.

Topology Hiding for IWF with an Internal Home-Realm

You can configure the SBC to mask the IP address of the originating caller in the SIP From and/or P-Asserted-Identity headers when calls are placed from H.323 to SIP endpoints.

The option **NoPAssertedID** checks for incoming SETUP messages have the presentation indicator set to restricted and instructs the SBC to send a Privacy header without the P-Asserted-Identity and not to make the From header anonymous.

The option **replace-call-origin-ip** removes the calling party's IP address in the SIP From header. The IP address from the internal home realm is used instead.

The topology hiding feature uses the presentation indicator field from an inbound H.323 setup message to determine if/how the headers will be masked. The following table summarizes the configurable SBC parameters and the values for the From and P-asserted-identity headers.

H.255 Presentation Indicator Setting	P-Asserted-Identity Header Value	From Header Value	SD Session Agent Option
Allow	IP address from home realm of SD SIP Config	H.255 Calling Party IP/ Port	No Option Set
Allow	IP address from home realm of SD SIP Config	IP address of Home Realm SIP-Interface	NoPAssertedID
Allow	IP address from home realm of SD SIP Config	IP address of Home Realm SIP-Interface	replace-call-origin-ip

H.255 Presentation Indicator Setting	P-Asserted-Identity Header Value	From Header Value	SD Session Agent Option
Allow	IP address from home realm of SD SIP Config	IP address of Home Realm SIP-Interface	replace-call-origin-ip, NoPAssertedID
Restricted	PAI Header not present	Anonymous	No Option Set
Restricted	PAI Header not present	Anonymous	NoPAssertedID
Restricted	PAI Header not present	Anonymous	replace-call-origin-ip
Restricted	PAI Header not present	Anonymous	NoPAssertedID, replace-call-origin-ip

IWF Topology Hiding Configuration

To enable IWF topology hiding on your Oracle Communications Session Border Controller :

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. The system prompt changes.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Use the ACLI **select** command so that you can work with the session agent configuration to which you want to add these options.

   ```
   ORACLE(session-agent)# select
   ```

5. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**replace-call-origin-ip**), and then press Enter. Follow the same steps to add the **NoPAssertedID** option.

   ```
   ORACLE(session-agent)# options +replace-call-origin-ip
   ORACLE(session-agent)# options +NoPAssertedID
   ```

   If you type either of these options without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

DTMF Support

For calls that require the IWF, you can enable support for the relay of RFC 2833 DTMF digits. The availability of this feature means that the Oracle Communications Session Border Controller is compliant with RFC 2833, which defines two payload formats for DTMF digits. To learn more about this RFC, refer to http://www.ietf.org/rfc/rfc2833.txt.

Until the exchange of TCS messages with the H.323 endpoint, the Oracle Communications Session Border Controller has no information about the endpoint's RFC 2833 capabilities. The Oracle Communications Session Border Controller adds telephone-event to the SDP on the SIP side of the call.

For calls that require SIP/H.323 translation, you can enable support for the relay of RFC 2833 DTMF digits.

To use this feature, you need to configure a media profile called telephone-event and set relevant parameters for it. Application of the media profile can happen either in a session agent configuration or in the IWF configuration.

- The **name** parameter in the media profiles configuration
- The **media-profiles** list in the IWF configuration
- The **media-profiles** list in the session agent configuration

All of the scenarios outlined here assume that you have established a telephone-event media profile configuration.

You can configure DTMF support using the following parameters. The way that the Oracle Communications Session Border Controller uses these values is described below. The payload type, part of the media profiles configuration, is dynamic and varies with different endpoints, so there is no default configuration for the telephone-event media profile.

The telephone-event media profile is used as follows in these types of IWF sessions:

- **Calls that require the IWF originating in H.323 Slow Start**—There is no channel (media) information available on the H.323 side.
  - If the incoming H.323 endpoint is configured as a session agent on the Oracle Communications Session Border Controller , then the telephone-event parameter in the media profiles set for that session agent configuration will be used in the SDP on the SIP side of the session.
  - If the H.323 endpoint is not a session agent or the telephone-event media profile is not configured in the session agent configuration corresponding to the endpoint, then the Oracle Communications Session Border Controller refers to the media profile information configured for the IWF configuration.
- **Calls that require the IWF originating in SIP—**If the TCS was not exchanged before a 200 OK was sent on the SIP side, the Oracle Communications Session Border Controller will behave in one of these two ways.
  - If the outbound H.323 endpoint is configured as a session agent, then the media profiles from that session agent configuration will be used.
  - If the outbound H.323 endpoint is not configured as a session agent, the media profile configured within the IWF configuration with the telephone-event value will be used.

As mentioned above, DTMF support is configured by using a combination of the telephone-event media profile and either the session agent or IWF configuration. First you set up the media profile, then you apply it to a session agent or to the IWF configuration.

DTMF Configuration

DTMF support requires you to configure a media profile named telephone-event. This section shows you how to set it up.

To configure a telephone-event media profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ```

   From this point, you can configure parameters for media profiles. To view see all media profile configuration parameters, enter a **?** at the system prompt.

4. **name**—Enter the name telephone-event and press Enter.
5. **parameters**—Enter the parameters to be applied for the codec; these are the digits that endpoints can support.
6. **media-type**—Leave the default media type set to audio.
7. **payload-type**—Set the payload type to **101**, which is the dynamic payload type needed to support this feature.
8. **transport**—Leave the default transport protocol set to RTP/AVP.
9. **frames-per-packet**—You can leave this parameter set to **0** (default).
10. **req-bandwidth**—You can leave this parameter set to **0** (default).

Applying the Media Profile

After you have configured the telephone-event media profile, you need to apply it either to a H.323 session agent or the global IWF configuration.

DTMF for all IWF translations

To use DTMF for all IWF translations:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **iwf-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# iwf-config
   ```

   From this point, you can configure IWF parameters. To view see all IWF configuration parameters, enter a **?** at the system prompt.

4. Add the telephone-event media profile to the media profiles list and save your work. If you already have a media profiles for the IWF configuration set up and want to keep them (adding telephone-event to the list), then you must type in all of the media profiles that you want to use.

```
ORACLE(iwf-config)# media-profiles "PCMU telephone-event"
ORACLE(iwf-config)# done
```

DMTF Support on a Per-Session-Agent Basis

To use DMTF support on a per-session-agent basis:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
```

From this point, you can configure IWF parameters. To view see all IWF configuration parameters, enter a **?** at the system prompt.

4. When you configure a new H.323 session agent, you can configure DTMF support by simply adding the telephone-event media profile to the list of media profiles. You can add it along with the other media profiles you might want to use for that session agent.

```
ORACLE(session-agent)# media-profiles "telephone-event g711Ulaw64k"
```

5. When you want to add DTMF support to an H.323 session agent that you have already configured, you need to select that session agent, add the media profile, and save your work.

```
ORACLE(session-agent)# select
<hostname>:
1: 192.168.1.48 realm=         ip=
2: 192.168.1.49 realm=         ip=
3: 192.168.1.50 realm=external ip=
selection:3
ORACLE(session-agent)# media-profiles "telephone-event g711Ulaw64k"
ORACLE(session-agent)# done
```

RFC 2833 DTMF Interworking

This section explains the Oracle Communications Session Border Controller's support of transporting Dual Tone Multi-Frequency (DTMF) in Real-Time Transport Protocol (RTP) packets (as described in RFC 2833) to H.245 User Input Indication (UII) or SIP INFO method interworking.

Multimedia devices and applications must exchange user-input DTMF information end-to-end over IP networks. The Oracle Communications Session Border Controller provides the interworking capabilities required to interconnect networks that use different signaling protocols. Also, the Oracle Communications Session Border Controller provides DTMF translation to communicate DTMF across network boundaries.

The Oracle Communications Session Border Controller supports:

- RFC 2833 to H.245 UII translation for H.323-to-H.323 calls, when one side is a version 4 H.323 device requiring RFC-2833 DTMF event packets, and the other side is a pre-version 4 H.323 device that only uses H.245 UII.
- RFC 2833 to H.245 UII or INFO translation of H.323 to SIP (and SIP to H.323) IWF calls, when one side is a version 4 H.323 device requiring RFC 2833 DTMF event packets and the SIP endpoint only supports INFO messages. Or when one side is a pre-version 4 H.323 device that only uses H.245 UII and the SIP endpoint supports RFC-2833 DTMF event packets.

About RFC 2833

RFC 2833 specifies a way of encoding DTMF signaling in RTP streams. It does not encode the audio of the tone itself, instead a signal indicates the tone is being sent. RFC 2833 defines how to carry DTMF events in RTP packets. It defines a payload format for carrying DTMF digits used when a gateway detects DTMF on the incoming messages and sends the RTP payload instead of regular audio packets.

About H.245 UII

H.245 provides a capability exchange functionality to allow the negotiation of capabilities and to identify a set of features common to both endpoints. The media and data flows are organized in logical channels. H.245 provides logical channel signaling to allow logical channel open/close and parameter exchange operations. The H.245 signaling protocol is reliable, which ensures that the DTMF tones will be delivered.

H.245 User Input Indication (UII) plays a key role in all the services that require user interaction. For video messaging, typical uses of UII include selection of user preferences, message recording and retrieval, and typical mailbox management functions. H.245 UII provides two levels of UII, alphanumeric and signal.

About RFC 2833 to H.245 UII Interworking

The Oracle Communications Session Border Controller provides 2833 to H.245-UII interworking by checking 2833-enabled RTP streams for packets matching the payload type number for 2833. It then sends the captured packet to the host for processing and translation to H.245 UII messages. A H.245 UII message received by the Oracle Communications Session Border Controller is translated to 2833 packets and inserted into the appropriate RTP stream.

About DTMF Transfer

DTMF transfer is the communication of DTMF across network boundaries. It is widely used in applications such as interactive voice response (IVR) and calling card applications.

The multiple ways to convey DTMF information for packet-based communications include:

- In-band audio: DTMF digit waveforms are encoded the same as voice packets. This method is unreliable for compressed codecs such as G.729 and G.723
- Out-of-band signaling events:
  H.245 defines out-of-band signaling events (UII) for transmitting DTMF information. The H.245 signal or H.245 alphanumeric methods separate DTMF digits from the voice stream and send them through the H.245 signaling channel instead of through the RTP channel. The tones are transported in H.245 UII messages.

  All H.323 version 2 compliant systems are required to support the H.245 alphanumeric method, while support of the H.245 signal method is optional.

  SIP INFO – uses the SIP INFO method to generate DTMF tones on the telephony call leg. The SIP INFO message is sent along the signaling path of the call. Upon receipt of a SIP INFO message with DTMF content, the gateway generates the specified DTMF tone on the telephony end of the call.
- RTP named telephony events (NTE): uses NTE to relay DTMF tones, which provides a standardized means of transporting DTMF tones in RTP packets according to section 3 of RFC 2833.

Of the three RTP payload formats available, the Oracle Communications Session Border Controller supports RTP NTE. NTE is most widely used for SIP devices but is also supported in H.323 version 4 or higher endpoints.

RFC 2833 defines the format of NTE RTP packets used to transport DTMF digits, hookflash, and other telephony events between two peer endpoints. With the NTE method, the endpoints perform per-call negotiation of the DTMF transfer method. They also negotiate to determine the payload type value for the NTE RTP packets.

The NTE payload takes the place of codec data in a standard RTP packet. The payload type number field of the RTP packet header identifies the contents as 2833 NTE. The payload type number is negotiated per call. The local device sends the payload type number to use for 2833 telephone event packets using a SDP or H.245 Terminal Capability Set (TCS), which tells the other side what payload type number to use when sending the named event packets to the local device. Most devices use payload type number 101 for 2833 packets, although no default is specified in the standard.

The 2833 packet's RTP header also makes use of the timestamp field. Because events often last longer than the 2833 packets sending interval, the timestamp of the first 2833 packet an event represents the beginning reference time for subsequent 2833 packets for that same event. For events that span multiple RTP packets, the RTP timestamp identifies the beginning of the event. As a result, several RTP packets might carry the same timestamp.

See RFC 2833 and draft-ietf-avt-rfc2833bis-07.txt for more information.

Preferred and Transparent 2833

To support preferred (signaled) 2833 and transparent 2833, the Oracle Communications Session Border Controller provides 2833 detection and generation (if necessary) when the endpoint signals support for 2833.

- Preferred: the Oracle Communications Session Border Controller only generates and detects 2833 for endpoints if they negotiate support for 2833 through signaling
- Transparent: the Oracle Communications Session Border Controller offers and answers based on end-to-end signaling and transparently relaying 2833

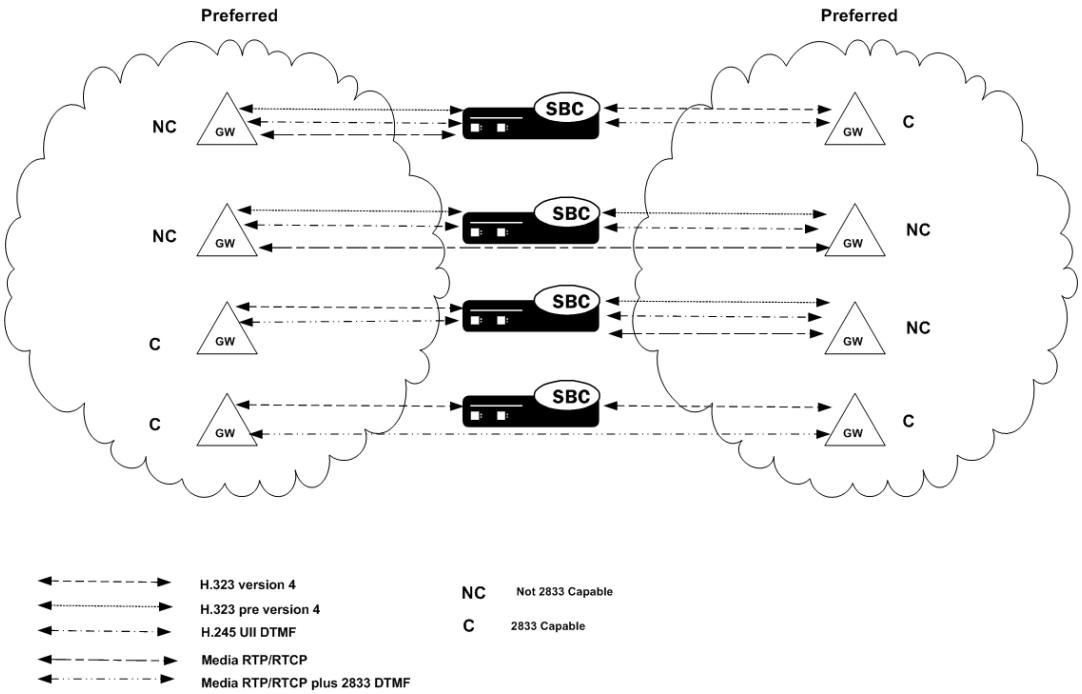
Preferred 2883 Support

If one side of the call, or a SIP interface, or a session agent, is configured for preferred 2833, the Oracle Communications Session Border Controller only generates and detects 2833 for endpoints if they signal support for 2833. The Oracle Communications Session Border Controller will offer 2833 in the TCS SDP, even if the originating caller did not.

- When the Oracle Communications Session Border Controller manages calls originating from a preferred source going to a preferred target, it:
  Performs 2833 translation for an endpoint when the originating side requests 2833 but the target does not negotiate 2833

  Allows 2833 to pass through if the originating side and target of the call are configured as preferred and negotiate 2833
- When the Oracle Communications Session Border Controller manages calls originating from a preferred source going to a transparent target, it:
  Performs 2833 translation when the originating side requests 2833 but the target is configured as transparent and does not negotiate 2833.

  Allows 2833 to pass through if the originating side and the target of the call are configured as transparent and negotiate 2833. The Oracle Communications Session Border Controller does not perform active translation because both ends support 2833.



If one SIP endpoint does not signal 2833 capability, but the other SIP or H.323 endpoints do, the Oracle Communications Session Border Controller does not perform 2833 translation.

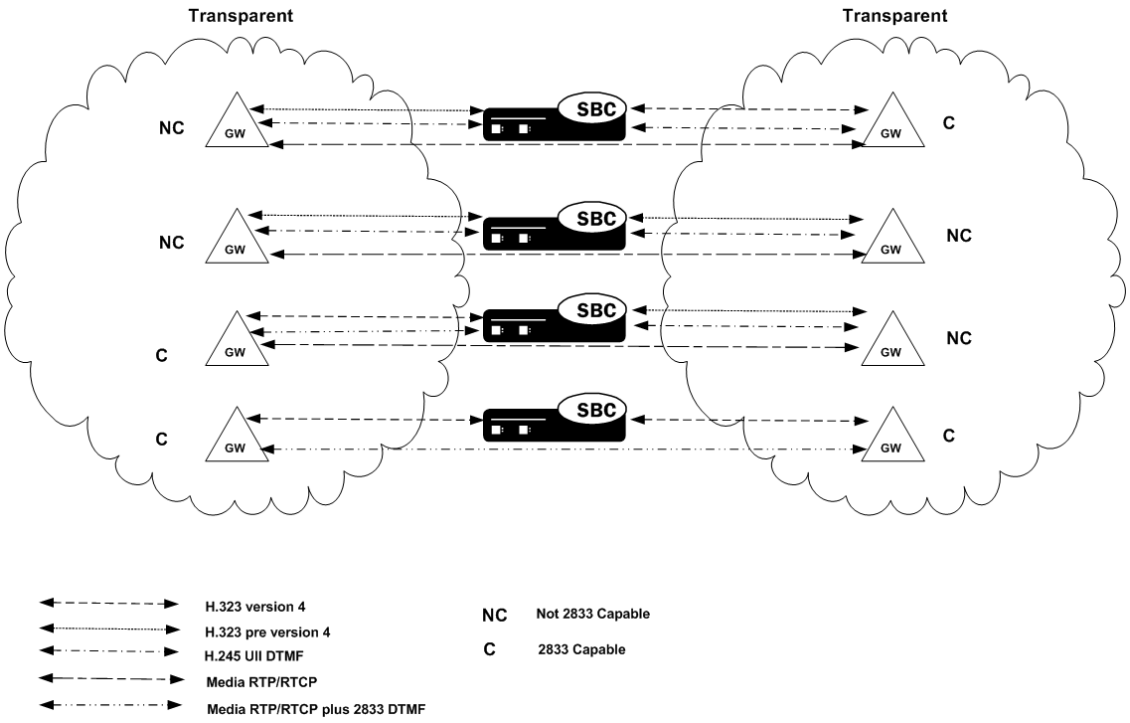
Transparent 2833 Support

The default configuration of the Oracle Communications Session Border Controller for H.323 is transparent 2833. The Oracle Communications Session Border Controller passes on the offered capabilities to the next-hop signaling element. If the next-hop endpoint is for a transparent 2833 target, typical capability negotiation determines the DTMF method. The

Oracle Communications Session Border Controller transparently relays the DTMF as it has in previous releases.

With transparent 2833, the Oracle Communications Session Border Controller acts as a typical B2BUA or B2BGW/GK. However when the target of the call is configured as preferred 2833, the Oracle Communications Session Border Controller:

- Relays the 2833 packets if the originating endpoint signals 2833 and the next-hop endpoint for the preferred target signals 2833
- Performs 2833 translation if the originating endpoint does not signal 2833 and the next-hop endpoint for the preferred target does signal 2833
- Does not perform 2833 translation or transparently relay 2833 if the originating endpoint signals 2833 and the next-hop endpoint for the preferred target (or even a transparent 2833 target) does not signal 2833.



Payload Type Handling

The Oracle Communications Session Border Controller supports the RTP NTE for telephony events such as transport of DTMF digits and hook flash. Using RTP NTE, endpoints perform per-call negotiation of the DTMF transfer method and negotiate payload type value for the RTP NTE packets.

Although most endpoints use payload type number 101, the RTP payload type formats can become asymmetrical when being interworked between SIP and H.323 because there is no default standard and endpoints use different types. This means that the payload type negotiated on one side of the Oracle Communications Session Border Controller, and that ends up being used for the call, might not be the same payload type negotiated on the other side of the Oracle Communications Session Border Controller. And while certain endpoints handle the asymmetry well, others do not.

Consider the simplified example of an IWF call initiated in SIP and translated to H.323. In this scenario, the SIP endpoint negotiates the payload type 106 with the Oracle Communications Session Border Controller. And despite the fact that the H.323 endpoint negotiates payload type 101, the Oracle Communications Session Border Controller returns type 106 and the call proceeds using type 106.

However, you can enable forced symmetric payload type handling so the Oracle Communications Session Border Controller changes the payload type of RFC 2833 packets to avoid using asymmetrical payload types.

For H.323 session agents and H.323 interfaces (stacks), you can configure an option that forces symmetric payload type use. The Oracle Communications Session Border Controller can detect when the payload types negotiated by the SIP and H.323 endpoints are symmetrical and when they are not. When it detects asymmetrical payload type use, the Oracle Communications Session Border Controller forces the remote endpoint to use the RFC 2833 payload type you configure in the SIP interface.

Basic RFC 2833 Negotiation Support

If H.323, SIP, or session agents on either side of the call are configured for preferred 2833 support, the Oracle Communications Session Border Controller supports end-to-end signaled negotiation of DTMF on a call-by-call basis. If the calling party is not configured for preferred support but sends 2833, the Oracle Communications Session Border Controller sends 2833 to the next-hop called party. If the calling party sends H.245 signals or alphanumeric UII, the Oracle Communications Session Border Controller sends H.245 signals or alphanumeric UII to the next-hop called party (if it is an H.323 next-hop).

The Oracle Communications Session Border Controller also supports hop-by-hop negotiation of DTMF capability on a call-by-call basis, if the signaling protocols or session agents on either side of the call are configured for preferred 2833 support.

H.323 to H.323 Negotiation

The Oracle Communications Session Border Controller serves as the H.323 called gateway. It answers RFC 2833 audio telephony event capability in the version 4 H.323/H.245 TCS when it receives a call from an H.323 endpoint configured for preferred RFC 2833.

If the Oracle Communications Session Border Controller is the answering device, configured for preferred support, and the calling device sends 2833, the Oracle Communications Session Border Controller accepts the 2833 regardless of the next-hop's DTMF capabilities. The received dynamic RTP payload type is used for detecting 2833 packets, while the response dynamic payload type is used for generating 2833 packets.

The Oracle Communications Session Border Controller supports:

- RFC-2833 audio telephony events in the version 4 H.323/H.245 TCS as the H.323 calling gateway, when the Oracle Communications Session Border Controller calls an H.323 endpoint configured for preferred RFC 2833 support. The Oracle Communications Session Border Controller sends 2833 to the called party regardless of whether the calling party sends it.
- H.245 UII and RFC-2833 packets sent at the same time, to the same endpoint, even if only half of the call is being provided 2833 support by the Oracle Communications Session Border Controller.
  If one half of the call supports H.245 UII, and the other half is being provided 2833 translation by the Oracle Communications Session Border Controller, the Oracle Communications Session Border Controller can also forward the H.245 UII it receives to the 2833 endpoint. For example, when the signaling goes through a gatekeeper or third

party call control, sending the H.245 UII in the signaling path allows those devices to learn the DTMF digits pressed.

Signal and Alpha Type Support

The Oracle Communications Session Border Controller supports:

- H.245 signal and alpha type UII in the H.323/H.245 TCS as the H.323 calling gateway when the:
  Oracle Communications Session Border Controller calls an H.323 endpoint configured for transparent 2833 support

  calling endpoint's target is configured as preferred

  If the originating preferred side also sends 2833, the Oracle Communications Session Border Controller forwards it to the transparent side. The Oracle Communications Session Border Controller sends signal and alpha UII support to the called party regardless of whether the calling party sends it, if the call originates from a preferred side to a transparent side.
- H.245 alphanumeric UII for DTMF for H.323 endpoints that do not signal 2833 or contain explicit H.245 UII capability, for stacks configured for transparent 2833 support.
  When the other half of the call is an H.323 endpoint of a stack configured for preferred 2833, the Oracle Communications Session Border Controller translates incoming H.245 UII on the transparent side, to 2833 packets on the preferred side, and vice versa. If the other half of the call is an H.323 endpoint of a transparent stack, the Oracle Communications Session Border Controller relays the H.245 UII messages.
- H.245 signal type UII for DTMF for H.323 endpoints that do not signal 2833, but do signal explicit H.245 UII capability, for stacks configured for transparent 2833 support.
  When the other half of the call is an H.323 endpoint of a stack configured for preferred 2833, the Oracle Communications Session Border Controller translates incoming H.245 signaled UII on the transparent side, to 2833 packets on the preferred side, and vice versa. If the other half of the call is an H.323 endpoint of a transparent stack, the Oracle Communications Session Border Controller relays the H.245 UII messages if both sides support it.

H.323 to SIP Calls

This section explains DTMF interworking specific to H.323 to SIP calls.

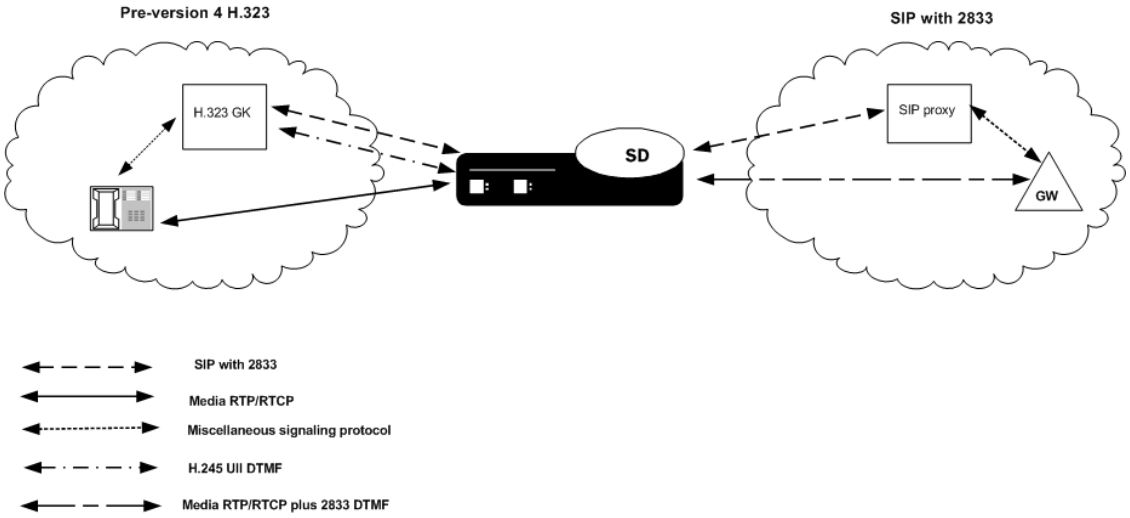
SIP Endpoints

SIP endpoints include those that support:

- RFC 2833
- SIP INFO method

H.323 Non-2833 Interworking with SIP

RFC 2833 and the SIP INFO method can be used for conveying DTMF information for SIP based-services. (RFC 2833 is the most widely used.) To provide end-to-end DTMF for SIP devices supporting RFC-2833 interworking with H.323 devices that do not, an RFC 2833 to H.323 UII interworking function is provided.



How H.323 to SIP Calls Work

For H.323 to SIP IWF calls, if 2833-related information is to be sent in the INVITE, the SIP interface of the SIP session agent has to be configured with the **rfc2833-mode** parameter set to preferred.

The following example shows an INVITE without 2833 in the SDP:

```
Apr  5 04:28:50.073 On 127.0.0.1:5070 sent to 127.0.0.1:5060
INVITE sip:780@192.168.200.6:5060 SIP/2.0
Via: SIP/2.0/UDP
127.0.0.1:5070;branch=z9hG4bKIWF0000gl2018604agg71c0;acme_irealm=external;acme
_sa=192.168.1.6
Contact: "jdoe"<sip:127.0.0.1:5070>
GenericID: 114421133000000@000825010100
Supported: 100rel^M
From: "msmith"<sip:192.168.200.68:5060>;tag=000000ab00011940
To: <sip:780@192.168.200.6:5060>
Call-ID: 7f00000113ce000000ab000101d0@127.0.0.1
CSeq: 2 INVITE
Content-Length: 225
Content-Type: application/sdp
v=0
o=IWF 3 3 IN IP4 192.168.1.6
s=H323 Call
c=IN IP4 192.168.1.6
t=0 0
m=audio 5214 RTP/AVP 0  18
a=rtpmap:0 PCMU/8000/1
a=rtpmap:18 G729/8000/1
a=fmtp:18 annexb=no
m=video 5216 RTP/AVP 31
a=rtpmap:31 H261/9000/1
```

SIP INFO—RFC 2833 Conversion

The Oracle Communications Session Border Controller can perform SIP INFO—RFC 2833 conversion. The Oracle Communications Session Border Controller also provides a way for you to enable a dual conversion mode, where the Oracle Communications Session Border Controller:

- Inserts telephone-event in the SDP offer
- Generates both RFC 2833 event packets and SIP INFO messages regardless of whether or not the SDP offer indicates RFC 2833

You can enable this feature either for SIP interfaces or session agents. The following apply:

- If the next hop SIP interface or session agent's **rfc2833-mode** is set to preferred, then the SD inserts RFC 2833 into the SDP offer/answer. This occurs regardless of whether:
  - The original SDP on the opposite side of the call does not support RFC 2833
  - The opposite side's SIP interface or session agent is set to transparent mode
- If the next hop SIP interface or session agent is set to transparent, then the behavior of the Oracle Communications Session Border Controller depends on the previous hop.
  - If the previous hop is a SIP interface or session agent configured for transparent mode, then the S Oracle Communications Session Border Controllerdoes not perform any conversion.
  - If the previous hop is a SIP interface or session agent configured for preferred mode, the Oracle Communications Session Border Controller does not insert RFC-2833 into the SDP on the transparent side. It does, however, translate from RFC 2833 to SIP INFO if the originating endpoint supports RFC 2833.

IPv6 SIP INFO to RFC 2833 Telephone Event Interworking

The Oracle Communications Session Border Controller can interwork SIP INFO and RFC Telephone Event messages for IPv4, IPv6—or for any session requiring interworking between IPv4 and IPv6. Other than installing the applicable lisences on your Oracle Communications Session Border Controller and enabling IPv6 support in your system configuration (**system-config**), you do not have to perform any configuration steps for this capability to work.

RFC 2833 Interworking Configuration

This section explains how to configure RFC 2833 to H.245 User Input Indication (UII) or SIP INFO method interworking.

RFC 2833 Mode for H.323 Stacks

To configure RFC 2833 mode for H.323 stacks:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

```
ORACLE(session-router)# h323
```

4. Type **h323-stacks** and press Enter.

```
ORACLE(h323)# h323-stacks
ORACLE(h323-stack)#
```

From this point, you can configure H.323 stack parameters. To view all H.323 stack parameters, enter a **?** at the system prompt.

5. **rfc2833-mode**—Set the RFC2833 mode. The default value is **transparent**. Valid values are:
   - **transparent**—The Oracle Communications Session Border Controller and H.323 stack behave exactly the same way as before and the 2833 or UII negotiation is transparent to the Oracle Communications Session Border Controller.
   - **preferred**—The H.323 stack uses 2833 for DTMF transfer, which it signals in its TCS. However, the remote H323 endpoint makes the decision. If the endpoint supports 2833, 2833 is used. If not, the H.323 stack reverts back to using UII. You configure the payload format by configuring the h323-config element.

RFC 2833 Payload for H.323

To configure the RFC 2833 payload in preferred mode:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **h323** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# h323
```

From this point, you can configure global H.323 parameters. To view all H.323 configuration parameters, enter a **?** at the system prompt.

4. **rfc2833-payload**—Enter a number that indicates the payload type the Oracle Communications Session Border Controller will use for RFC 2833 packets while interworking 2833 and UII. The default value is **101**.
   - Minimum—96
   - Maximum—127

Configuring the SIP Interface

You configure the 2833 mode and payload for the SIP interface. You must configure the payload the Oracle Communications Session Border Controllerwill use for RFC 2833 packets, while interworking 2833 and INFO/UII.

To configure the SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. **rfc2833-payload**—Enter a number that indicates the payload type the SIP interface will use for RFC 2833 packets while interworking 2833 and UII. The default value is **101**.The valid range is:
   - Minimum—96
   - Maximum—127

5. **rfc2833-mode**—Set the RFC 2833 mode for the SIP interface. The default value is **transparent**. Valid values are:
   - **transparent**—The SIP INFO and RFC 2833 translation is transparent to the Oracle Communications Session Border Controller.
   - **preferred**—The RFC 2833 transfer method is the preferred method for sending DTMF, and a telephone event is inserted in the SDP of the outgoing offer. The actual method of transfer, however, depends on the SDP offer/answer exchange that occurs between the Oracle Communications Session Border Controller and remote endpoint. If the remote endpoint supports RFC 2833, the Oracle Communications Session Border Controller performs SIP INFO—RFC 2833 conversion.
   - **dual**—The Oracle Communications Session Border Controller behaves the same as it does when set to preferred mode, and it forwards both the original DTMF mechanism and the translated one to the remote endpoint.

Configuring Session Agents

You configure session agents with:

- payload type the Oracle Communications Session Border Controller wants to use for RFC 2833 packets while interworking 2833 and UII.
  The default value for this attribute is 0. When this value is zero, the global rfc2833-payload configured in the h323-configuration element will be used instead. For SIP session agents, the payload defined in the SIP interface is used, if the SIP interface is configured with the preferred RFC 2833 mode.
- 2833 mode
  A value of transparent or preferred for the session agent's 2833 mode will override any configuration in the h323-stack configuration element.

  To configure session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# session-router
```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. **rfc2833-mode**—Set the RFC 2833 mode you want the session agent to use. The default value is **none**. Valid values are:
   - **none**—RFC 2833 to UII interworking is based on the H.323 stack configuration.
   - **transparent**—The RFC 2833 or UII negotiation is transparent to the Oracle Communications Session Border Controller. This overrides the H.323 stack configuration, even if the stack is configured for preferred mode.
   - **preferred**—RFC 2833 for DTMF transfer is preferred, which is signaled in the TCS. If the endpoint supports 2833, 2833 is used. If not, the H.323 stack configured as preferred will revert back to using UII. This overrides any configuration in the h323-stack even if the stack is configured for transparent mode.

     For SIP INFO—RFC 2833 conversion, you can choose:
   - **none**—The 2833-SIP INFO interworking will be decided based on the sip-interface configuration.
   - **transparent**—The session agent behaves the same as it did without the SIP INFO—RFC 2833 conversion feature. The SIP INFO and RFC 2833 translation is transparent to the Oracle Communications Session Border Controller.
   - **preferred**—The RFC 2833 transfer method is the preferred method for sending DTMF, and a telephone event is inserted in the SDP of the outgoing offer. The actual method of transfer, however, depends on the SDP offer/answer exchange that occurs between the Oracle Communications Session Border Controller and remote endpoint. If the remote endpoint supports RFC 2833, the Oracle Communications Session Border Controller performs SIP INFO—RFC 2833 conversion.
   - **dual**—The Oracle Communications Session Border Controller behaves the same as it does when set to preferred mode, and it forwards both the original DTMF mechanism and the translated one to the remote endpoint.
5. **rfc2833-payload**—Enter a number that indicates the payload type the session agent will use for RFC 2833 packets while interworking 2833 and UII. The default value is 0. The valid range is:
   - Minimum—0, 96
   - Maximum—127

Enabling Payload Type Handling

You can configure H.323 session agents and H.323 interfaces (stacks) with an option that forces symmetric payload type use. For Payload Type Handling to work properly, you must set the following SIP interface and the global H.323 configuration parameters with these values:

- **rfc2833-mode**—Set this parameter to **preferred**; the default is **transparent**.
- **rfc2833-payload**—Set this parameter to the payload type you want forced for the remote endpoint. Your entry will be between **96** and **127**, with **101** as the default.
To enable forced symmetric payload type handling for an H.323 session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-agent** and press Enter.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

If you want to add this option to a pre-existing H.323 session agent, select the one you want to edit.

4. **options**—Set the options parameter by typing options, a Space, the option name **Map2833ForceRemotePT** with a plus sign in front of it. Then press Enter.

```
ORACLE(session-agent)# options +Map2833ForceRemotePT
```

If you type options and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

To enable forced symmetric payload type handling for an H.323 interface:

6. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

7. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

8. Type **h323-config** and press Enter.

```
ORACLE(session-router)# h323-config
ORACLE(h323-config)#
```

9. Type **h323-stacks** and press Enter.

```
ORACLE(h323-config)# h323-stacks
ORACLE(h323-stack)#
```

10. **options**—Set the options parameter by typing options, a Space, the option name **Map2833ForceRemotePT** with a plus sign in front of it. Then press Enter.

```
ORACLE(h323-stack)# options +Map2833ForceRemotePT
```

    If you type **options** and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

11. Save and activate your configuration.

DTMF Transparency for IWF

In certain vendors' implementations of DTMF during SIP/H.323 IWF, there have been discrepancies between the RFC 2833 and UII/INFO negotiations and what type of messages actually get sent. Instead of correcting these errors on its own end, the Oracle Communications Session Border Controller has perpetuated these inaccuracies.

To ensure that the Oracle Communications Session Border Controller always sends the correctly negotiated protocols, a **media-manager-config** parameter called **translate-non-rfc2833-event** has been created. When **translate-non-rfc2833-event** is enabled, the Oracle Communications Session Border Controller always sends the type of messages that were initially negotiated, regardless of the type of messages it may be receiving.

DTMF Transparency Configuration

To enable DTMF transparency for SIP/H.323 IWF:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **media-manager** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

4. **translate-non-rfc2833-event**—To enable this feature, set this parameter to **enabled**. If you do not want to use the feature leave it set to its default behavior, disabled.
5. Save and activate your configuration.

RFC 2833 Packet Sequencing

You can configure your Oracle Communications Session Border Controller to generate either the entire start-interim-end RFC 2833 packet sequence or only the last three end 2833 packets for non-signaled digit events.

RFC 2833 Packet Sequencing Configuration

To send only the last three end 2833 packets for non-signaled digits events:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-profile** and press Enter.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **rfc2833-end-pkts-only-for-non-sig**—By default, this parameter is enabled—meaning that only the last three end 2833 packets are used for non-signaled digits events. Set this parameter to disabled if you want the entire start-interim-end RFC 2833 packet sequence for non-signaled digit events
5. Save and activate your configuration.

SIP Tel URI Support

The Oracle Communications Session Border Controller maps H.323 addresses to either SIP URIs or Tel URIs. You can configure the Oracle Communications Session Border Controller to include Tel URIs in the following SIP headers for calls that require the IWF:

- Request Line
- To
- From

When Tel URI support is not used on a Oracle Communications Session Border Controller performing IWF translations, the SIP INVITE is formatted like it is in the following example. This example uses 192.168.5.5 as the external proxy address, or the next hop (as configured in the local policy).

```
INVITE sip:602@192.168.5.5:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.5.58:5060;branch=z9hG4bKIWF0aqoqg001g11a7kos4g0
Contact: <sip:603@192.168.5.58:5060>
From: <sip:603@192.168.5.58:5060>;tag=4069ac210018a0
To: <sip:602@192.168.5.5:5060>
```

In the example above, the session needs to be routed to another SIP proxy that can resolve an E.164 number to a SIP address. However, the next SIP proxy must be informed that the message will be routed based on the included E.164 number; the SIP address of the Request URI does not have a routable SIP address. To devise a routable address, the Request URI must be reconstructed as a Tel URI.

Without Tel URI support configured, the terminating SIP user would be required to have an address of 602@192.168.5.5, where the IPv4 address portion is the same as the address for the proxy. If it were not the same, then the session would terminate at the proxy. However, the proxy would be unable to handle the session because the SIP address it received would be unknown/unroutable.

Because it is not desirable to have an IPv4 address be the user-identity and rely on the configuration of the IP network, the SIP INVITE generated by the Oracle Communications Session Border Controller and sent to the proxy must have the following format if it is sent to an H.323 entity.

```
INVITE tel:2345 SIP/2.0
Via: SIP/2.0/UDP 192.168.5.52:5060;branch=z9hG4bKIWFaqoqq00cobgf9so10o0
Contact: <sip:1234@192.168.5.58:5060>
From: <tel:1234>;tag=4069ac35000c5ff8
To: <tel:2345>
Call-ID:7f0000113ce4069ac35000c5440
CSeq: 1 INVITE
Content-Length: 155
Content-Type: application/sdp
```

SIP Interface Configuration

You enable this feature in the SIP interface configuration.

To configure SIP Tel URI support for calls that require the IWF:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ```

   From this point, you can configure SIP interface parameters. To view see all SIP interface parameters, enter a **?** at the system prompt.

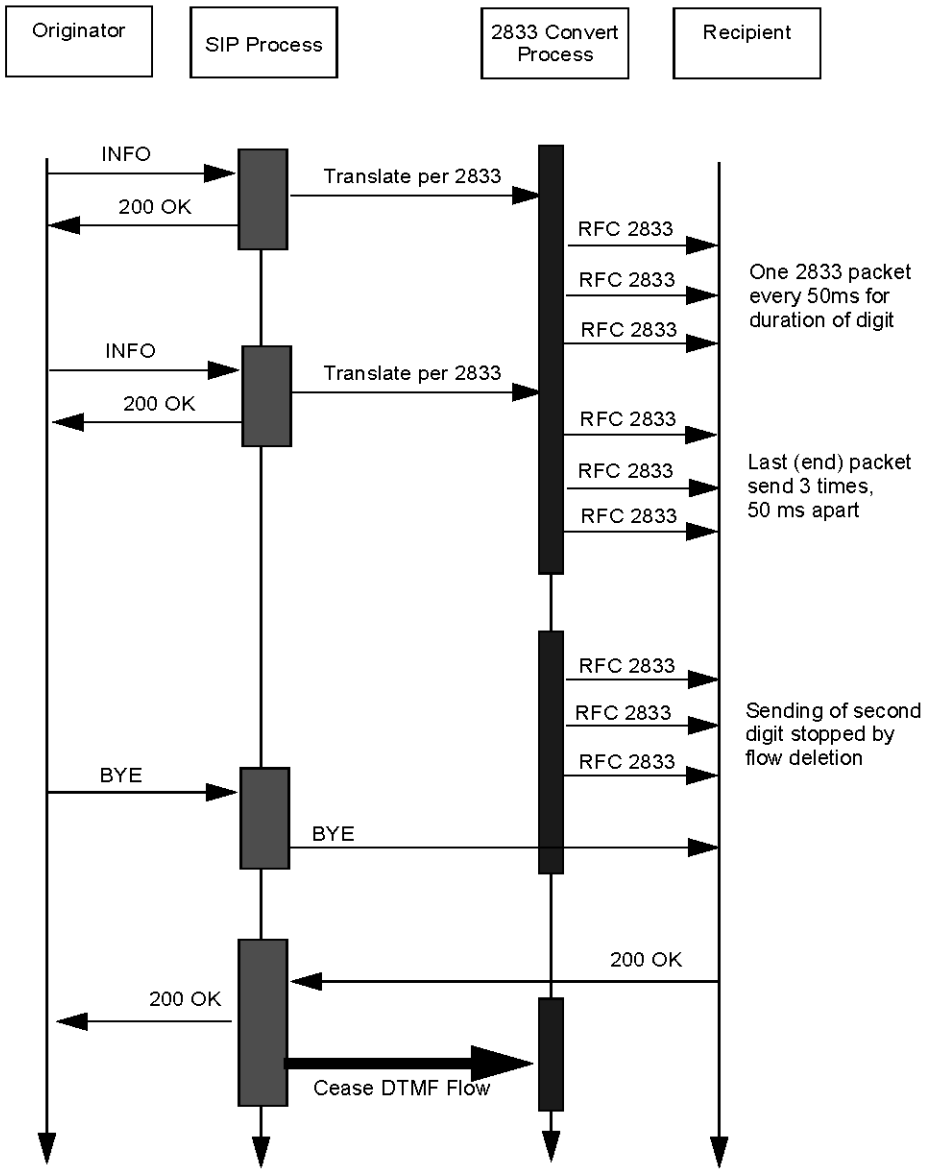
4. **teluri-scheme—Enable or disable the conversion of SIP URIs to Tel URIs**. The default value is **disabled**. Valid values are:

   - enabled | disabled

   ```
   ORACLE(sip-interface)# teluri-scheme enabled
   ORACLE(sip-interface)# done
   ```

Graceful DTMF Conversion Call Processing

The default implementation for SIP INFO to RFC2833 events performs well in most network topologies is shown below.



When the Oracle Communications Session Border Controller receives an INFO DTMF request, the SIP process determines whether or not it needs to perform DTMF-to-RFC2833 translation. If translation is required, the process forwards the DTMF to the 2833 convert process for translation and transmission to the recipient. Immediately after off-loading the DTMF, the SIP process sends a 200 OK response for the INFO. As shown in the figure, the 2833 convert process generates a number of RFC2833 packets to represent received DTMF digits.

Specifically, the 2833 convert process generates one RFC 2833 packet every 50 milliseconds for the duration of the DTMF digit, whose length is specified in the INFO request, and two retransmits of the last packet (known as the end packet) 50 milliseconds apart.

Consequently, the time interval between the 200 OK and the actual transmission of the RFC 2833 translation is the sum of the DTMF duration and 100 ms.

> **Note:**
>
> This time interval can be shortened to 100 ms by enabling the rfc2833-end-pkts-only-for-non-sig parameter in media-manger which results in Oracle Communications Session Border Controller only generating the last packet and its two retransmits.

The early 200 OK allows the endpoint to send the next DTMF digit before the SD has sent all the RFC2833 packets, resulting in the next digit being queued internally by the 2833 convert process before being sent.

A problem arises if the SIP process receives a BYE request from the DTMF originator while queued digits are awaiting translation and transmission. In such an event, the SIP process immediately forwards the BYE request to the recipient, ending the session with DTMF digits awaiting translation and transmission.

An alternative DTMF conversion model provides for a feedback mechanism from the 2833 convert process to the SIP process. With this model enabled, the SIP process buffers a received BYE until it obtains confirmation that all queued DTMF digits have been translated and transmitted. Only after obtaining confirmation, does it forward the BYE to terminate the session.

This processing model is enable by a SIP option, **sync-bye-and-2833**, and requires that **rfc2833-mode** parameter on the egress interfaces is NOT set to dual, any value other than dual, is supported.

1. From superuser mode, use the following command sequence to access sip-config configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. Use the SIP option **sync-bye-and-2833** to delay BYE processing until DTMF-to-RFC2833 translation has been completed.

```
ORACLE(sip-config)# options +sync-bye-and-2833="enabled"
ORACLE(sip-config)#
```

3. Use the **done** and **exit** commands to complete configuration.

```
ORACLE(sip-config)# done
ORACLE(sip-config)# exit
ORACLE(session-router)#
```

IWF Inband Tone Option

This option enables the Oracle Communications Session Border Controller to send a progress indicator (PI) =8 in an H.225 message when an SDP is received in a provisional message. In effect, this option sends network announcements inband. It is also applicable because in some networks H.323 endpoints support early H.245.

The H.323 inband tone option is enabled by adding the **inbandTone** as an option in a configured H.323 stack.

When this option is not used, the ringtone is generated locally (NO PI=8 in PROGRESS OR ALERTING) is the default behavior.

IWF Inband Tone Configuration

To configure the IWF inband tone option:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stacks)#
   ```

5. Use the ACLI **select** command add this feature to an existing H.323 interface.

   ```
   ORACLE(h323-stacks)# select
   ```

6. If you are adding this service to a new H.323 interface, type **option inbandTone** and press Enter.

   ORACLE(h323-stacks)# **option inbandTone**

7. If you are adding this service to an H.323 interface that already exists, type **select** to select the interface to which you want to add the service. Then use the options command and prepend the option with a plus (+) sign.
   - If you know the same of the interface, you can type the name of the interface at the name: prompt and press Enter.
   - If you do not know the name of the interface, press Enter at the name: prompt. A list of interfaces will appear. Type the number corresponding to the interface you want to modify, and press Enter.
   - **If are adding service to an existing interface and type in the option without a plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign: options +inbandTone.**

RFC 3326 Support

This section explains the Oracle Communications Session Border Controller's ability to map Q.850 cause values with SIP responses for calls that require IWF.

RFC 3326 defines a header that might be included in any in-dialogue request. This reason header includes cause values that are defined as either a SIP response code or ITU-T Q.850 cause values. You can configure the Oracle Communications Session Border Controller to support sending and receiving RFC 3326 in SIP messages for:

- Mapping H.323 Q.850 cause values to SIP responses with reason header and cause value
- Mapping SIP response messages and RFC 3326 reason header and cause
- Locally generated SIP response with RFC 3326 reason header and cause

As specified in RFC 3326, the Oracle Communications Session Border Controller sends SIP responses to the softswitch that contain the received Q.850 cause code and the reason.

Though the Oracle Communications Session Border Controller can generate RFC 3326 headers, the default behavior for this feature is disabled. Furthermore, the Oracle Communications Session Border Controller can receive and pass SIP error messages (4xx, 5xx, and 6xx) that contain the SIP reason header with a Q.850 cause code and reason (as specified in RFC 3326). If the Oracle Communications Session Border Controller receives an error message without the Reason header, then the Oracle Communications Session Border Controller is not required to insert one.

In calls that require IWF, the Q.850 cause generated in the SIP response are the same as the cause received in the following H.225 messages: Disconnect, Progress, Release, Release Complete, Resume Reject, Status, and Suspend Reject. In addition, the Q.850 cause codes that the Oracle Communications Session Border Controller receives in RFC 3326 headers are passed to the H.323 part of the call unmodified; the H.323 call leg uses this cause code for releasing the call.

For interworking calls between SIP and H.323, you can configure:

- Mappings for SIP status codes to Q.850 values
- Mappings for particular Q.850 cause codes to SIP status codes

If it cannot find the appropriate mapping, then the Oracle Communications Session Border Controller uses default mappings defined in the Default Mappings table below.

The following describes how the Oracle Communications Session Border Controller handles different IWF call scenarios:

- SIP request containing a Reason header—When it receives a request containing a Reason header, the Oracle Communications Session Border Controller determines if the request is a SIP BYE or SIP CANCEL message. RFC 3326 states that the Reason header is mainly used for these types of requests. If there is a Reason header and it contains the Q.850 cause value, then the Oracle Communications Session Border Controller releases the call on the H.323 side using the specified cause value.
- SIP response—When it receives the error response to an initial SIP INVITE, the Oracle Communications Session Border Controller uses its SIP-Q.850 map to determine the Q.850 that it will use to release the call. If there is not a map entry, then the Oracle Communications Session Border Controller uses the default mappings shown in the Default Mappings table.
- Active call released from the H.323 side—If an active call is released from the H.323 side, the Oracle Communications Session Border Controller checks the outgoing realm (the SIP side) to see if the addition of the Reason header is enabled. If it is, then the Oracle Communications Session Border Controller adds the Reason header in the SIP BYE request with the Q.850 value it received from the H.323 side.

- Error during setup of the call on the H.323 side—In the event of an error during setup on the H.323 side of the call, the Oracle Communications Session Border Controller needs to send:
  - An error response, if this is a SIP to H.323 call
  - A SIP CANCEL, if this is a H.323 to SIP call and the H.323 side hangs up before the call is answered on the SIP side
    In this case, the Oracle Communications Session Border Controller checks to see if adding the Reason header is enabled in the IWF configuration. If it is, then the Oracle Communications Session Border Controller adds the Reason header with the Q.850 cause value it received from the H.323 side.
- Call released due to a Oracle Communications Session Border Controller error—If the call is released due a Oracle Communications Session Border Controller error and adding the Reason header is enabled in the IWF configuration, the error response to the initial INVITE contains the Reason header. The Oracle Communications Session Border Controller checks the SIP to Q.850 map configurations to determine whether or not the SIP error response code it is generating is configured. If it is, then the system maps according to the configuration. If if it not, the Oracle Communications Session Border Controller derives cause mapping from the default table.

Like the configuration for SIP-only calls that enable this feature, you can set a parameter in the IWF configuration that enables adding the Reason header in the SIP requests or responses.

Default Mappings

This table defines the default mappings the Oracle Communications Session Border Controller uses when it cannot locate an appropriate entry that you have configured.

Q.850 Cause Value		SIP Status	
1	Unallocated number	404	Not found
2	No route to specified transit network	404	Not found
3	No route destination	404	Not found

Q.850 Cause Value		SIP Status	
16	Normal calling clearing	N/A	BYE message

Note:

A call clearing BYE message containing cause value 16 typically results in the sending of a SIP BYE or CANCEL request. However, if a SIP response is to be sent to the INVITE request, the default response code

Q.850 Cause Value		SIP Status	
			is used .
17	User busy	486	Busy here
18	No user responding	408	Request timeout
19	No answer from the user	480	Temporarily unavailable
20	Subscriber absent	480	Temporarily unavailable
21	Call rejected	603	Decline (if location filed in Cause information element indicates user; otherwise 403 Forbidden is used)
22	Number changed	301	Moved permanently (if information in diagnostic field of Cause information element is suitable for generating SIP Contact header; otherwise 410 Gone is used)
23	Redirection to new destination	410	Gone
25	Exchange routing error	483	Too many hops
27	Destination out of order	502	Bad gateway
28	Address incomplete	484	Address incomplete
29	Facility rejected	501	Not implemented
31	Normal, unspecified	480	Temporarily unavailable
34	No circuit, channel unavailable	503	Service unavailable
38	Network out of order	503	Service unavailable
41	Temporary failure	503	Service unavailable
42	Switching equipment congestion	503	Service unavailable
47	Resource unavailable unspecified	503	Service unavailable
55	Incoming calls barred with CUG	403	Forbidden
57	Bearer capability not authorized	403	Forbidden
58	Bearer capability not presently available	503	Service unavailable
65	Bearer capability not implemented	488	Not acceptable here
69	Requested facility not implemented	501	Not implemented
70	Only restricted digital information available	488	Not acceptable here
79	Service or option not implemented, unspecified	501	Not implemented
87	User not member of CUG	403	Forbidden
88	Incompatible destination	503	Service unavailable
102	Recovery on timer expiry	504	Server time-out

RFC 3326 Support Configuration

To configure a SIP status to Q.850 Reason with cause mapping:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-q850-map** and press Enter.

   ```
   ORACLE(session-router)# sip-q850-map
   ORACLE(sip-q850-map)#
   ```

4. Type **entries** and press Enter.

   ```
   ORACLE(sip-q850-map)# entries
   ORACLE(sip-q850-map-entry)#
   ```

   From here, you can view the entire menu for the SIP status to Q.850 Reason with cause mapping entries configuration by typing a **?**.

5. **sip-status**—Set the SIP response code that you want to map to a particular Q.850 cause code and reason. There is no default, and the valid range for values is:
   - Minimum—100
   - Maximum—699
6. **q850-cause**—Set the Q.850 cause code that you want to map to the SIP response code that you set in step 5. There is no default.
   - Minimum—0
   - Maximum—2147483647
7. **q850-reason**—Set the Q.850 reason corresponding to the Q.850 cause code that you set in step 6. There is no default. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
8. Repeat this process to create the number of local response map entries that you need.
9. Save and activate your configuration for changes to take effect.

   To configure a Q.850 cause to a SIP status with reason mapping:

10. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ```

11. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ```

12. Type **sip-q850-map** and press Enter.

```
ORACLE(session-router)# q850-sip-map
ORACLE(q850-sip-map)#
```

13. Type **entries** and press Enter.

```
ORACLE(q850-sip-map)# entries
ORACLE(q850-sip-map-entry)#
```

From here, you can view the entire menu for the Q.850 cause to a SIP response code with reason mapping entries configuration by typing a **?**.

14. **q850-cause**—Set the Q.850 cause code that you want to map to a SIP status with reason. There is no default.
    - Minimum—0
    - Maximum—2147483647
15. **sip-status**—Set the SIP response code to which you want to map the Q.850 cause that you set in step 5. There is no default, and the valid range for a value is
    - Minimum—100
    - Maximum—699
16. **sip-reason**—Set the reason that you want to use with the SIP response code that you specified in step 6. There is no default. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
17. Repeat this process to create the number of local response map entries that you need.

To enable the Oracle Communications Session Border Controller to add the Reason header for calls that require IWF:

18. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

19. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

20. Type **iwf-config** and press Enter.

```
ORACLE(session-router)# iwf-config
ORACLE(iwf-config)#
```

21. **add-reason-header**—Enable this parameter to add the Reason header to IWF calls. The default is **disabled**. Valid values are:
    - enabled | disabled

IWF Privacy Caller Privacy on Unsecure Networks

This feature enables bi-directional SIP/H.323 IWF support for CPID hiding by using the presentation indicators in the Calling Party Number information element for H.323 signaling, and RFC 3325-based privacy support for SIP signaling. It lets the Oracle Communications

Session Border Controller insert the P-Asserted-Identity and the Privacy header in the INVITE when the presentation indicator is set to restricted.

The presence, or absence, of P-Asserted-Identity and Privacy headers in the SIP INVITE informs the remote SIP proxy or endpoint to either block or advertise the CPID.

About the Presentation Indicator

When address information represents a telephone number, the relevant information can appear in the Calling Party Number information element (IE). This IE contains the caller's number, information about the number, and presentation and screening indicators found in octet 3a. In order to prevent a calling party number to be passed through, the presentation indicator parameter (octet 3a) in the Calling Party IE must be set to a value other than 00.

In a H.323 to SIP IWF call, octet 3a in the Q.931 message indicates the caller's preference for CPID restriction. If bits 7 and 6 are set to (0 1), the presentation is restricted and the outbound SIP INVITE from the IWF stack must be constructed as such.

H.323 to SIP IWF Call

When the presentation indicator in the calling party IE is set to restricted, the INVITE's From and Contact headers sent from to sipd will be modified according to RFC 3325. When the Oracle Communications Session Border Controller receives calls initiated as H.323, it will recognize the caller's presentation bits as defined in Q.931 and use that information to construct a SIP INVITE in accordance with the user's indicated preference.

- Inclusion of a P-Asserted-Identity header in the INVITE, containing the calling party's CPID and the Oracle Communications Session Border Controller's IP address, constructed as a SIP URI (same mechanism used to construct the From-URI today).
- Addition of a Privacy header with its value set to id. This addition indicates to the upstream proxies and gateways that the caller address is to be hidden.

The sipd will either proxy or strip these headers according to RFC 3325, depending on the SIP interface and SIP session agent configurations.

Example 1 SETUP Sent from h323d to Remote H.323 Endpoints

```
Q.931
Protocol discriminator: Q.931
Call reference value length: 2
Call reference flag: Message sent from originating side
Call reference value: 2F62
Message type: SETUP (0x05)
Bearer capability
Information element: Bearer capability
Length: 3
...0 1000 = Information transfer capability: Unrestricted digital information
(0x08)
.00. .... = Coding standard: ITU-T standardized coding (0x00)
1... .... = Extension indicator: last octet
...1 0011 = Information transfer rate: 384 kbit/s (0x13)
.00. .... = Transfer mode: Circuit mode (0x00)
1... .... = Extension indicator: last octet
...0 0101 = User information layer 1 protocol: Recommendation H.221 and H.242
(0x05)
```

```
1... .... = Extension indicator: last octet
Display  'jdoe\000'
Information element: Display
Length: 9
Display information: jdoe\000
Calling party number
Information element: Calling party number
Length: 2
.... 0000 = Numbering plan: Unknown (0x00)
.000 .... = Number type: Unknown (0x00)
0... .... = Extension indicator: information continues through the next octet
.... ..00 = Screening indicator: User-provided, not screened (0x00)
.01. .... = Presentation indicator: Presentation restricted (0x01)
1... .... = Extension indicator: last octet
```

Example 2 INVITE from h323d to sipd

The two new headers will be stripped by the sipd when the INVITE is sent to a untrusted SIP proxy or endpoint and will be proxied over to a trusted SIP proxy or end point.

```
INVITE sip:780@192.168.200.6:5060;acme_realm=internal SIP/2.0
Via: SIP/2.0/UDP
127.0.0.1:5070;branch=z9hG4bKIWF00000510d031s9kou5c0;acme_irealm=external
Contact: "Anonymous"<sip:anonymous@127.0.0.1:5070
GenericID: 7400000@000825010100
Supported: 100rel
From: "Anonymous"<sip:anonymous@anonymous.invalid>;tag=0000004a000d8cc0
To: <sip:780@192.168.200.6:5060
Call-ID: 7f00000113ce0000004a000d88d8@127.0.0.1
CSeq: 2 INVITE
P-Asserted-Identity: "jdoe"<sip:42343@192.168.200.68:5060>
Privacy: id
Content-Length: 175
Content-Type: application/sdp
v=0
o=IWF 3 3 IN IP4 192.168.1.6
s=H323 Call
c=IN IP4 192.168.1.6
t=0 0
m=audio 5666 RTP/AVP 0 101  18
a=rtpmap:0 PCMU/8000/1
a=rtpmap:101 telephone-event/8000/1
a=fmtp:101 0-15
a=rtpmap:18 G729/8000/1
a=fmtp:18 annexb=no
m=video 5668 RTP/AVP 31
a=rtpmap:31 H261/9000/1
```

SIP to H.323

For a SIP to H.323 call, the Oracle Communications Session Border Controller must recognize the caller's Privacy request and set the presentation bits accordingly when constructing the outbound RAS/SETUP message. It must check SIP calls for the Privacy header (with value set

to id). If this header is present, the SETUP's octet 3a's presentation bits must be set to restricted.

The Oracle Communications Session Border Controller does not support any other value for the Privacy header. For those calls, the SETUP will not include a presentation indicator.

Example INVITE from SIP End Point to sipd

```
Apr 21 08:50:38.786 On [0:0]192.168.200.68:5060 received from
192.168.200.6:5062
INVITE sip:800@192.168.200.68:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.6:5062
From: anonymous <sip:anonymous@192.168.200.6:5062>;tag=1
To: sut <sip:800@192.168.200.68:5060
P-Asserted-Identity: sipp <sip:7789@192.168.200.6:5062
Privacy: id
Call-ID: 1.1688.192.168.200.6@sipp.call.id
Cseq: 1 INVITE
Contact: sip:anonymous@192.168.200.6:5062
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 136
v=0
o=user1 53655765 2353687637 IN IP4 127.0.0.1
s=-
t=0 0
c=IN IP4       127.0.0.1
m=audio 10000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
Sample INVITE from sipd to h323d
Apr 21 08:50:38.807 On 127.0.0.1:5070 received from 127.0.0.1:5060
INVITE sip:800@127.0.0.1:5070;acme_sag=sag1;acme_irealm=internal SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5060;branch=z9hG4bK0804o700c0f0t9gpj0g0.1
From: anonymous <sip:anonymous@192.168.200.6:5062>;tag=SDm8kvc01-1
To: sut <sip:800@192.168.200.68:5060
P-Asserted-Identity: sipp <sip:7789@192.168.200.6:5062
Privacy: id
Call-ID: SDm8kvc01-083221d8c0fa33f71ae85dd6ed0e4ea4-06ahc21
Cseq: 1 INVITE
Contact: <sip:anonymous@192.168.200.68:5060;transport=udp
Max-Forwards: 69
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 136
GenericID: 9883100005@000825010100
v=0
o=user1 53655765 2353687637 IN IP4 127.0.0.1
s=-
t=0 0
c=IN IP4       127.0.0.1
m=audio 10000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
Sample SETUP sent from h323d to remote H323 EP
Q.931
    Protocol discriminator: Q.931
```

```
    Call reference value length: 2
    Call reference flag: Message sent from originating side
    Call reference value: 664D
    Message type: SETUP (0x05)
Bearer capability
        Information element: Bearer capability
        Length: 3
        ...1 0000 = Information transfer capability: 3.1 kHz audio (0x10)
        .00. .... = Coding standard: ITU-T standardized coding (0x00)
        1... .... = Extension indicator: last octet
        ...1 0000 = Information transfer rate: 64 kbit/s (0x10)
        .00. .... = Transfer mode: Circuit mode (0x00)
        1... .... = Extension indicator: last octet
        ...0 0011 = User information layer 1 protocol: Recommendation G.711 A-
law (0x03)
1... .... = Extension indicator: last octet
    Display  'anonymous'
        Information element: Display
        Length: 9
        Display information: anonymous
    Calling party number
        Information element: Calling party number
Length: 2
        .... 0000 = Numbering plan: Unknown (0x00)
        .000 .... = Number type: Unknown (0x00)
        0... .... = Extension indicator: information continues through the
next octet
        .... ..00 = Screening indicator: User-provided, not screened (0x00)
        .01. .... = Presentation indicator: Presentation restricted (0x01)
        1... .... = Extension indicator: last octet
```

IWF Privacy Caller Privacy on Secure Connections

In prior releases, when the H.323 endpoint sends a SETUP with presentation indicator set to allowed, the Oracle Communications Session Border Controller does not insert the P-Asserted-Identity in the INVITE. The SIP INVITE needs the P-Asserted-Identity header to support calling line identification presentation (CLIP) to calling line identification restriction (CLIR) in an IP multimedia subsystem (IMS) solution. This feature lets the Oracle Communications Session Border Controller insert the P-Asserted-Identity in the INVITE when the presentation indicator is set to allowed.

- CLIP is a service provided to the called party that allows the display of the calling number (caller ID). The user-provided calling number must be transported from the caller to the called party.
- CLIR is a service provided to the calling party that lets it indicate whether or not the calling number is to be displayed to the called party. It sets a calling number presentation indicator to allowed or restricted. Regulations require that network administrations remove the calling number before it is sent to the called party, if the calling party has so requested.

H.323 to SIP IWF

When the Oracle Communications Session Border Controller translates incoming H.323 messages to SIP on a secure connection (which means the Oracle Communications Session

Border Controller can rely on the data sent from the originator); it will translate the information in the H.323 messages into SIP messages as detailed in the following sections.

Calls with Presentation Allowed

When the Oracle Communications Session Border ControllerC receives a SETUP from the H.323 domain where presentation is allowed, it generates an INVITE to the SIP domain with the following header. (Presentation is allowed when the calling party's information element presentation indicator (octet 3a) equals 00.)

- P-Asserted-ID: the userpart should be derived from the Calling Party Number Information Element digits.

H.323 to SIP

When h323d receives a SETUP with the calling party's information element presentation indicator set to allowed, the Oracle Communications Session Border Controller will add the P-Asserted-Identity header to the INVITE. The P-Asserted-Identity is very similar to the FROM header, except for the tag.

Sample SETUP sent from h323d to Remote H323 Endpoints

```
Q.931
Protocol discriminator: Q.931
Call reference value length: 2
Call reference flag: Message sent from originating side
Call reference value: 2F62
Message type: SETUP (0x05)
Bearer capability
Information element: Bearer capability
Length: 3
...0 1000 = Information transfer capability: Unrestricted digital information
(0x08)
.00. .... = Coding standard: ITU-T standardized coding (0x00)
1... .... = Extension indicator: last octet
...1 0011 = Information transfer rate: 384 kbit/s (0x13)
.00. .... = Transfer mode: Circuit mode (0x00)
1... .... = Extension indicator: last octet
...0 0101 = User information layer 1 protocol: Recommendation H.221 and H.242
(0x05)
1... .... = Extension indicator: last octet
Display  'jdoe\000'
Information element: Display
Length: 9
Display information: jdoe\000
Calling party number: '42343'
Information element: Calling party number
Length: 6
.... 1001 = Numbering plan: Private numbering (0x09)
.110 .... = Number type: Abbreviated number (0x06)
0... .... = Extension indicator: information continues through the next octet
.... ..00 = Screening indicator: User-provided, not screened (0x00)
.00. .... = Presentation indicator: Presentation allowed (0x00)
```

```
1... .... = Extension indicator: last octet
Calling party number digits: 42343
```

SIP to H.323

When the sipd receives an INVITE with the P-Asserted-Identity header but without the Privacy header, the Oracle Communications Session Border Controller will set the presentation indicator to allowed in H.323's SETUP.

When the Privacy header is present with the value id, the presentation indicator will be set to restricted. The Oracle Communications Session Border Controller does not support any other value for the Privacy header and so for those call flows, the presentation indicator will be absent in the SETUP.

Example 1 INVITE from sip EP to sipd

```
Apr 20 04:43:54.220 On [0:0]192.168.200.68:5060 received from
192.168.200.6:5062
INVITE sip:800@192.168.200.68:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.6:5062
From: sipp <sip:7789@192.168.200.6:5062>;tag=1
To: sut <sip:800@192.168.200.68:5060>
P-Asserted-Identity: sipp <sip:7789@192.168.200.6:5062>
Call-ID: 1.1336.192.168.200.6@sipp.call.id
Cseq: 1 INVITE
Contact: sip:7789@192.168.200.6:5062
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 136
^M
v=0
o=user1 53655765 2353687637 IN IP4 127.0.0.1
s=-
t=0 0
c=IN IP4       127.0.0.1
m=audio 10000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Example INVITE from sipd to h323d

```
Apr 20 04:43:54.240 On 127.0.0.1:5070 received from 127.0.0.1:5060
INVITE sip:800@127.0.0.1:5070;acme_sag=sag1;acme_irealm=internal SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5060;branch=z9hG4bK000c0210385hv9gpt001.1
From: sipp <sip:7789@192.168.200.6:5062>;tag=SDk0jpc01-1
To: sut <sip:800@192.168.200.68:5060>
Call-ID: SDk0jpc01-8e15e11e7f9a20523462972843c7e579-06ahc21
Cseq: 1 INVITE
Contact: <sip:7789@192.168.200.68:5060;transport=udp>
Max-Forwards: 69
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 136
```

```
GenericID: 160400004@000825010100
v=0
o=user1 53655765 2353687637 IN IP4 127.0.0.1
s=-
t=0 0
c=IN IP4       127.0.0.1
m=audio 10000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
Sample SETUP sent from h323d to remote H323 EP
Q.931
    Protocol discriminator: Q.931
Call reference value length: 2
    Call reference flag: Message sent from originating side
    Call reference value: 664D
    Message type: SETUP (0x05)
    Bearer capability
        Information element: Bearer capability
        Length: 3
        ...1 0000 = Information transfer capability: 3.1 kHz audio (0x10)
        .00. .... = Coding standard: ITU-T standardized coding (0x00)
        1... .... = Extension indicator: last octet
        ...1 0000 = Information transfer rate: 64 kbit/s (0x10)
        .00. .... = Transfer mode: Circuit mode (0x00)
        1... .... = Extension indicator: last octet
        ...0 0011 = User information layer 1 protocol: Recommendation G.711 A-
law (0x03)
        1... .... = Extension indicator: last octet
    Display  'sipp'
        Information element: Display
        Length: 4
        Display information: sipp
    Calling party number: '7789'
        Information element: Calling party number
        Length: 6
        .... 1001 = Numbering plan: Private numbering (0x09)
        .110 .... = Number type: Abbreviated number (0x06)
        0... .... = Extension indicator: information continues through the
next octet
.... ..00 = Screening indicator: User-provided, not screened (0x00)
        .00. .... = Presentation indicator: Presentation all 1... .... =
Extension indicator: last octet
        Calling party number digits: 7789
```

IWF Privacy Extensions for Asserted Identity in Untrusted Networks

For IWF privacy, the Oracle Communications Session Border Controller supports:

- IWF caller privacy on unsecure networks—A variant of RFC 3325, where the P-Asserted-Id is inserted when the presentation indicator is set to allowed. This feature enables bi-directional SIP/H.323 IWF support for CPID hiding by using the presentation indicators in the Calling Party Number information element forH.323 signaling, and RFC 3325-based privacy support for SIP signaling. It allows the Oracle Communications Session Border

Controller to insert the P-Asserted-Identity and the Privacy header in the INVITE when the presentation indicator is set to restricted.
The presence, or absence, of P-Asserted-Identity and Privacy headers in the SIP INVITE informs the remote SIP proxy or endpoint to either block or advertise the CPID.

- IWF caller privacy on secure connections—When the H.323 endpoint sends a SETUP with presentation indicator set to allowed, the Oracle Communications Session Border Controller does not insert the P-Asserted-Identity in the INVITE. The SIP INVITE needs the P-Asserted-Identity header to support calling line identification presentation (CLIP) to calling line identification restriction (CLIR) in an IP multimedia subsystem (IMS) solution. This feature allows the Oracle Communications Session Border Controller to insert the P-Asserted-Identity in the INVITE when the presentation indicator is set to allowed.

Now the Oracle Communications Session Border Controller supports an enhancement to IWF caller privacy where the P-Preferred-Identity is inserted instead of the P-Asserted-Identity.

In this implementation, when the incoming H.323 Setup message has a presentation indicator set to restricted and the ingress H.323 session agent has the new PPreferredId option configured, the Oracle Communications Session Border Controller sends the Privacy header with P-Preferred-Identity (instead of P-Asserted-Identity).

IWF Call Originating in H.323

This section shows an example H.323 Setup that arrives from an H.323 endpoint, and how the Oracle Communications Session Border Controller adds the P-Preferred-Identity header (which has calling party number information) and the Privacy header to the SIP INVITE.

Sample H.323 Setup from a Remote Endpoint

```
Q.931
    Protocol discriminator: Q.931
    Call reference value length: 2
    Call reference flag: Message sent from originating side
    Call reference value: 2FB6
    Message type: SETUP (0x05)
    Bearer capability
        Information element: Bearer capability
        Length: 3
        ...0 1000 = Information transfer capability: Unrestricted digital
information (0x08)
        .00. .... = Coding standard: ITU-T standardized coding (0x00)
        1... .... = Extension indicator: last octet
        ...1 0011 = Information transfer rate: 384 kbit/s (0x13)
        .00. .... = Transfer mode: Circuit mode (0x00)
        1... .... = Extension indicator: last octet
...0 0101 = User information layer 1 protocol: Recommendation H.221 and H.242
(0x05)
        1... .... = Extension indicator: last octet
    Display  'rdoe\000'
        Information element: Display
        Length: 9
        Display information: jdoe\000
    Calling party number: '42343'
        Information element: Calling party number
        Length: 6
        .... 0001 = Numbering plan: E.164 ISDN/telephony numbering (0x01)
```

```
        .000 .... = Number type: Unknown (0x00)
        0... .... = Extension indicator: information continues through the
next octet
        .... ..00 = Screening indicator: User-provided, not screened (0x00)
        .01. .... = Presentation indicator: Presentation restricted (0x01)
        1... .... = Extension indicator: last octet
        Calling party number digits: 42343
E.164 Calling party number digits: 42343
    Called party number: '780'
        Information element: Called party number
        Length: 4
        .... 0001 = Numbering plan: E.164 ISDN/telephony numbering (0x01)
        .000 .... = Number type: Unknown (0x00)
        1... .... = Extension indicator: last octet
        Called party number digits: 780
        E.164 Called party number digits: 780
User-user
        Information element: User-user
        Length: 161
        Protocol discriminator: X.208 and X.209 coded user information
```

Sample SIP INVITE from the SBC to a SIP Endpoint

```
Aug 29 15:46:25.214 On [0:0]192.168.200.68:5060 sent to 192.168.200.6:5060
INVITE sip:780@192.168.200.6:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.68:5060;branch=z9hG4bK6810pr20205h2akqe381.1
Contact: "Anonymous"<sip:anonymous@192.168.200.68:5060;transport=udp>
Supported: 100rel
From:
"Anonymous"<sip:anonymous@anonymous.invalid>;tag=SDfd9sa01-000000ba00023280
To: <sip:780@192.168.200.6:5060>
Call-ID: SDfd9sa01-6f93292521b83a0980647f34451c5afd-06ahc21
CSeq: 2 INVITE
P-Preferred-Identity: "rdoe"<sip:42343@192.168.200.68:5060>
<b>Privacy: id<\b>
Content-Length: 180
Content-Type: application/sdp
Max-Forwards: 70
v=0
o=IWF 5 5 IN IP4 192.168.200.5
s=H323 Call
c=IN IP4 192.168.200.65
t=0 0
m=audio 5010 RTP/AVP 0
a=rtpmap:0 PCMU/8000/1
m=video 5014 RTP/AVP 31
a=rtpmap:31 H261/9000/1
```

Before You Configure

Before you configure your Oracle Communications Session Border Controller to support this feature, note the following considerations:

- The ingress H.323 session agent cannot be configured with the NoPAssertedId option

- For use in Release 4.1.1 and higher, the global SIP configuration should be configured with the disable-ppi-to-pai option; the older disable-privacy option will also work

P-Preferred-Identity Configuration

To enable the inclusion of P-Preferred-Identity:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Select the session agent where you want to apply this feature.

   ```
   ORACLE(session-agent)# select
   <hostname>:
   1: 204.12.60.5      realm=private
   2: 124.21.5.3       realm=public
   selection:1
   ORACLE(session-agent)#
   ```

5. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**PPreferredId**), and then press Enter.

   ```
   ORACLE(realm-config)# options +PPreferredId
   ```

   If you type options PPreferredId, you will overwrite any previously configured options. In order to append the new option to the session agent's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

6. Save and activate your configuration.

IWF Privacy for Business Trunking

The Oracle Communications Session Border Controller supports IWF Privacy: Caller Privacy on Unsecure Networks and IWF Privacy: Caller Privacy on Secure Connections, but IWF Privacy for Business Trunking, supports the case where SIP and H.323 PBXs are connected to the core IMS system. Traffic originated at the IP PBXs terminates either at other PBXs or at the PSTN, and includes the possibility of accepting incoming traffic from the PSTN. CLIP and CLIR must be supported for calls in either direction for calls that require interworking between SIP and H.323. Unlike the two features described above, this new feature supports the fact that only a network-based application server has sufficient privilege to assert the identity of the calling party.

Thus, for this feature, the Oracle Communications Session Border Controller does not force privacy. Instead, the implemented feature assumes that the H.323 session agent is an IP PB X, and the Oracle Communications Session Border Controller only indicates to the SIP core that privacy is being requested. In other words, the Oracle Communications Session Border Controller is not required to interwork the H.323 presentation indicator parameter to RFC 3325 by including the P-Asserted-Identity header. The indication to the SIP core that privacy is being requested excludes identity assertion.

You configure this feature using two session agent options:

- allowCPN—Set in the egress H.323 session agent, allows the Oracle Communications Session Border Controller to send the calling party number information element (IE), even when the presentation indicator is set to restricted.
- NoPAssertedId—Set in the ingress H.323 session agent; when the incoming SETUP message has the presentation indicator is set to restricted, instructs the Oracle Communications Session Border Controller to send a Privacy header without the P-Asserted-Identity and not to make the From header anonymous.

A Call Originating in H.323

This section describes for the IWF Privacy for Business trunking feature works for a call originating in H.323 that requires interworking to SIP.

When the Oracle Communications Session Border Controller receives an H.323 SETUP with a presentation indicator of the calling party information element (IE) is set to restricted and this SETUP was received from a session agent is configured with the NoPAssertedID option, the Oracle Communications Session Border Controller only adds the Privacy header with the value ID. In this case, there will be no P-Asserted-Identity and the From header will contain the calling Party information that was extracted from the callingPartyIE. The Oracle Communications Session Border Controller assumes that the PBX will send the callingPartyNumber in the IE, even though it would like to have the calling party number restricted.

Sample SETUP Message from an H.323 Endpoint

```
Q.931
    Protocol discriminator: Q.931
    Call reference value length: 2
    Call reference flag: Message sent from originating side
    Call reference value: 2FB6
    Message type: SETUP (0x05)
    Bearer capability
        Information element: Bearer capability
Length: 3
        ...0 1000 = Information transfer capability: Unrestricted digital
information (0x08)
        .00. .... = Coding standard: ITU-T standardized coding (0x00)
        1... .... = Extension indicator: last octet
        ...1 0011 = Information transfer rate: 384 kbit/s (0x13)
        .00. .... = Transfer mode: Circuit mode (0x00)
        1... .... = Extension indicator: last octet
        ...0 0101 = User information layer 1 protocol: Recommendation H.221
and H.242 (0x05)
        1... .... = Extension indicator: last octet
    Display  'jdoe\000'
```

```
        Information element: Display
Length: 9
        Display information: jdoe\000
    Calling party number: '42343'
        Information element: Calling party number
        Length: 6
        .... 0001 = Numbering plan: E.164 ISDN/telephony numbering (0x01)
        .000 .... = Number type: Unknown (0x00)
        0... .... = Extension indicator: information continues through the
next octet
        .... ..00 = Screening indicator: User-provided, not screened (0x00)
        .01. .... = Presentation indicator: Presentation restricted (0x01)
        1... .... = Extension indicator: last octet
        Calling party number digits: 42343
        E.164 Calling party number digits: 42343
    Called party number: '780'
        Information element: Called party number
        Length: 4
        .... 0001 = Numbering plan: E.164 ISDN/telephony numbering (0x01)
        .000 .... = Number type: Unknown (0x00)
        1... .... = Extension indicator: last octet
        Called party number digits: 780
        E.164 Called party number digits: 780
User-user
        Information element: User-user
        Length: 161
        Protocol discriminator: X.208 and X.209 coded user information
```

Sample INVITE from the Oracle Communications Session Border Controller to the SIP Endpoint

```
May  5 15:11:51.996 On [0:0]192.168.200.68:5060 sent to 192.168.200.6:5060
INVITE sip:780@192.168.200.6:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.68:5060;branch=z9hG4bK00020a20eg11s94pg700.1
Contact: "jdoe"<sip:42343@192.168.200.68:5060;transport=udp>
Supported: 100rel
From: "jdoe"<sip:42343@192.168.200.68:5060>;tag=SDetur801-00000194000e2ce8
To: <sip:780@192.168.200.6:5060>
Call-ID: SDetur801-231c7b30909ca525ce12cbfeb57754ea-06ahc21
CSeq: 2 INVITE
Privacy: id
Content-Length: 231
Content-Type: application/sdp
Max-Forwards: 70
v=0
o=IWF 2 2 IN IP4 192.168.200.65
s=H323 Call
c=IN IP4 192.168.200.65
t=0 0
m=audio 5004 RTP/AVP 8 0
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000/1
m=video 5006 RTP/AVP 31 34
```

```
a=rtpmap:31 H261/8000
a=rtpmap:34 H263/9000/1
```

A Call Originating in SIP

This section describes for the IWF Privacy for Business trunking feature works for a call originating in SIP that requires interworking to H.323.

When the Oracle Communications Session Border Controller receives a SIP INVITE with a Privacy header that has the value ID, it sets the presentation indicator to restricted in the corresponding H.323 SETUP message. If the H.323 session agent is configured with the allowCPN option, the Oracle Communications Session Border Controllersends the display IE and the calling party number to the H.323 session agent. If that option is not set in the H.323 session agent, then the Oracle Communications Session Border Controller reverts to its default behavior, which is to not to send the display IE and to hide the calling party number.

Sample INVITE from a SIP Endpoint to the Oracle Communications Session Border Controller

```
May  5 14:41:54.513 On [0:0]192.168.200.68:5060 received from
192.168.200.6:5060
INVITE sip:800@192.168.200.68:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.6:5060
From: sipp <sip:sipp@192.168.200.6:5060>;tag=1
To: sut <sip:800@192.168.200.68:5060>
Call-ID: 1.3068.192.168.200.6@sipp.call.id
Cseq: 1 INVITE
Contact: sip:sipp@192.168.200.6:5060
Privacy: id
P-Asserted-Identity: sipp <sip:1234@192.168.200.6:5060>
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 136
v=0
o=user1 53655765 2353687637 IN IP4 127.0.0.1
s=-
t=0 0
c=IN IP4       127.0.0.1
m=audio 10000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Sample SETUP from the Oracle Communications Session Border Controller to the H.323 Endpoint

```
Q.931
    Protocol discriminator: Q.931
    Call reference value length: 2
    Call reference flag: Message sent from originating side
    Call reference value: 44B0
    Message type: SETUP (0x05)
    Bearer capability
```

```
        Information element: Bearer capability
        Length: 3
        ...1 0000 = Information transfer capability: 3.1 kHz audio (0x10)
        .00. .... = Coding standard: ITU-T standardized coding (0x00)
        1... .... = Extension indicator: last octet
        ...1 0000 = Information transfer rate: 64 kbit/s (0x10)
        .00. .... = Transfer mode: Circuit mode (0x00)
        1... .... = Extension indicator: last octet
        ...0 0011 = User information layer 1 protocol: Recommendation G.711 A-
law (0x03)
        1... .... = Extension indicator: last octet
    Display  'sipp'
        Information element: Display
        Length: 4
        Display information: sipp
Calling party number: '1234'
        Information element: Calling party number
Length: 6
        .... 0001 = Numbering plan: E.164 ISDN/telephony numbering (0x01)
        .010 .... = Number type: National number (0x02)
        0... .... = Extension indicator: information continues through the
next octet
        .... ..00 = Screening indicator: User-provided, not screened (0x00)
        .01. .... = Presentation indicator: Presentation restricted (0x01)
        1... .... = Extension indicator: last octet
        Calling party number digits: 1234
        E.164 Calling party number digits: 1234
    Called party number: '800'
        Information element: Called party number
        Length: 4
.... 0001 = Numbering plan: E.164 ISDN/telephony numbering (0x01)
        .010 .... = Number type: National number (0x02)
        1... .... = Extension indicator: last octet
        Called party number digits: 800
        E.164 Called party number digits: 800
    User-user
        Information element: User-user
        Length: 159
        Protocol discriminator: X.208 and X.209 coded user information
```

allowCPN Configuration

You can set both of these options in the same H.323 session agent.

To set the allowCPN option for an H.323 session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
```

4. Use the ACLI **select** command so that you can work with the session agent configuration to which you want to add this option.

```
ORACLE(session-agent)# select
```

5. **options**—Set the options parameter by typing **options**, a Space, the option name **allowCPN** with a plus sign in front of it, and then press Enter.

```
ORACLE(session-agent)# options +allowCPN
```

If you type options allowCPN (without the plus sign), you will overwrite any previously configured options. In order to append the new option to the session-agent's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

To set the NoPAssertedId option for an H.323 session agent:

7. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

8. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

9. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
```

10. Use the ACLI **select** command so that you can work with the session agent configuration to which you want to add this option.

```
ORACLE(session-agent)# select
```

11. **options**—Set the options parameter by typing **options**, a Space, the option name **NoPAssertedId** with a plus sign in front of it, and then press Enter.

```
ORACLE(session-agent)# options +NoPAssertedId
```

If you type options NoPAssertedId (without the plus sign), you will overwrite any previously configured options. In order to append the new option to the session-agent's options list, you must prepend the new option with a plus sign as shown in the previous example.

12. Save and activate your configuration.

Trunk Group Documentation

The Oracle Communications Session Border Controller supports routing using trunk groups within the context of SIP and H.323 interworking. Refer to the Trunk Group URI section of the SIP Signaling chapter for full explanation of Trunk Group and Trunk Context support and configuration.

This documentation was formerly duplicated in this IWF chapter.

IWF COLP COLR Support

When you enable the connected line identity presentation (COLP) and connected line identity restriction (COLR) feature for calls being translated between SIP and H.323, the Oracle Communications Session Border Controller converts the H.323 Connected Number Information element (IE) to the SIP P-Asserted-Identity (PAI) header and vice versa.

When there is no Q.931 Connected Number IE, the Oracle Communications Session Border Controller converts the H.225 Connected Address alias (either E.164 or Public Party Number).

This section describes show the IWF COLP/COLR feature works for IWF calls that originate in SIP and are translated to H.323, and for calls that originate in H.323 and are translated to SIP.

SIP to H.323 Calls

For this type of call, the Oracle Communications Session Border Controller checks the Connect that it receives for a Q.931 Connected Number IE. If it does not find one, then it continues by checking for H.225 Connected Address alias (either E.164 or Public Party Number). Then, it takes one of the following courses of action depending on circumstances:

- If it finds the Q.931 Connected Number IE, the Oracle Communications Session Border Controller extracts the screening indicator and the presentation indicator.
- If there is no Q.931 Connected Number IE, the Oracle Communications Session Border Controller extracts the screening indicator and the presentation indicator from the H.225 Connect-UUIE of the Connect message.

With these pieces of information in place, the Oracle Communications Session Border Controller performs the conversion from H.323 Connected Number IE to SIP P-Asserted-Identity (PAI) header if and only if the screening indicator is either one of the following:

- Network provided
- User-provided, verified and passed

Then the Oracle Communications Session Border Controller adds a SIP PAI header (with URI value) to the 200 OK message that it sends in the SIP call leg. The user part of the URI is set to the value of the Q.931 Connected Number IE's numberDigits field, or to dialDigits value from the Connected Address alias. When the number type is a national number, the Oracle Communications Session Border Controller adds a plus sign (+) and the IWF country code (that you configure) to the beginning of the user part. If the number type is an international number, the Oracle Communications Session Border Controller only adds a plus sign (+). And when the Connected Number is empty, the Oracle Communications Session Border Controller sets the user part of the PAI header URI to anonymous. When the value in the presentation indicator is Presentation restricted, the Oracle Communications Session Border Controller adds the SIP Privacy header (with the value id) to the 200 OK.

In cases when it does not find a screening indicator, the Oracle Communications Session Border Controller will not perform the conversion from the H.323 Connected Number IE to the SIP P-Asserted-Identity (PAI) header.

H.323 to SIP Calls

For this type of call, the Oracle Communications Session Border Controller checks the 200 OK message for a SIP PAI header and a SIP Privacy header. Before it sends a Connect message on the H.323 call leg, the Oracle Communications Session Border Controller generates a Connected Number. It uses the Connected Number to insert a Q.931 Connected Number IE and an H.225 Connected Address alias (type E.164) into the Connect message. The Connected Number is generated in this way:

- If the
  - SIP PAI header is not found, or
  - User part of its URI value is unknown or anonymous, or
  - User part of its URI does not follow the H.225 NumberDigits syntax,
    then the Connect Number that the Oracle Communications Session Border Controller generates is a Q.931 Connected Number IE that has no digits and a number type of unknown. In this case, the Oracle Communications Session Border Controller will not insert an H.225 Connected Address alias into the Connect message.

    The presentation indicator is set to Number not available due to interworking, and the screening indicator to Network provided. The H.225 NumberDigits's syntax requires that it be between 1 and 128 characters, and only contain these characters: 0 through 9, the pound sign (#), the asterisk (*), and the comma (,).
- In all other cases, the Oracle Communications Session Border Controller uses the user part of the URI as the digits for the Connected Number after it performs the following:
  - Strips the plus sign in front of the number, if there is one
  - Strips the IWF country code at the beginning of the number, if there is one

Then the Oracle Communications Session Border Controller inserts the Connected Number into the Connect message as the Q.931 Connected Number IE and an H.225 Connected Address alias (type E.164).

If the IWF country code is found in the PAI, the Oracle Communications Session Border Controller sets the type of Q.931 Connected Number IE to National Number. Otherwise, the Oracle Communications Session Border Controller sets it to international. The screening indicator is set to Network provided, and the presentation indicator is set to Presentation Restricted if the Oracle Communications Session Border Controller finds a SIP Privacy header with a value of id, or Presentation Allowed is there is not SIP Privacy header.

IWF COLP COLR Configuration

You configure IWF COLP/COLR support in the IWF configuration by setting two options:

- **colp-colr-iwf**—Setting this option enables support for IWF COLP/COLR
- **colp-colr-country-code**—Must be set if you configure the **colp-colr-iwf** option to recognize or build a national number; the value you enter here:
  - Must be a string of digits from 0 to 9
  - Cannot exceed 32 digits

– Cannot contain any non-numeric characters; while it allows you to enter them, the system ignores any non-digits characters and so the feature might not work as needed To enable IWF COLP/COLR support:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **iwf-config** and press Enter. The system prompt will change to let you know that you can configure individual

```
ORACLE(session-router)# iwf-config
```

4. **options**—Set the options parameter by typing **options**, a Space, the option names with a plus sign in front, and then press Enter.

   Your entry for the **colp-colr-country-code** option require that you type in the

   entire option name, an equal sign (=), and then the country code value.

   To enter both options at once, separate the two with one command and enclose your entire entry in quotation marks ( ); see the following example for command-line syntax.

```
ORACLE(iwf-config)# options +colp-colr-iwf,colp-colr-country-code=1
```

   If you type this enter without the plus sign, you will overwrite any previously configured options. In order to append options to the IWF configuration's options list, you must prepend the new options with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Options for Calls that Require the IWF

You can configure several specific behaviors by configuring options for calls that require the IWF, and set them for the H.323 side of the call. These options are listed and defined in the table below. Options can be configured either globally for the H.323 configuration, individually for an H.323 interface, or for H.323 session agents.

Global Configuration for H.323

To configure options globally for H.323:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **h323** and press Enter.

```
ORACLE(session-router)# h323
```

From this point, you can configure H.323 parameters. To view see all H.323 parameters, enter a **?** at the system prompt.

4. Type **options**, a space, and the name of the option you want to use. In this example, the MapG729 will map H.245 G.729 to SDP G.729 with Annex B and vice versa.

```
ORACLE(h323)# options MapG729
```

Individual Configuration for H.323

To configure options per individual H.323 interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **h323** and press Enter.

```
ORACLE(session-router)# h323
```

4. Type **h323-stacks** and press Enter. The system prompt changes again to let you know that you can begin configuring individual parameters.

```
ORACLE(h323)# h323-stacks
ORACLE(h323-stack)#
```

From this point, you can configure H.323 interface parameters. To view see all H.323 interface parameters, enter a **?** at the system prompt.

5. Type **options**, a space, and the name of the option you want to use. In this example, the MapG729 will map H.245 G.729 to SDP G.729 with Annex B and vice versa.

```
ORACLE(h323-stack)# options
```

Configuring H.323 SA Options

To configure options for H.323 session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **session-agent** and press Enter.

```
ORACLE(session-router)# session-agent
```

From this point, you can configure session agent parameters. To view see all session agent parameters, enter a **?** at the system prompt.

4. Type **options**, a space, and the name of the option you want to use. In this example, the MapG729 will map H.245 G.729 to SDP G.729 with Annex B and vice versa.

```
ORACLE(h323-stack)# options MapG729
```

H.323 SA Options

Options	Description
MapG729	Oracle Communications Session Border Controller maps H.245 G.729 to SDP G.729 with Annex B and vice versa. Applicable only to calls that require the IWF.
ColonG729	Oracle Communications Session Border Controller uses the : (colon) instead of the = (equal sign) in the media attribute line a=fmtp:18 annexb=yes/no when mapping H.245 G.729 or SDP G.729 with Annex B. Applicable only to calls that require the IWF.
IwfLRQ	Oracle Communications Session Border Controller sends an INVITE (with no SDP) to a redirect server in response to an incoming LRQ received on an H.323 interface. If a 3xx message with a redirected contact header is returned, the Oracle Communications Session Border Controller will send an LCF in response to the LRQ. Otherwise, it will send an LRJ.
NoG729AnnexB	SDP received by the IWF with H.729 and no FMTP will be mapped to G.729 on the H.323 side of the call. Can also be set in the session agent options parameter.
sameT38Port	Oracle Communications Session Border Controller's H.323 process does not allocate separate ports for audio and T.38. Oracle Communications Session Border Controller will send the same audio port in the OLCAck that it sees in a request mode for T.38 and a new OLC for T.38.
pvtStats	Oracle Communications Session Border Controller includes program value tree (PVT) statistics in the show h323d display that are a sum of the PVT statistics for all H.323 interfaces. Used for debugging purposes.
acceptAl	Oracle Communications Session Border Controller accepts all the codecs received in the SIP 200OK and builds the TCS accordingly.

Suppress SIP Reliable Response Support for IWF

For IWF-originated calls, the Oracle Communications Session Border Controller now allows you to configure the suppression of the SIP 100rel option tag on a per-H.323 interface (stack) basis.

When a calls originates on the H.323 side for a call that requires interworking between H.323 and SIP, the Oracle Communications Session Border Controller inserts the 100rel option tag in the Supported header of the outgoing SIP INVITE. Although this behavior is required for RFC 3262 conformance, and is ignored by endpoints that do not support this RFC, suppressing the reliable response can alleviate processing burdens and avoid the possibility that an endpoint could mishandle the response.

In addition, enabling this feature suppresses the same 100rel options tag in the Required header for outgoing IWF responses for which an incoming SIP INVITE had that same tag in its Supported header. If an incoming INVITE requires reliable provisional responses and the SIP feature configuration is set to accept the 100rel, the Oracle Communications Session Border Controller then includes the 100rel option tag in the outgoing response's Required header. When the SIP feature is not so configured, the Oracle Communications Session Border Controller rejects the INVITE with a 420 Bad Extension response.

Without this option, you can suppress the reliable response on a global basis or per SIP next-hop by using the SIP feature configuration. However, using this feature allows a finer degree of granularity by making the functionality only applicable to IWF calls that originate in H.323.

suppress100rel Configuration

To suppress the SIP 100rel option tag:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **h323** and press Enter.

   ```
   ORACLE(session-router)# h323
   ORACLE(h323)#
   ```

4. Type **h323-stacks** and press Enter.

   ```
   ORACLE(h323)# h323-stacks
   ORACLE(h323-stack)#
   ```

   If you are adding support for this feature to a pre-existing H.323 interface (stack), then you must select (using the ACLI **select** command) the configuration that you want to edit.

5. **options**—Set the options parameter by typing **options**, a Space, the option name **suppress100rel** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(h323-stack)# options +suppress100rel
   ```

   If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to this** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

IWF Codec Negotiation H.323 Slow Start to SIP

For instances when the Oracle Communications Session Border Controller is translating a call initiated in H.323 slow start to SIP, you can enable a setting in the IWF configuration that prevents the sending an SDP offer in the SIP INVITE. Instead, the Oracle Communications Session Border Controller expects to see an SDP offer from the SIP endpoint in a provisional or reliable/provisional 200 OK, and then sends an answer in an ACK or PRACK.

With this parameter disabled (default), the Oracle Communications Session Border Controller populates the SIP INVITE with SDP based on the media profiles applied to the ingress H.323 session agent or the IWF configuration.

IWF Codec Negotiation Configuration

To prevent the Oracle Communications Session Border Controller from sending an SDP offer in the SIP INVITE for a call being translated between H.323 slow start and SIP:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **iwf-config** and press Enter.

   ```
   ORACLE(session-router)# iwf-config
   ORACLE(iwf-config)#
   ```

4. **slow-start-no-sdp-in-invite**—Enable this parameter if you want to prevent the Oracle Communications Session Border Controller from sending an SDP offer in the SIP INVITE for an IWF call initiated in H.323 slow start (being translated to SIP). The default is disabled. Valid values are:

   - enabled | disabled

5. Save and activate your configuration.

IWF H.245 Signaling Support for G.726

In addition to providing G.726 support for pure SIP and pure H.323 calls, the Oracle Communications Session Border Controller supports the G.726 payload type for H.245 and calls that require interworking (IWF) between SIP and H.323.

For IWF calls using ITU-T G.726 as the audio codec, the SIP call leg requires G.726 in the SDP. The H.323 side of the call signals G.726 (in the H.245 openLogicalChannel and TerminalCapabilitySet messages) by including a GenericCapability defining G.726 as the codec. In the GenericCapability, the capabilityIdentifier and maxBitRate parameters identify G.726. While a capabilityIdentifier with 0.0.7.726.1.0 designates G.726, the maxBitRate designate the data transmission rate.

Codec	Max Bit Rate	Data Rate
G726-16	160	16 kbit/s
G726-24	240	24 kbit/s
G726-32	320	32 kbit/s
G726-40	400	40 kbit/s

To support G.726 for IWF calls, the Oracle Communications Session Border Controller converts the G726-X value in the SDP of SIP messages to a GenericCapability structure in H.323/H.245 messages, and the conversion works the same way in reverse.

H.245 and G.726 Configuration

To enable this feature, you do need to set up media profile configurations appropriately. Media profiles now allow you to set the configuration to any of the four G.726 encodings (as defined by ITU G726 Annex B and RFC 3551). You must create one media profile for each of the four different supported data rates. In addition, you are also required to set a genericAudioCapability media profile.

Media Profile for H.245 and G.726 Configuration

To set a media profile for H.245 and IWF G.726 support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **media-profile** and press Enter.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. **name**—Set the name of the media profile to G726-16. Values to support this feature are: G726-16, G726-24, G726-32, and G726-40.
5. **media-type**—Set the media type to use for this media profile; for generic video, set this parameter to **audio**. Valid values are:
   - audio | video | application | data
6. **payload-type**—Set the payload type to use for the generic video media profile.
7. **transport**—Set the transport type to use for the generic video media profile. The default value is **RTP/AVP**. Valid values are:
   - UDP | RTP/AVP
8. Complete the rest of the media profile configuration as needed.
9. Save and activate your configuration.

   The following is a sample of a media profile configuration for H.245/IWF G.726 support:

   ```
   media-profile
           name                            g726-40
           media-type                      audio
           payload-type                    105
           transport                       RTP/AVP
           req-bandwidth                   0
           frames-per-packet               0
           parameters
           average-rate-limit              0
           sdp-rate-limit-headroom         0
           sdp-bandwidth                   disabled
   ```

Media Profile Configuration for Generic Audio Support

To set a media profile for generic audio support:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **media-profile** and press Enter.

```
ORACLE(session-router)# media-profile
ORACLE(media-profile)#
```

4. **name**—Set the name of the generic audio media profile to genericAudioCapability. There is no default for this parameter.
5. **media-type**—Set the media type to use for this media profile; for generic video, set this parameter to **audio**. Valid values are:
   - audio | video | application | data
6. **payload-type**—Set the payload type to use for the generic audio media profile.
7. **transport**—Set the transport type to use for the generic audio media profile. The default value is **RTP/AVP**. Valid values are:
   - UDP | RTP/AVP
8. Complete the rest of the media profile configuration as needed.
9. Save and activate your configuration.

   The following is a sample of a generic audio media profile configuration:

```
media-profile
        name                            genericAudioCapability
        media-type                      audio
        payload-type                    104
        transport                       RTP/AVP
        req-bandwidth                   0
        frames-per-packet               0
        parameters
        average-rate-limit              0
        sdp-rate-limit-headroom         0
        sdp-bandwidth                   disabled
```

Flow Control Mapping for Interworking Function (IWF) Video

H.245 is a protocol for the transmission of call management and control signals in networks using H.323 equipment. The H.245 specification is used in audio, video, and data transmissions, as well as in Voice over IP (VoIP). H.245 messages are sent over special channels called H.245 control channels.

H.245 signaling is used to manage and control call setup and connection. Functions of H.245 include determining which endpoint is to be the primary and which is to be the secondary during the call, opening and closing of multiplexed data-transfer paths between the endpoints, establishing an upper limit to the data transfer speed on each logical channel, information exchanges between endpoints concerning the types of data each endpoint can send and receive, requests by the receiving endpoint for changes in the mode of the data sent by the transmitting endpoint, and requests by either endpoint to end the call.

In the H.245 standard, the FlowControlCommand message is used to specify the upper limit of bit rate of either a single logical channel or the whole multiplex. The following is an excerpt from the H.245 standard.

Command Message: Flow Control (from H.245 standard)

```
==================================================
FlowControlCommand ::= SEQUENCE
{
        scope CHOICE
    {
    logicalChannelNumber LogicalChannelNumber,
    resourceID INTEGER (0..65535),
    wholeMultiplex NULL
    },
    restriction CHOICE
    {
    maximumBitRate INTEGER (0..16777215), -- units 100 bit/s
    noRestriction NULL
    },
    ...
}
==================================================
```

A terminal may send this command to restrict the bit rate that the far-end terminal sends. A receiving terminal must comply with this command.

In an H.323 environment, the Oracle Communications Session Border Controller previously used the FlowControlCommand to map to SIP using either the Real-Time Control Protocol (RTCP) feedback function, or the SIP signaling path (for example, the INFO method).

The Oracle Communications Session Border Controller now supports the SIP counter part of the H.245 FlowControlCommand using the SIP signaling path with the INFO method. The Oracle Communications Session Border Controller sends the SIP INFO message with "change_bitrate" rate parameter that has the value 100* maxBitRate from the corresponding H.245 FlowControlCommnad message. For example, in the following messages, the incoming H.323 message with the H.245 FlowControlCommand, is converted into the outgoing SIP INFO message with the message body.

Incoming H.323 Message with H.245 FlowControlCommand:

```
H.245
PDU Type: command (2)
        command: flowControlCommand (4)
            flowControlCommand
                scope: logicalChannelNumber (0)
                    logicalChannelNumber: 102
```

```
                restriction: maximumBitRate (0)
                    maximumBitRate: 4480
```

Outgoing SIP INFO Message:

```
Message Body
        eXtensible Markup Language
            <?xml
                version="1.0"
                encoding="utf-8"
                ?>
            <media_control>
                <vc_primitive>
                    <to_encoder>
                        <change_bitrate>
                            4480000
                            </change_bitrate>
                        </to_encoder>
                    </vc_primitive>
                </media_control>
```

Customized G.729 Support

The Oracle Communications Session Border Controller supports the use of custom G.729 encoding for calls that require interworking between SIP and H.323. If you use a proprietary G.729 encoding format in your network, then you might need to use this feature.

When you set the **acceptG729abFormat** option in the global H.323 configuration, the Oracle Communications Session Border Controller performs conversions like those in the following examples:

- For calls initiated in SIP, the Oracle Communications Session Border Controller can parse RTP map strings such as G.729a and G.729ab in the SDP, and then map them to H.245 data types.
  - G.729a becomes g729AnnexA.
  - G.729ab becomes g729AnnexAwAnnexB.
- For calls initiated in H.323, the Oracle Communications Session Border Controller can create non-standard RTP map strings such as G.729a and G.729ab from mapped H.245 data types.
  - g729 becomes G729.
  - g729AnnexA becomes G.729a.
  - g729AnnexAwAnnexB becomes G.729ab.

When you enable the **acceptG729abFormat** option, the Oracle Communications Session Border Controller performs customized G.729 mapping in the following instances.

- For calls initiated in SIP and translated to H.323, the Oracle Communications Session Border Controller:
  - Converts the SDP in an incoming SIP INVITE to a list of fastStart OpenLogicalChannel requests that are in turn included in the outgoing Setup message.

  - Converts the list of fastStart OpenLogicalChannelAck responses (which can be received in any message up to and including the Connect message) to SDP sent with a SIP response.
- For calls initiated in H.323 and translated to SIP, the Oracle Communications Session Border Controller:
  - Converts the list of fastStart OpenLogicalChannel requests to SDP in the outgoing SIP INVITE.
  - Converts SDP in a SIP response (such as a 200 OK) to the list of fastStart OpenLogicalChannelAck responses included with the callProceeding, Progress, Alerting, or Connect message. This depends on when the SDP is received on the SIP side.
- For all IWF calls regardless of initiating protocol, the Oracle Communications Session Border Controller:
  - Converts SDP on the SIP side to the terminalCapabilitySet message to be sent on the H.323 side.

Also note that when the format is G729, the Oracle Communications Session Border Controller maps it to g729wAnnexB if the a=fmtp:18 annexb=yes attribute is present. When the a=fmtp:18 annexb=no attribute is present, the Oracle Communications Session Border Controller maps G729 to g729. And with no a=fmtp:18 annexb=no attribute, the Oracle Communications Session Border Controller also maps G729 to g729 when this option is enabled.

The Oracle Communications Session Border Controller also maps G729 to g729 because pure G729 with static payload type 18 does not include an fmtp attribute where annexb=no.

About Dynamic Payload Mapping

G.729a and G.729ab use dynamic payload types, but the Oracle Communications Session Border Controllerdoes not propagate these dynamic payload types to corresponding dynamicRTPPayloadType (an optional field in OpenLogicalChannel requests) on the H.323 side.

For an IWF call initiated in H.323, the dynamic payload types for G.729a and G.729ab are retrieved from media profile configurations when the Oracle Communications Session Border Controller converts the list of fastStart OpenLogicalChannel requests to SDP sent on the SIP side. As a result, you must set up media profile configurations for G.729a and G.729ab for the feature to work properly. In these media profiles, the following parameters must be set as follows:

- **name**—For the G.729a profile, set the **name** to **G.729a**. For the **G.729ab** profile, set the **name** to **G.729ab**.
- **payload-type**—For each media profile (**G.729a** and **G.729ab**), DO NOT use payload type 18, which is the static payload type used for G729.

Customized G.729 Configuration

This section shows you how to configure the **acceptG729abFormat** option in the global H.323 configuration.

To enable customized G.729 support for IWF calls:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **h323-config** and press Enter.

```
ORACLE(session-router)# h323-config
ORACLE(h323-config)#
```

If you are adding this feature to a pre-existing configuration, select the configuration to edit it.

4. **options**—Set the options parameter by typing options, a Space, the option name **acceptG729abFormat** with a plus sign in front of it. Then press Enter.

```
ORACLE(h323-stack)# options +acceptG729abFormat
```

If you type **options** and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

SIP-H.323 IWF Support for H.264 and H.263+

Signaling protocol interworking between SIP and H.323 supports the H.264 and H.263+ video codecs.

H.264 in H.323 (H.241)

This section describes the H.264 capabilities and media packetization in H.323. Capability exchange signaling looks like this:

```
openLogicalChannel .  SEQUENCE [EMPTY -1] ...
forwardLogicalChannelNumber = 3 .  INTEGER [EMPTY -1] (1..65535)
forwardLogicalChannelParameters .  SEQUENCE [EMPTY -1] ...
 ..   dataType .  CHOICE [EMPTY -1] ...
. . . . . videoData .  CHOICE [EMPTY -1] ...
. . . . . genericVideoCapability .  SEQUENCE [EMPTY -1] ...
. . . . . capabilityIdentifier .  CHOICE [EMPTY -1] ...
. . . . . standard = 7 {itu-t recommendation h 241 0 0 1}.OBJECT
IDENTIFIER [EMPTY-1]
. . . . . maxBitRate = 4480 .  INTEGER [EMPTY -1] (0..-1)
. . . . . collapsing .  SEQUENCE OF [EMPTY -1] SEQUENCE [EMPTY -1] ...
. . . . . * .  SEQUENCE [EMPTY -1] ...
. . . . . parameterIdentifier .  CHOICE [EMPTY -1] ...
. . . . . standard = 41 .  INTEGER [EMPTY -1] (0..127)
```

```
. . . . . . . . . parameterValue .  CHOICE [EMPTY -1] ...
. . . . . . . . . . booleanArray = 64 .  INTEGER [EMPTY -1] (0..255)
. . . . . . . . . * .  SEQUENCE [EMPTY -1] ...
. . . . . . . . . parameterIdentifier .  CHOICE [EMPTY -1] ...
. . . . . . . . . . standard = 42 .  INTEGER [EMPTY -1] (0..127)
. . . . . . . . . parameterValue .  CHOICE [EMPTY -1] ...
. . . . . . . . . . unsignedMin = 29 .  INTEGER [EMPTY -1] (0..65535)
. . . . . . . . . * .  SEQUENCE [EMPTY -1] ...
. . . . . . . . . parameterIdentifier .  CHOICE [EMPTY -1] ...
. . . . . . . . . . standard = 3 .  INTEGER [EMPTY -1] (0..127)
. . . . . . . . . parameterValue .  CHOICE [EMPTY -1] ...
. . . . . . . . . . unsignedMin = 81 .  INTEGER [EMPTY -1] (0..65535)
. . . . . . . . . * .  SEQUENCE [EMPTY -1] ...
. . . . . . . . . parameterIdentifier .  CHOICE [EMPTY -1] ...
. . . . . . . . . . standard = 6 .  INTEGER [EMPTY -1] (0..127)
. . . . . . . . . parameterValue .  CHOICE [EMPTY -1] ...
. . . . . . . . . . unsignedMin = 15 .  INTEGER [EMPTY -1] (0..65535)
. . . . . . . . . * .  SEQUENCE [EMPTY -1] ...
. . . . . . . . . parameterIdentifier .  CHOICE [EMPTY -1] ...
. . . . . . . . . . standard = 4 .  INTEGER [EMPTY -1] (0..127)
. . . . . . . . . parameterValue .  CHOICE [EMPTY -1] ...
. . . . . . . . . . unsignedMin = 7 .  INTEGER [EMPTY -1] (0..65535)
. . . . multiplexParameters .  CHOICE [EMPTY -1] ...
. . . . . h2250LogicalChannelParameters .  SEQUENCE [EMPTY -1] ...
. . . . . . sessionID = 2 .  INTEGER [EMPTY -1] (0..255)
. . . . . . mediaControlChannel .  CHOICE [EMPTY -1] ...
. . . . . . . unicastAddress .  CHOICE [EMPTY -1] ...
. . . . . . . . iPAddress .  SEQUENCE [EMPTY -1] ...
. . . . . . . . . network = 4 '.e.' =0xac10650b <172.16.101.11> .OCTET STRING
[EMPTY -1]
. . . . . . . . . tsapIdentifier = 50137 .  INTEGER [EMPTY -1] (0..65535)
. . . . . . dynamicRTPPayloadType = 109 .  INTEGER [EMPTY -1] (96..127)
. . . . . . mediaPacketization .  CHOICE [EMPTY -1] ...
. . . . . . . rtpPayloadType .  SEQUENCE [EMPTY -1] ...
. . . . . . . . payloadDescriptor .  CHOICE [EMPTY -1] ...
. . . . . . . . . oid = 8 {itu-t recommendation h 241 0 0 0 0}.OBJECT
IDENTIFIER [EMPTY -1]
. . . . . . . . payloadType = 109 .  INTEGER [EMPTY -1] (0..127)
```

This table outlines H.241 to H.264 mappings.

Identifier	Description
Capability name	ITU-T Rec H.241 H.264 Video Capabilities
Capability identifier type	Standard
Capability identifier value	{itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) generic-capabilities(1)}
maxBitRate	This field shall be included, in units of 100 bit/s. This field represents the maximum bitrate of the H.264 Type II bitstream as defined in Annex C/H.264.
collapsing	This field shall contain the H.264 Capability Parameters as given below.

Capabilities

The H.264 capability set is structured as a list of one or more H.264 capabilities, each of which has:

- Profile (mandatory)
- Level (mandatory)
- Zero or more additional parameters

These capabilities communicate the ability to decode using one or more H.264 profiles contained in a GenericCapability structure. For each H.264 capability, optional parameters can appear. These parameters permits a terminal to communicate that it has capabilities in addition to meeting the support requirements for the signaled profile and level.

Optional parameters include: CustomMaxMBPS, CustomMaxDPB, CustomMaxBRandCPB, MaxStaticMBPS, max-rcmd-unit-size, max-nal-unit-size, SampleAspectRatiosSupported, AdditionalModesSupported, and AdditionalDisplayCapabilities.

H.264 Media Packetization

For H.323, systems signal their H.264 mediaPacketization by including: MediaPacketizationCapability.rtpPayload.Type.payloadDescriptor.oid, with the OID having the value {itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) iPpacketization(0) h241AnnexA(0)}.

In compliance with RFC 3984's non-interleaved mode, the following is supported: MediaPacketizationCapability.rtpPayloadType.payloadDescriptor.oid, with the OID having the value {itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) iPpacketization(0) RFC3984NonInterleaved(1)}.

In compliance with RFC 3984's interleaved mode, the following is supported: MediaPacketizationCapability.rtpPayloadType.payloadDescriptor.oid, with the OID having the value {itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0)iPpacketization(0) RFC3984Interleaved(2)}.

H.264 in SIP

H.264 in SIP can contain these optional parameters, which be included in the "a=fmtp" line of SDP if they appear: profile-level-id, max-mbps, max-fs, max-cpb, max-dpb, maxbr, redundant-pic-cap, sprop-parameter-sets, parameter-add, packetization-mode, spropinterleaving-depth, deint-buf-cap, sprop-deint-buf-req, sprop-init-buf-time, sprop-max-dondiff, and max-rcmd-nalu-size.

The profile-level-id parameter is a base 16[6] hexidecimal representation of the following three bytes in sequence:

1. profile_idc
2. profile_oip—Composed of the values from constraint_set0_flag, constraint_set1_flag, constraint_set2_flag, and reserved_zero_5bits—in order of bit significance, starting from the most significant bit.
3. level_idc—Note that reserved_zero_5bits is required to be equal to 0 in [1], but other values for it may be specified in the future by ITU-T or ISO/IEC.

H.264 Packetization Mode

In SIP, the packetization-mode parameter signals the properties of the RTP payload type or the capabilities of a receiver's implementation. Only a single configuration point can be indicated. So when capabilities support more than one packetization-mode are declared, multiple configuration points (RTP payload types) must be used.

- When the value of packetization-mode equals 0 or packetization-mode is not present, the single NAL mode is used.
- When the value of packetization-mode equals 1, the non- interleaved mode is used.
- When the value of packetization-mode equals 2, the interleaved mode is used.

This example shows a SIP offer-answer exchange. Here is the offer SDP:

```
m=video 49170 RTP/AVP 100 99 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E; packetization-mode=0;
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=42A01E; packetization-mode=1;
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42A01E; packetization-mode=2;
```

And here is the answer SDP for the example:

```
m=video 49170 RTP/AVP 100 99 97
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42A01E; packetization-mode=0;
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=42A01E; packetization-mode=1;
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42A01E; packetization-mode=2;
```

H.264 IWF Conversions

This section contains two table that show profile, level, and media packetization conversions for H.264 undergoing interworking.

Profile	H.264 in SIP	H.264 (H.241 in H.323)
H264_PROFILE_STR_BASELINE	66	64
H264_PROFILE_STR_MAIN	77	32
H264_PROFILE_STR_EXTENDED	88	32

H.264 Level	H.2264 in SIP	H.264 (H.241 in H.323)	Constraints
1	10	15	0x00
1b	11	19	0x10
1.1	11	22	0x00
1.2	12	29	0x00
1.3	13	36	0x00
2	20	43	0x00
2.1	21	50	0x00

H.264 Level	H.2264 in SIP	H.264 (H.241 in H.323)	Constraints
2.2	22	57	0x00
3	30	64	0x00
3.1	31	71	0x00
3.2	32	78	0x00
4	40	85	0x00
4.1	41	92	0x00
4.2	42	99	0x00
5	50	106	0x00
5.1	51	113	0x00

H.264 SIP Packetization	H.264 (H.241 in H.323) OID in mediaPacketization
packetization-mode=0	{itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) iPpacketization(0) h241AnnexA(0)}
packetization-mode=1	{itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) iPpacketization(0) RFC3984NonInterleaved(1)}
packetization-mode=2	{itu-t(0) recommendation(0) h(8) 241 specificVideoCodecCapabilities(0) h264(0) iPpacketization(0) RFC3984Interleaved(2)}

IWF Unsupported Parameters

The following H.241 parameters are not supported for interworking: CustomMaxMBPS, CustomMaxFS CustomMaxDPB, CustomMaxBRandCPB, MaxStaticMBPS, max-rcmd-nal-unit-size, max-nal-unit-size, SampleAspectRatiosSupported, AdditionalModesSupported, and AdditionalDisplayCapabilities.

The following SDP parameters are not supported for interworking: max-mbps, max-fs, max-cpb, max-dpb, maxbr, redundant-pic-cap, sprop-parameter-sets, parameter-add, spropinterleaving-depth, deint-buf-cap, sprop-deint-buf-req, sprop-init-buf-time, sprop-max-dondiff, and max-rcmd-nalu-size.

H.263+ in H.323

This section describes the H.264 capabilities and media packetization in H.323. Capability exchange signaling looks like this:

```
. . . . . capability .  CHOICE [EMPTY -1] ...
. . . . . . receiveVideoCapability .  CHOICE [EMPTY -1] ...
. . . . . . . h263VideoCapability .  SEQUENCE [EMPTY -1] ...
. . . . . . . . sqcifMPI = 1 .  INTEGER [EMPTY -1] (1..32)
. . . . . . . . qcifMPI = 1 .  INTEGER [EMPTY -1] (1..32)
. . . . . . . . cifMPI = 1 .  INTEGER [EMPTY -1] (1..32)
. . . . . . . . maxBitRate = 1000 .  INTEGER [EMPTY -1] (1..192400)
. . . . . . . . unrestrictedVector = 0 .  BOOLEAN [EMPTY -1]
. . . . . . . . arithmeticCoding = 0 .  BOOLEAN [EMPTY -1]
. . . . . . . . advancedPrediction = 0 .  BOOLEAN [EMPTY -1]
. . . . . . . . pbFrames = 0 .  BOOLEAN [EMPTY -1]
. . . . . . . . temporalSpatialTradeOffCapability = 0 .  BOOLEAN [EMPTY -1]
```

```
. . . . . errorCompensation = 0 .  BOOLEAN [EMPTY -1]
. . . . . h263Options .  SEQUENCE [EMPTY -1] ...
. . . . . advancedIntraCodingMode = 1 .  BOOLEAN [EMPTY -1]
. . . . . deblockingFilterMode = 1 .  BOOLEAN [EMPTY -1]
. . . . . improvedPBFramesMode = 0 .  BOOLEAN [EMPTY -1]
. . . . . unlimitedMotionVectors = 0 .  BOOLEAN [EMPTY -1]
. . . . . fullPictureFreeze = 1 .  BOOLEAN [EMPTY -1]
. . . . . partialPictureFreezeAndRelease = 0 .  BOOLEAN [EMPTY -1]
. . . . . resizingPartPicFreezeAndRelease = 0 .  BOOLEAN [EMPTY -1]
. . . . . fullPictureSnapshot = 0 .  BOOLEAN [EMPTY -1]
. . . . . partialPictureSnapshot = 0 .  BOOLEAN [EMPTY -1]
. . . . . videoSegmentTagging = 0 .  BOOLEAN [EMPTY -1]
. . . . . progressiveRefinement = 0 .  BOOLEAN [EMPTY -1]
. . . . . dynamicPictureResizingByFour = 0 .  BOOLEAN [EMPTY -1]
. . . . . dynamicPictureResizingSixteenthPel = 1 .  BOOLEAN [EMPTY -1]
. . . . . dynamicWarpingHalfPel = 0 .  BOOLEAN [EMPTY -1]
. . . . . dynamicWarpingSixteenthPel = 0 .  BOOLEAN [EMPTY -1]
. . . . . independentSegmentDecoding = 0 .  BOOLEAN [EMPTY -1]
. . . . . slicesInOrder-NonRect = 0 .  BOOLEAN [EMPTY -1]
. . . . . slicesInOrder-Rect = 0 .  BOOLEAN [EMPTY -1]
. . . . . slicesNoOrder-NonRect = 0 .  BOOLEAN [EMPTY -1]
. . . . . slicesNoOrder-Rect = 0 .  BOOLEAN [EMPTY -1]
. . . . . alternateInterVLCMode = 1 .  BOOLEAN [EMPTY -1]
. . . . . modifiedQuantizationMode = 1 .  BOOLEAN [EMPTY -1]
. . . . . reducedResolutionUpdate = 0 .  BOOLEAN [EMPTY -1]
. . . . . separateVideoBackChannel = 0 .  BOOLEAN [EMPTY -1]
. . . . . videoBadMBsCap = 0 .  BOOLEAN [EMPTY -1]
. . . . . h263Version3Options .  SEQUENCE [EMPTY -1] ...
. . . . . . dataPartitionedSlices = 0 .  BOOLEAN [EMPTY -1]
. . . . . . fixedPointIDCT0 = 0 .  BOOLEAN [EMPTY -1]
. . . . . . interlacedFields = 0 .  BOOLEAN [EMPTY -1]
. . . . . . currentPictureHeaderRepetition = 0 .  BOOLEAN [EMPTY -1]
. . . . . . previousPictureHeaderRepetition = 0 .  BOOLEAN [EMPTY -1]
. . . . . . nextPictureHeaderRepetition = 0 .  BOOLEAN [EMPTY -1]
. . . . . . pictureNumber = 0 .  BOOLEAN [EMPTY -1]
. . . . . . spareReferencePictures = 0 .  BOOLEAN [EMPTY -1]
```

H.263+ in SIP

H.263+ in SIP appears looks like this:

```
a=rtpmap:100 H263-1998/90000
a=fmtp:100 CIF=1; QCIF=1; SQCIF=1; D=1; F=1; I=1; J=1; L=1; S=1; T=1
a=rtpmap:34 H263/90000
a=fmtp:34 CIF=1; QCIF=1; SQCIF=1
```

H.263+ IWF Conversions

This section contains a table showing H.263+ conversions for SIP-h.323 interworking.

H.263+ in H.323 Parameters (Annex) in ftmp line	H.263+ in SIP
sqcifMPI	SQCIF

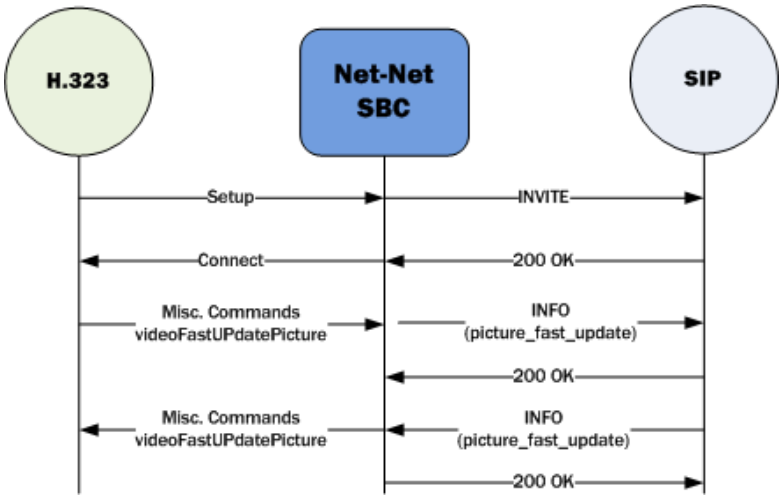
H.263+ in H.323 Parameters (Annex) in ftmp line	H.263+ in SIP
qcifMPI	QCIF
cifMPI	• CIF • CIF4 • CIF16
maxBitRate	N/A
unrestrictedVector	D
arithmeticCoding	E
advancedPrediction	F
pbFrames	G
temporalSpatialTradeOffCapability	N/A
errorCompensation	H
h263Options	N/A
advancedIntraCodingMode	I
deblockingFilterMode	J
improvedPBFramesMode	N/A
unlimitedMotionVectors	N/A
fullPictureFreeze	L
partialPictureFreezeAndRelease	N/A
resizingPartPicFreezeAndRelease	N/A
fullPictureSnapshot	N/A
partialPictureSnapshot	N/A
videoSegmentTagging	N/A
progressiveRefinement	N/A
dynamicPictureResizingByFour	P = 1
dynamicPictureResizingSixteenthPel	P = 2
dynamicWarpingHalfPel	P = 3
DynamicWarpingSixteenthPel	P = 4
independentSegmentDecoding	R
slicesInOrder-NonRect	K = 1
slicesInOrder-Rect	K = 2
slicesNoOrder-NonRect	K = 3
slicesNoOrder-Rect	K = 4
alternateInterVLCMode	S
modifiedQuantizationMode	T
reducedResolutionUpdate	Q
separateVideoBackChannel	N/A
videoBadMBsCap	• PAR • CPCF • CUSTOM
h263Version3Options	N/A

IWF Unsupported Parameters

The following optional SDP parameters are not supported for H.263+ interworking: SQCIF, QCIF, CIF, CIF4, CIF16, CUSTOM, PAR, CPCF.

SIP-H.323 IWF in Video Conferencing Applications

For video conferencing and other video applications, the SBC supports interworking between the H.323 Miscellaneous Commands videoFastUpdatePicture and the SIP INFO containing XML schema for Full Update. The noted H.323 message commands the video encoder to enter fast-update mode.



There is no configuration required for the interworking between these two messages to work.

International Peering with IWF and H.323 Calls

When you do not enable this feature, SIP to H.323 IWF calls default to a National Q.931 Number Type and it is not possible to change it to an International number. This feature allows you to override that behavior by configuring the option **cpnType=X**, where X is an integer that maps to various Q.931 Number Types. When this option is set, Q.931 Number Type for both calling party and called party are updated to the configured value for all outing calls on the h323-stack.

The following is a list of possible **cpnType=X** option values for X:

- 0—Unknown public number
- 1—International public number
- 2—National public number
- 3—Specific public network number
- 4—Public subscriber number
- 5—Public abbreviated number
- 6—Private abbreviated number

International Peering Configuration

You configure this feature as an option in the **h323-stack** configuration.

To configure the cpnType=X option for H323-H323 calls:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **h323-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# h323-config
ORACLE(h323)#
```

4. Type **h323-stacks** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(h323)# h323-stack
ORACLE(h323-stack)#
```

5. Set the options parameter by typing **options**, a Space, the option name **cpnType=x** with a plus sign in front of it, and then press Enter.

```
ORACLE(h323-stack)# options +cpnType=x
```

   If you type **options** without the plus sign, you will overwrite any previously configured options. In order to append the new options to the h323-stack's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

IWF Codec Renegotiation for Audio Sessions

For calls requiring interworking between SIP and H.323, there can be several instances for audio sessions when a mid-call codec change is necessary. These are some examples of when the codec used for voice transportation is necessary:

- Sessions between analog FAX machines that start as regular voice calls but then must use a codec that is fax-signalling tolerant (like transparent G.711) when FAX tones are detected; detection takes place after the call has been answered. The case of modem calls is similar.
- An established call is redirected in one carrier's network either to a different enduser or to a media server. In this case, the party to which the call is redirected might not support the codec used in the redirection. If request for a codec change is carried out at the signalling level, the call can proceed with the party to which the call was redirected.
- Endusers might want to change codecs when they suffer low voice quality.

Both SIP and H.323 provide mechanisms for changing codecs during a call: SIP uses the ReINVITE, and H.323 uses the H.245 Request Mode. Using the option called **processRequestModeForIWF=all** either in an H.323 interface (stack) or an H.323 session agent configuration, you can enable the Oracle Communications Session Border Controller to interwork SIP ReINVITE and H.245 Request Mode requests.

RTN 1976

Codec Request Change from the SIP Side

When a SIP party requests a code change, the Oracle Communications Session Border Controller communicates with the H.323 endpoint to renegotiate support for an updated codec. In this renegotiation, the Oracle Communications Session Border Controller presents codec for use ordered according to the SIP side's preference and one is selected. Then the Oracle Communications Session Border Controller handles opening of a new logical channel that uses the updated codec, and closes the old logical channel (that uses the now-outdated codec). On the SIP side, the Oracle Communications Session Border Controller sends a 200 OK with the necessary RTP port and codec information for the new logical channel.

Codec Request Change from the H.323 Side

When the Oracle Communications Session Border Controller receives a codec request change on the H.323 side of an IWF call, it sends a Re-INVITE to the SIP endpoint containing new codec and information. The Oracle Communications Session Border Controller uses IP address and port information it has cached for the H.323 side of the call for the Re-INVITE since H.245 Request Mode requests do not have this data. If the IP address and port combination should subsequently change (in an OLC from the H.323 side), the Oracle Communications Session Border Controller handles additional INTVITEs on the SIP side to support the change.

Exceptional Cases

When the relevant option in enabled, the Oracle Communications Session Border Controller can handle properly the following cases of codec change:

- When the H.323 side rejects the request mode change, the Oracle Communications Session Border Controller response to the SIP side with a 488 Not Acceptable. Session description and state remain unchanged, and the call continues using the original session description.
- When the H.323 side does not respond to the request mode change within the timeout limitation, the Oracle Communications Session Border Controller releases the call on both sides.
- When the SIP side does not respond to the ReINVITE within in the timeout limitation, the Oracle Communications Session Border Controller releases the call on both sides.
- When the intersection of codec is empty, the Oracle Communications Session Border Controller rejects the codec change on the SIP side with a 488 Not Acceptable and on the H.323 side with an H.245 RequestModeReject. Session description and state remain unchanged, and the call continues using the original session description.
- If the Oracle Communications Session Border Controller does not receive any of the LogicalChannel request or acknowledgement messages, the Oracle Communications Session Border Controller releases the call on both sides.

Note that for protocol timeout errors, the preferred behavior is to release the call on both sides. Timeout errors usually indicate network problems, such as an endpoint being unreachable.

IWF Codec Renogotiation Configuration

You can apply the **processRequestModeForIWF=all** to H.323 interfaces (stacks) and to H.323 session agents (sessions agents for which H.323 has been identified in the **protocol**

parameter). The example below shows you how to enable this option for an H.323 session agent.

To enable IWF codec renegotiation for an H.323 session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter. If you are adding this feature to a pre-existing configuration, you will need to select and edit it.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **options**—Set the options parameter by typing options, a Space, the option name **processRequestModeForIWF=all** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(session-agent)# options +processRequestModeForIWF=all
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save your work.

8

Application Layer Gateway Services

DNS ALG

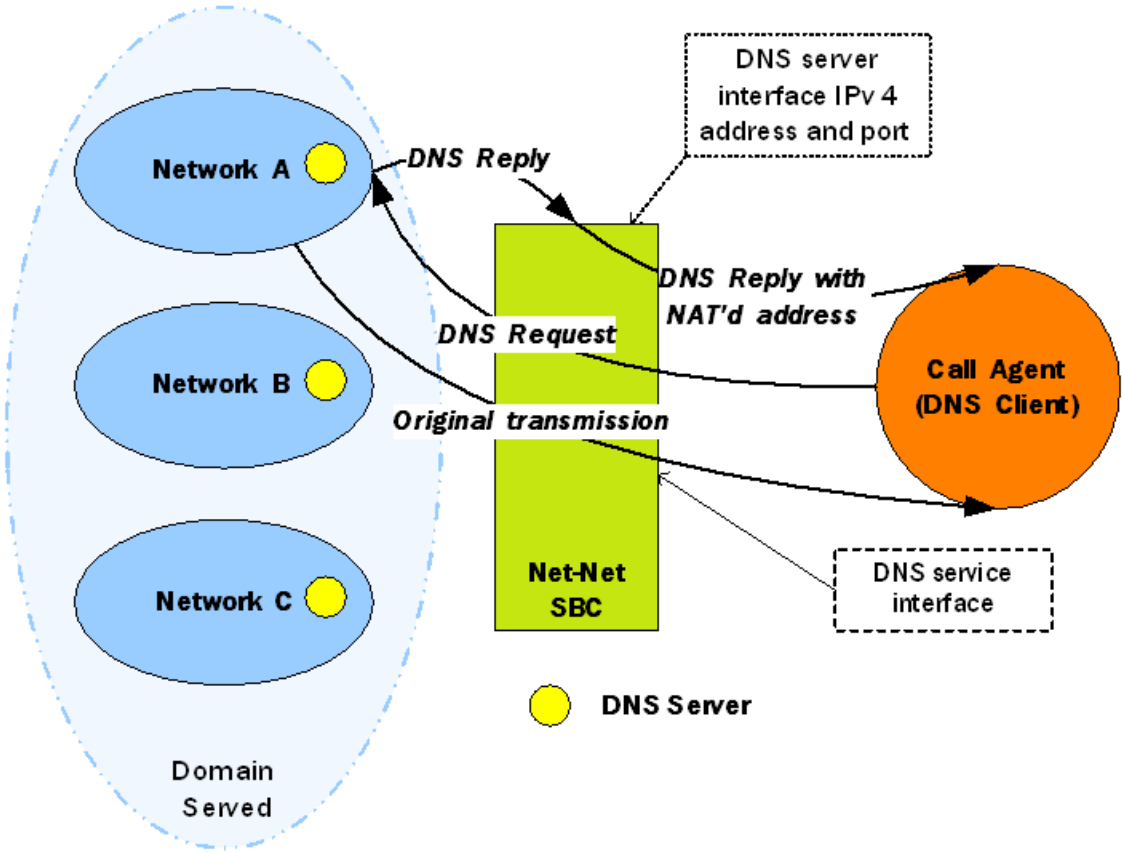
The Oracle Communications Session Border Controller's DNS Application Layer Gateway (ALG) feature provides an application layer gateway for DNS transactions on the Oracle Communications Session Border Controller. With DNS ALG service configured, the Oracle Communications Session Border Controller can support the appearance of multiple DNS servers on one side and a single DNS client on the other.

Overview

DNS ALG service provides an application layer gateway for use with DNS clients. DNS ALG service allows a client to access multiple DNS servers in different networks and provides routing to/from those servers. It also supports flexible address translation of the DNS query/response packets. These functions allow the DNS client to query many different domains from a single DNS server instance on the client side of the network.

The Oracle Communications Session Border Controller's DNS ALG service is commonly used when a DNS client (such as a call agent) needs to authenticate users. In this case, the DNS client that received a message from a certain network would need to authenticate the endpoint in a remote network. Since the DNS client and the sender of the message are on different networks, the Oracle Communications Session Border Controller acts as an intermediary by interoperating with both.

In the following diagram, the DNS client has received a message from an endpoint in Network A. Since the DNS client is in a different realm, however, the DNS client receives the message after the Oracle Communications Session Border Controller has performed address translation. Then the DNS client initiates a DNS query on the translated address. The Oracle Communications Session Border Controller forwards the DNS request to the DNS server in Network A, using the domain suffix to find the appropriate server. Network A's DNS server returns a response containing its IPv4 address, and then the Oracle Communications Session Border Controller takes that reply and performs a NAT on the private address. The private address is turned into a public one that the DNS client can use to authenticate the endpoint.



Configuring DNS ALG Service

This section tells you how to access and set the values you need depending on the configuration mechanism you choose. It also provides sample configurations for your reference.

Configuring DNS ALG service requires that you carry out two main procedures:

- Setting the name, realm, and DNS service IP interfaces
- Setting the appropriate parameters for DNS servers to use in other realms

Before You Configure

Before you begin to configure DNS ALG service on the Oracle Communications Session Border Controller, complete the following steps.

1. Configure the client realm that you are going to use in the main DNS ALG profile and note its name to use in this chapter's configuration process.
2. Configure the server realm that contains the DNS servers and note its name to use in this chapter's configuration process.
3. Determine the domain suffixes for the network where the DNS servers are located so that you can enter them in the domain suffix parameter.
4. Devise the NAT scheme that you want to use when the DNS reply transits the Oracle Communications Session Border Controller.

DNS ALG Service Name Configuration

This section explains how to configure the name of the DNS ALG service you are configuring and set its realm.

To add DNS ALG service:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **dns-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# dns-config
   ORACLE(dns-config)#
   ```

   From this point, you can configure DNS ALG parameters and access this configuration's DNS server subelement. To view all DNS ALG service parameters and the DNS server subelement, enter a **?** at the system prompt.

   ```
   dns-config
           client-realm
           description                    dns-alg1
           client-address-list
           last-modified-date             2005-02-15 10:50:07
           server-dns-attributes
                   server-realm
                   domain-suffix
                   server-address-list
                   source-address
                   source-port                    53
                   transaction-timeout            10
                           address-translation
                                   server-prefix                  10.3.0.0/16
                                   client-prefix
   192.168.0.0/16
   ```

Identity Realm and Interface Addresses

To configure the identity, realm, and IPv4 interface addresses for your DNS ALG profile:

1. **description**—Set a name for the DNS ALG profile using any combination of characters entered without spaces. You can also enter any combination with spaces if you enclose the whole value in quotation marks. For example: DNS ALG service.
2. **client-realm**—Enter the name of the realm from which DNS queries are received. If you do not set this parameter, the DNS ALG service will not work.

3. **client-address-list**—Configure a list of one or more addresses for the DNS server interface. These are the addresses on the Oracle Communications Session Border Controller to which DNS clients send queries.

   To enter one address in this list, type client-address-list at the system prompt, a Space, the IPv4 address, and then press Enter

   ```
   ORACLE(dns-config)# client-address-list 192.168.0.2
   ```

   To enter more than one address in this list, type **client-address-list** at the system prompt, and a Space. Then type an open parenthesis (( ), each IPv4 address you want to use separated by a Space, and closed parenthesis ( )), and then press Enter.

   ```
   ORACLE(dns-config)# client-address-list (192.168.0.2 196.168.1.1
   192.168.1.2)
   ```

DNS Server Attributes

To configure attributes for the DNS servers that you want to use in the DNS ALG profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **dns-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# dns-config
   ```

4. Type **server-dns-attributes** and then press Enter.

   ```
   ORACLE(dns-config)# server-dns-attributes
   ```

   From this point, you can configure DNS server parameters. To see all parameters for the DNS server, enter a **?** at the system prompt.

5. **server-realm**—Enter the name of the realm in which the DNS server is located. This value is the name of a configured realm.
6. **domain-suffix**—Enter a list of one or more domain suffixes to indicate the domains you want to serve. These values are matched when a request is sent to a specific DNS server. If you leave this list empty (default), then your configuration will not work.

   > **Note:**
   >
   > If you want to use a wildcard value, you can start your entry to an asterisk (*) (e.g. *.com). You can also start this value with a dot (e.g., .com).

To enter one address in this list, type client-address-list at the system prompt, a Space, the domain suffix, and then press Enter

```
ORACLE(server-dns-attributes)# domain-suffix acmepacket.com
```

To enter more than one address in this list, type **domain-suffix** at the system prompt, and a Space. Then type an open parenthesis (( ), each IPv4 address you want to use separated by a Space, and closed parenthesis ( )), and then press Enter.

```
ORACLE(server-dns-attributes)# domain-suffix (acmepacket.com
acmepacket1.com acmepacket2.com)
```

7. **server-address-list**—Enter a list of one or more DNS IPv4 addresses for DNS servers. These DNS servers can be used for the domains you specified in the domain suffix parameter. Each domain can have several DNS servers associated with it, and so you can populate this list with multiple IPv4 addresses. If you leave this list empty (default), your configuration will not work.
8. **source-address**—Enter the IPv4 address for the DNS client interface on the Oracle Communications Session Border Controller. If you leave this parameter empty (default), your configuration will not work.
9. **source-port**—Enter the number of the port for the DNS client interface on the Oracle Communications Session Border Controller. The default value is **53**. The valid range is:
   - Minimum—1025
   - Maximum—65535
10. **transaction-timeout**—Enter the time in seconds that the ALG should keep information to map a DNS server response back to the appropriate client request. After the transaction times out, further response to the original request will be discarded. The default value is **10**. The valid range is:
    - Minimum—0
    - Maximum—999999999
11. **address-translation**—Enter a list of address translations that define the NAT function for the DNS servers.

    You can access the NAT parameters for the DNS servers by typing address-translation and pressing enter within the DNS server attributes configuration.

    ```
    ORACLE(dns-config)# server-dns-attributes
    ORACLE(server-dns-attributes)# address-translation
    ```

    To configure the NAT, enter two values:

    - **server-prefix**: address/prefix that will be returned by the DNS server
    - **client-prefix**: address/prefix that to which a response is returned

      Each of these is a two-part value:

    - IPv4 address
    - Number of bits indicating how much of the IPv4 address to match

      If you do not specify the number of bits, then all 32 bits of the IPv4 address will be used for matching. If you set the number of bits to 0, then the address will simply be copied.

For example, if you set the server prefix to 10.3.17.2/16 and the client prefix to 192.168.0.0/16, then the Oracle Communications Session Border Controller will return an address of 192.168.17.2 to the DNS client.

```
ORACLE(server-dns-attributes)# address-translation
ORACLE(address-translation)# server-prefix 10.3.17.2/16
ORACLE(address-translation)# client-prefix 192.168.0.0/16
```

DNS Transaction Timeout

To provide resiliency during DNS server failover, you can now enable a transaction timeout for DNS servers. If you have endpoints that are only capable of being configured with a single DNS server, this can allow DNS queries to be sent to the next configured server—even when contacting the Oracle Communications Session Border Controller's DNS ALG on a single IP address. So when the first server in the list times out, the request is sent to the next server in the list.

The Oracle Communications Session Border Controller uses the transaction timeout value set in the **dns-server-attributes** configuration (part of the **dns-config**).

DNS Transaction Timeout Configuration

To enable the DNS transaction timeout:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **dnsalg-server-failover**—Change this parameter from **disabled** (default) to **enabled** to allow DNS queries to be sent to the next configured server—even when contacting the Oracle Communications Session Border Controller's DNS ALG on a single IP address. So when the first server in the list times out, the request is sent to the next server in the list. The Oracle Communications Session Border Controller uses the transaction timeout value set in the **dns-server-attributes** configuration (part of the **dns-config**).
5. Save your work.

DNS Server Operation States

After the first failed attempt to reach a DNS server, The Oracle Communications Session Border Controller places it in a Time Out state. The server stays in Time Out state for 30 seconds. The Oracle Communications Session Border Controller does not send DNS queries to a server in Time Out state. Instead, it directs queries to the next DNS server in the server-

address-list. After 30 seconds, the DNS server goes back to an In Service state and the Oracle Communications Session Border Controller sends queries to it.

If a DNS server fails to respond to 5 consecutive queries, it goes into Out of Service (OOS) state. The Oracle Communications Session Border Controller directs all subsequent queries to the first In Service server. The Oracle Communications Session Border Controller returns OOS DNS servers to In Service state after 600 seconds and repeats the cycle above. If, for any given call, all configured DNS servers are OOS, the Oracle Communications Session Border Controller fails the call.

DNS Entry Maximum TTL Configuration for DNS ALG

Set parameter for DNS entry maximum time to live (TTL) value in **dns-config** for the DNS ALG feature.

dns-config

1. Access the **dns-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# dns-config
ORACLE(dns-config)# select
```

2. Select the **dns-config** object to edit.

```
ORACLE(dns-config)# select
client-realm:

ORACLE(dns-config)#
```

3. **dns-max-ttl**— set to the maximum time for a DNS record to remain in cache.
   - **Minimum: 30**— The lowest value to which the **dns-max-ttl** parameter can be set (in seconds)
   - **Maximum: 2073600**— The maximum value (in seconds) for which the **dns-max-ttl** parameter can be set.
   - **Default: 86400**— The value in seconds which the system uses by default.
4. Type **done** to save your configuration.

DNS ALG Message Throttling

You can limit DNS ALG-bound throughput by using the dns-alg-constraints configuration element. Message throttling is performed on request messages, and since DNS-ALG is transaction stateful, the responses are automatically throttled. Once a dns-alg-constraints element is configured, it can be referenced by a DNS configuration object. This allows users to create constraint profiles and apply them to multiple DNS configuration objects.

When any of the constraints are exceeded, the DNS server's status changes to Constraints Exceeded and requests are rejected with a 503 error message. The server, via the ALG, remains in that state until the time-to-resume parameter period ends. When this period ends, the Oracle Communications Session Border Controller re-examines the server's traffic with respect to the current burst or sustain window and the state is determined again, along with the decision to readmit traffic.

Bursty Traffic Throttling

You can create separate inbound and outbound maximum burst rates. First configure a **burst rate window** in seconds. Then configure the **max inbound burst rate** and **max outbound burst rate** parameters for the traffic you wish to constrain. In addition you can set an overall maximum burst rate with the **max burst rate** parameter.

Regardless of the inbound or outbound burst rates' headroom before exceeding the message constraints, if the total in/out traffic exceeds the **max burst rate** (when configured), the target DNS server is set to Constraints Exceeded status and taken out of service.

Sustained Traffic Throttling

You can create separate inbound and outbound maximum sustained traffic rates. First configure a **sustain rate window** in seconds. Then configure the **max inbound sustain rate** and **max outbound sustain rate** parameters. In addition you can set an overall sustain rate with the **max sustain rate** parameter.

Regardless of the inbound or outbound sustained traffic rates' headroom before exceeding the message constraints, if the total in/out traffic exceeds the **max sustained rate** (when configured), the target DNS server is set to Constraints Exceeded status and taken out of service.

Maximum Latency

The max-latency parameter sets a threshold under which requests must be responded to. If this value is exceeded, the DNS server is put into the Constraints Exceeded State.

DNS ALG Constraints Configuration

To configure the DNS ALG Constraints object:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-manager path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **dnsalg-constraints** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# dnsalg-constraints
   ORACLE(dnsalg-constraints)#
   ```

4. **name**—Enter a name for this dnsalg-constraints configuration element. You will reference this value from the constraint-name parameter in the dns-config configuration element.
5. **state**—Set this parameter to enabled to enable this configuration element.
6. **max-burst-rate**—Enter the maximum number of messages that can pass through the system in the burst rate window before setting the element to Constraints Exceeded.

7. **max-inbound-burst-rate**—Enter the maximum number of inbound messages received by the referencing element within the burst rate window before setting the element to Constraints Exceeded.
8. **max-outbound-burst-rate**—Enter the maximum number of outbound messages forwarded from the referencing element within the burst rate window before setting the element to Constraints Exceeded.
9. **burst-rate-window**—Enter the number of seconds during which to count messages toward a maximum burst rate.
10. **max-sustain-rate**—Enter the maximum number of messages that can pass through the system in the sustained rate window before setting the element to Constraints Exceeded.
11. **max-inbound-sustain-rate**—Enter the maximum number of inbound messages received by the referencing element within the sustained rate window before setting the element to Constraints Exceeded.
12. **max-outbound-sustain-rate**—Enter the maximum number of outbound messages forwarded from the referencing element within the sustained rate window before setting the element to Constraints Exceeded.
13. **sustain-rate-window**—Enter the number of seconds during which to count messages toward a maximum sustained rate.
14. **max-latency**—Enter the time the Oracle Communications Session Border Controller waits to receive a response to a request. After this value is exceeded, the DNS server is placed in Constraints Exceeded state.
15. **time-to-resume**—Enter the number of seconds that the referencing element stays in Constraints Exceeded state and rejects messages before it returns to service.
16. Type **done** when finished.

Applying Traffic Constraints

To apply DNS ALG constraints to an existing DNS config object:

1. Type exit to return to the media-manager path.

```
ORACLE(dnsalg-constraints)# exit
ORACLE(media-manager)#
```

2. Type **dns-config** and **select** an existing configuration element.

```
ORACLE(media-manager)# dns-config
ORACLE(dns-config)# select
<realm-id>:
1: realm01
selection: 1
```

3. **constraint-name**—Enter the name of an existing diameter-director-constraints configuration element.
4. Type **done** when finished.

ACLI DNS ALG Statistics

The Oracle Communications Session Border Controller provides comprehensive DNS ALG monitoring with the show dns-alg command. This command is entered as:

```
ORACLE# show dnsalg [lookup | rate | stats | status]
```

Entered without any arguments, this command shows a summary of all dnsalg agents.

You must set the **extra-dnsalg-stats** parameter in the dns-config to enabled to collet these statistics.

```
ORACLE#show dnsalg stats
                ---Queries----  --Successful--  ---NotFound---  ---TimedOut--
DNS ALG Realm   Current  Total  Current  Total  Current  Total  Current  Total
net172          0        1      0        1      0       0       0        0
net192          0        0      0        0      0       0       0        0
Avg Latency=0.000 for 0
Max Latency=0.000
Rate =0.0
```

Other Show commands

```
ORACLE# show dnsalg stats <realm-id>
ORACLE# show dnsalg lookup <client realm>
ORACLE# show dnsalg status <dnsalg-client-realm>
```

SNMP DNS ALG Statistics and Traps

The AP-DNSALG-MIB is defined to provide standard-compliant object definitions and descriptions for the SNMP information provided by Oracle Communications Session Border Controller for dns-alg query related information. This MIB corresponds to the related ACLI show commands.

By setting the **trap-on-status-change** parameter in the dns-config to enabled, the following traps become active on the system:

Traps	Description
apDnsAlgStatusChangeTrap apDnsAlgStatusChangeClearTrap	This trap is set and clears when any DNS-config object changes its operating status.
apDnsAlgConstraintStateChangeTrap apDnsAlgConstraintStateChangeClearTrap	This trap is set and clears when any dns-config object enters constraints exceeded state.

Traps	Description
apDnsAlgSvrConstraintStateChangeTrap apDnsAlgSvrConstraintStateChangeClearTrap	This trap is set and clears when any single DNS server enters constraints exceeded state.

Refer to the MIB ZIP archive to view the AP-DNSALG-MIB MIB file.

9

Session Routing and Load Balancing

This chapter explains how to configure session routing and load balancing for SIP and H.323 services. It contains information about configuring session agents and session agent groups, as well as local policies that can be used for routing SIP or H.323 signals.

Routing Overview

This section provides an overview of routing SIP and H.323 sessions when using the Oracle Communications Session Border Controller. The SBC chooses the next hop through the network for each SIP and H.323 session based on information received from routing policies and constraints. Routing policies can be as simple as routing all traffic to a proxy or routing all traffic from one network to another. Routing policies can also be more detailed, using constraints to manage the volume and rate of traffic that can be routed to a specific network. For example, you can manage volume and rate of traffic to enable the SBC to load balance and route around softswitch failures.

When a call request arrives at the SBC, a decision making process then occurs to determine whether the message is coming from a session agent. If so, the SBC checks whether that session agent is authorized to make the call. Local policy is then checked to determine where to send the message on to.

Session Agents Session Groups and Local Policy

When you configure session routing for SIP and H.323, you can use session agents, session agent groups and local policies to define routing. (Using session agents and session agent groups is not required.)

- session agent: defines a signaling endpoint. It is a next hop signaling entity that can be configured to apply traffic shaping attributes.
- session agent group (SAG): contains individual session agents. Members of a SAG are logically equivalent (although they might vary in their individual constraints) and can be used interchangeably.
  You apply an allocation strategy to the SAG to allocate traffic across the group members. Session agent groups also assist in load balancing among session agents.
- local policy: indicates where session request messages, such as SIP INVITES, are routed and/or forwarded. You use a local policy to set a preference for selecting one route over another.

Another element of routing is the realm. Realms are used when an SBC communicates with multiple network elements over a shared intermediate connection. Defining realms allows sessions to go through a connection point between the two networks.

When you configure a realm, you give it an identifier, which stores the name of the realm associated with a sip-interface. The realm identifier value is also needed when you configure session agents and local policies. You can associate a realm with a session agent to identify the realm for sessions coming from or going to the session agent. You also need the realm identifier when you configure local policy to identify the egress realm (realm of the next hop).

About Session Agents

This section describes session agents. A session agent defines a signaling endpoint. It is a next hop signaling entity that can be configured to apply traffic shaping attributes. Service elements such as gateways, softswitches, and gatekeepers are defined automatically within the Oracle Communications Session Border Controller as session agents. For each session agent, concurrent session capacity and rate attributes can be defined. You can group session agents together into session agent groups and apply allocation strategies to achieve traffic load balancing.

You can assign a media profile to a session agent and indicate whether the transport protocol is SIP or H.323. If the protocol is H.323, you need to indicate whether the session agent is a gateway or a gatekeeper.

You can configure a set of attributes and constraints for each session agent to support the following:

- session access control: Oracle Communications Session Border Controller only accepts requests from configured session agents
- session admission control (concurrent sessions): Oracle Communications Session Border Controller limits the number of concurrent inbound and outbound sessions for any known service element.
- session agent load balancing: session agents are loaded based on their capacity and the allocation strategy specified in the session agent group.
- session (call) gapping: Oracle Communications Session Border Controller polices the rate of session attempts to send to and receive from a specific session agent.
- Static TCP source port—By default, the SBC allows ephemeral TCP port assignment of the source port used by the SBC when connecting to a **session-agent**. Some environments preclude this ephemeral source port selection. When deployed in these environments, you can configure the SBC to use a static TCP port when connecting to a **session-agent** by enabling the **static-tcp-source-port** parameter in the applicable **session-agent**. Enabling this feature requires that you save and activate your changes, then reboot the SBC.
  Consider the following limitations when deploying this Static TCP Source Port Feature:
  - The **static-tcp-source-port** feature only supports a single connection to the **session-agent**. This feature causes multiple connections to a **session-agent** to use the same port, thereby causing the connection to fail.
  - You cannot configure a **sip-interface** with the same IP address and port number you use in any **static-tcp-source-port** configuration. This generates a socket bind error, preventing the interface from connecting. An example of this configuration error is the use of **static-tcp-source-port** 5061 to connect to the SA and a TLS port also configured to 5061.
  - The SBC does not support reconnecting an existing session to a **session-agent** using a **static-tcp-source-port** after a high availability switchover. This scenario requires that the existing session be terminated and a new one started.
  - The SBC does not support reconnecting an existing session to a **session-agent** using a **static-tcp-source-port** after you reboot it. This scenario requires that the existing session be terminated and a new one started.

> **Note:**
>
> A new connection may become active very quickly, for example directly after you activate your **static-tcp-source-port** configuration and before you have a chance to reboot the system. This scenario would also require that this existing session be terminated and a new one started.

– The **inactive-conn-timeout** parameter on a **sip-interface** specifies how long the system waits before it tears down an inactive connection. When configured to a non-zero setting, this parameter also impacts a **session-agent**, including those configured with a **static-tcp-source-port**, that are operating over such a **sip-interface**. Expiry of this timer causes the TCP connection of the **session-agent** to enter TCP TIME-WAIT state, resulting in the connection to the **session-agent** being unavailable for 60 seconds.

SIP Session Agents

SIP session agents can include the following:

- softswitches
- SIP proxies
- application servers
- SIP gateways
- SIP endpoints

In addition to functioning as a single logical next hop for a signaling message (for example, where a SIP INVITE is forwarded), session agents can provide information about next or previous hops for packets in a SIP agent, including providing a list of equivalent next hops.

You can use the session agent to describe one or more SIP next or previous hops. Through the configured carriers list, you can identify the preferred carriers to use for traffic coming from the session agent. This set of carriers will be matched against the local policy for requests coming from the session agent. You can also set constraints for specific hops.

Session Agent Status Based on SIP Response

The Oracle Communications Session Border Controller can take session agents out of service based on SIP response codes that you configure, and you can also configure SIP response codes that will keep the session agent in service.

With this feature disabled, the SBC determines session agents' health by sending them ping messages using a SIP method that you configure. Commonly, the method is an OPTIONS request. If it receives any response from the session agent, then the SBC deems that session agent available for use.

However, issues can arise when session agents are administratively out of service, but able to respond to OPTIONs requests. A session agent like this might only respond with a 200 OK when in service, and send a 4xx or 5xx message otherwise.

The session agent status feature lets you set the SIP response message that either takes a session agent out of service or allows it to remain in service when it responds to the SBC's ping request.

Details of this feature are as follows:

- The SBC only considers a session agent in service when it responds to a request method you set with the final response code that you also set. If a final response code is set, then provisional responses are not used for determining whether or not to take a session agent out of service. If the SBC receives a final response code that does not match the session agent configuration, it treats the session agent as though it had not responded.
- The SBC takes a session agent out of service when it receives an error response for dialog creating request with a response code listed in the new **out-service-response-codes** parameter.

In the case where the session agent's response has a Retry-After header, the SBC tries to bring the session agent back into service after the period of time specified in the header. To do so, it sends another ping request.

There are two lists you can configure in the session agent configuration to determine status:

- In-service list—Set in the ACLI **ping-in-service-response-codes** parameter, this list defines the response codes that keep a session agent in service when they appear in its response to the SBC's ping request. Furthermore, the SBC takes the session agent out of service should a response code be used that does not appear on this list.
- Out-of-service list—Set in the ACLI **out-service-response-codes** parameter, this list defines the response codes that take a session agent out of service when they appear in its response to the SBC's ping request or any dialog-creating request.

When the SBC receives a session agent's response to its ping request, it first checks to see if there is an in-service list of responses configured for that session agent. If the list is configured and the SBC determines that there is a match, the session agent is deemed in service. Otherwise it takes the session agent out of service. In this way, the in-service list takes precedence over the out-of-service list. If you configure the in-service list, then the SBC ignores the out-of-service list.

If there is no list of in-service responses for the session agent, then the SBC checks the out of service list. If it is configured and the SBC determines that there is a match, the SBC removes that session agent from service. If there is no match, then the session agent is deemed in service.

SIP Pings from Multiple Realms to Global SAs

You can configure the SBC to send SIP OPTION pings to multiple physical agents through multiple realms using a global **session-agent** (SA). This feature monitors status of global SA targets on the system, supporting applicable deployments including UCaaS topologies that include multiple tenancies. You enable this feature by configuring the **session-agent** to act as a global session agent, and each **realm** to support status monitoring.

You can configure a global **session-agent** to operate with multiple physical agents instead of a single physical agent using DHCP and/or target lists. This creates a list of targets that can reside in multiple realms for forwarding. Using global session agents can cause problems when the target devices reside in multiple networks, some of which might not be reachable. If the system is unaware of the status of each physical device, it cannot reliably handle all calls to or from each device. You can resolve these problems using this feature.

> **Note:**
>
> This feature can impact system performance due to the system issuing multiple OPTIONS instead of a single OPTION. You can mitigate any performance issues using the ping-interval and contacting Oracle support.

With or without this feature, the SBC supports multi-tenancy use cases using nested realms and/or multiple realms. But by default, the SBC cannot show the status of the physical session agents devices connected to each realm correctly. Even though a physical session agent might be active, the SBC may show it as inactive. Option pings are needed to monitor this status.

This feature allows global session agents to support SIP ping options for both a single SIP interface with nested realms and for multiple realms using their own separate **sip-interface**:

- Nested Realms—These can be multiple child realms within a single parent realm as a hierarchical realm group. In this case, the **sip-interface** is associated with the parent realm only.
- Multiple Realms/**sip-interfaces**—These are similar to nested realms, with the exception that each customer hosted on the SBC is separated, using their own **realm**, **sip-interface**, certificate record and so forth.

Feature configuration includes configuring a **session-agent** as a global Session Agent and defining the list of multiple physical target devices:

- Wildcarded (*) **realm-id**:
  - Set the **realm-id** parameter to the wildcard (*) character. This establishes the element as a global session agent.
  - Configure the **egress-realm-id** with the list of realms that comprise all tenancies. The system sets all realms in the list to an equal priority, ensuring the system sends the ping tests out every realm.
  - Enable the **ping-method**, **ping-interval**, and **ping-all-address** configurations.

  Using this configuration, the **session-agent** presents all the egress realms to the system, which pings the physical global session agent through those realms.
- FQDN for **hostname**:
  - Configure the **hostname** parameter with an FQDN.
  - Enable **ping-method**, **ping-interval**, and **ping-all-address** configurations.

  When the **hostname** resolves to multiple IP addresses, the system sends SIP OPTIONS via all applicable realms to all the IPs resolved for that hostname.
- IP address in **address** for Nested Realms:
  - Configure the **egress-realm-id** with the list of realms that comprise all tenancies. The system sets all realms in the list to an equal priority, ensuring the system sends the ping tests out every realm.
  - Configure the **address** with a single IP address.
  - Enable **ping-method**, **ping-interval**, and **ping-all-address** configurations.

  This configuration results in the SA sending out Options from all realms to that IP.

> **Note:**
>
> The system throws a **verify-config** error if you set an **egress-realm-id** parameter with a non-existent realm.

All of these configurations allow the SBC to use SIP ping (keepalives) between a global **session-agent** and multiple realms. The system then tracks responses from the target agents in each realm. This information allows the architecture to monitor connection health between the SBC and all customer tenancies during call processing.

In addition to status monitoring, the feature supports vendor deployment requirement of multi-tenant UCaaS environments, such as MSFT Teams and Cisco Webex, that require your SIP pings contain the FQDN of the SBC in a contact header. Those infrastructures use the FQDN to determine the next hop when sending outbound ping responses. You configure the SBC for this on a per-realm basis by setting the **multi-tenancy-fqdn** parameter in each realm to your desired hostname. These deployments use this information to monitor the connection health between the SBC and individual customer tenancies. Because this feature allows the system to generate SIP ping options for multiple realms, each realm can be capable of inserting your configured FQDN for every global **session-agent** target.

Guidelines for determining which FQDN the system inserts into the contact header include with respect to your **teams-fqdn-in-uri** configuration include:

- If you enable the **teams-fqdn-in-uri** parameter on the realm and set either the **teams-fqdn** FQDN or the **fqdn-hostname** FQDN, the system inserts one of those FQDNs in the applicable contact header. If you set both of these FQDNs, the system uses the **fqdn-hostname**.
  This remains true when using this **multi-tenancy-fqdn** feature. If you set both the **fqdn-hostname** and the **multi-tenancy-fqdn**, the system uses the **fqdn-hostname**.
- Regardless of whether you enable or disable the **teams-fqdn-in-uri** parameter, if you have only configured the **multi-tenancy-fqdn** hostname, the system uses the **multi-tenancy-fqdn** to build the contact header.

> **Note:**
>
> This feature operates such that the system monitors each sip-interface/realm defined in the **egress-realm-id** list independently. So, although the Global SA shows In-Service if any 1 realm's connection is active, the system still selects call routes based on the availability between the specific egress realm and the SA.

Downgrade Caveat

You should clear all configuration lists from all **egress-realm-id** parameters prior to downgrading to a version that does not support this feature. If not, older software versions set lists as strings, generating **verify-config** errors.

Reporting

You can use the **show sipd agents <SA>** command to verify this feature's operation. This command displays the status of physical targets, including when your connectivity targets multiple realms. The system determines each status by pinging the multiple destinations established by a DNS SRV resolved **session-agent** or multiple **session-agent** configurations. The output below shows status for two FQDN-resolved **session-agent** targets residing within separate realms.

```
ORACLE#show sipd agents <DNS SRV SA Name>
14:05:10-36
….
Session Agent < DNS SRV SA Name > [Status]
Destination: <IP1>  Source : <Realm1>  <Status>
Destination: <IP1>  Source : <Realm2>  <Status>
Destination: <IP2>  Source : <Realm1>  <Status>
Destination: <IP2>  Source : <Realm2>  <Status>
```

When deployed in HA mode, this status is replicated on the standby SBC.

> **Note:**
>
> Oracle has tested this feature with a moderate range of realms (100-120) along with 10 DNS SRV SA (having a maximum of 4 IP addresses for each of 40 SAs, using a ping-interval 30 secs.

SIP Session Agent Continuous Ping

You can configure the Oracle Communications Session Border Controller to use either a keep-alive or continuous method for pinging SIP session agents to determine their health—i.e., whether or not the SBC should route requests to them.

To summarize the two methods:

- keep-alive— SBC sends a ping message of a type you configure to the session agent in the absence of regular traffic.
- continuous—The SBC sends a ping message regardless of traffic state (regular or irregular); the SBC regularly sends a ping sent based on the configured ping interval timer.

By sending ping messages, the SBC monitors session agents' health and can determine whether or not to take a session out of service (OOS), leave it in service, or bring it back into service after being OOS.

When you set it to use the keep-alive mode of pinging, the SBC starts sending a configured ping message to a session agent when traffic for that session agent has become irregular. The SBC only sends the ping if there are no SIP transactions with a session agent over a configurable period of time, to which the session agent's response can have one of the following results:

- Successful response—A successful response is either any SIP response code or any response code not found in the **out-service-response-codes** parameter; these leave the session agent in service. In addition, any successful response or any response in the **ping-in-service-response-codes** parameter can bring a session agent from OOS to in-service status.
- Unsuccessful response—An unsuccessful response is any SIP response code configured in the **out-service-response-codes** parameter and takes the session agent sending it OOS. Because this parameter is blank by default, the SBC considers any SIP response code successful.
- Transaction timeout—A transaction timeout happens when the session agent fails to send a response to the SBC's request, resulting in the session agent's being taken OOS.

Despite the fact that the keep-alive ping mode is a powerful tool for monitoring session agents' health, you might want to use the continuous ping method if you are concerned about the SBC not distinguishing between unsuccessful responses from next-hop session agents and ones from devices downstream from the next-hop session agent. For example, if a SIP hop beyond the session agent responds with a 503 Service Unavailable, the SBC does not detect whether a session agent or the device beyond it generated the response.

When you use the continuous ping method, only the next-hop session agent responds—preventing the request from being sent to downstream devices. The SBC also sends the ping in regular traffic conditions when in continuous ping mode, so it is certain the response comes from the next hop associated with the session agent. And in continuous ping mode, only

entries for the **ping-out-service-response-codes** parameter and transaction timeouts bring session agents OOS.

> **Note:**
>
> The SBC also brings a Session Agent from OOS to in-service if it receives any SIP request from the session agent device.

By default, if the SBC does not receive a response to the ping from the session-agent, it marks it as out-of-service. You can configure the number of ping failures that the SBC can receive before it marks the session-agent as out-of-service. This is achieved by configuring the OPTIONS parameter in the session-agent with a ping-failure value, where value is the number of ping response failures. This is true for both the keep-alive and continuous-ping modes.

For example, set the Options parameter by typing **options**, followed by a Space, the option name: **ping-failure-count=N** (where N is the number of ping response failures before the SA is set to OOS) and press Enter. You may prepend the option parameter with a plus sign to add and not replace this option to the existing realm-config option list.

```
ORACLE (session-agent) #options +ping-failure-count=3
```

The session-agent is set out of service after the third ping response failure.

To remove the ping-failure-count option configuration, enter options **-ping**

```
ORACLE (session-agent) #options -ping
```

SIP SA Continuous Ping Configuration

You can set the ping mode in the session agent or session constraints configuration. The default for the ping-send-mode parameter is keep-alive.

Configure Ping Mode for Session Agents

1. Access the session-agent configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)
   ```

2. If you are adding to an existing configuration, then you will need to select the configuration you want to edit.
3. **ping-send-mode**—If to want to use continuous ping mode to send ping messages to session agents in regular traffic conditions, set this parameter to **continuous**. If you want to use the keep-alive mode, leave this parameter set to **keep-alive** (default).
4. Save and activate your configuration.

Configure Ping Mode for Session Constraints

1. Access the **session-constraints** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-constraints
ORACLE(session-agent)
```

2. If you are adding to an existing configuration, then you will need to select the configuration you want to edit.
3. **ping-send-mode**—If to want to use continuous ping mode to send ping messages to session agents in regular traffic conditions, set this parameter to **continuous**. If you want to use the keep-alive mode, leave this parameter set to **keep-alive** (default).
4. Save and activate your configuration.

Ingress Session Agent Identification

The Oracle Communications Session Border Controller uses the Ingress Session Agent Identification to match the incoming requests to its respective Session Agents. CAC (Call Admission Control) is performed for the inbound traffic based on the originating endpoint of the Session Agents regardless of the IP address.

This feature is supported by the session-router configuration element **session-agent-id-rule** and the session-agent fields **match-identifier** and **associated-agents**.

The **session-agent-id-rule** specifies the SIP header (with or without the parameter) within the ingress message, which is used for the identification purposes. The configuration element contains the fields **name**, **match-header**, **match-parameter** and **uri-type**. The fields **name** and **match-header** are required fields.

The following actions occur when you configure the specific session-agent identifier elements.

Configured Elements	Identification action performed by the rule
uri-type with values: uri_param, uri_header, uri-user, uri-host, uri-port, uri_user_param, uri-display, uri-user-only, uri-phone-number-only	Identifies using URI values
match-header	Performs case-sensitive match on the entire value of the header
match-header and match-parameter	Finds the value of the parameter within the header and perform a case-sensitive match on the value of the parameter.
match-header and uri-type	Looks up the header and parse it as a sip URI followed by a case-sensitive comparison on the uri element specified by the uri-type.
match-parameter and uri-type with values: uri-user, uri-host, uri-port, uri-display, uri-user-only, uri-phone-number-only	The match-parameter is used to match the SIP header parameter.
match-parameter and uri-type with values: uri_param, uri_header, or uri_user_param	The match-parameter is used to specify the header for the parameter in URI.

The system uses two session-agent configuration fields for session-agent identification: **match-identifier** and **associated-agents**. You can configure the **match-identifier** on the

session-agents representing nodes behind the SBC. This determines the messages sent from a particular node without having the IP address.

Match-identifier has two fields of its own: **identifier-rule** and **match-value**. The **identifier-rule** is configured with the name of a session-agent-id-rule and the **match-value** is configured with the (case-sensitive) string value matched with the particular session agent.

About Master Agent and Associated-Agents

The SBC acts as the 'master agent' with a list of associated-agents. Associated-agents contains a list of the session-agents representing the nodes that sit behind the SBC. The Session Router identifies the correct session-agent by iterating through the associated-agents list and looking for a session-agent that has its match-identifier(s) satisfied in the SIP header. Once the exact match is found, it is confirmed that the message arrived from the corresponding node of that session-agent. The agent is now identified and CAC can be successfully be applied at this point.

This look up is performed on the master agent and its associate-agents every time the session or registration constraints are being checked. The system will only attempt to find a match in the selected session-agent's associated-agents list, (i.e. the associated agent list for an associated-agent will not be examined). In the event that no association is found, constraints will be checked and performed on the master session-agent as usual.

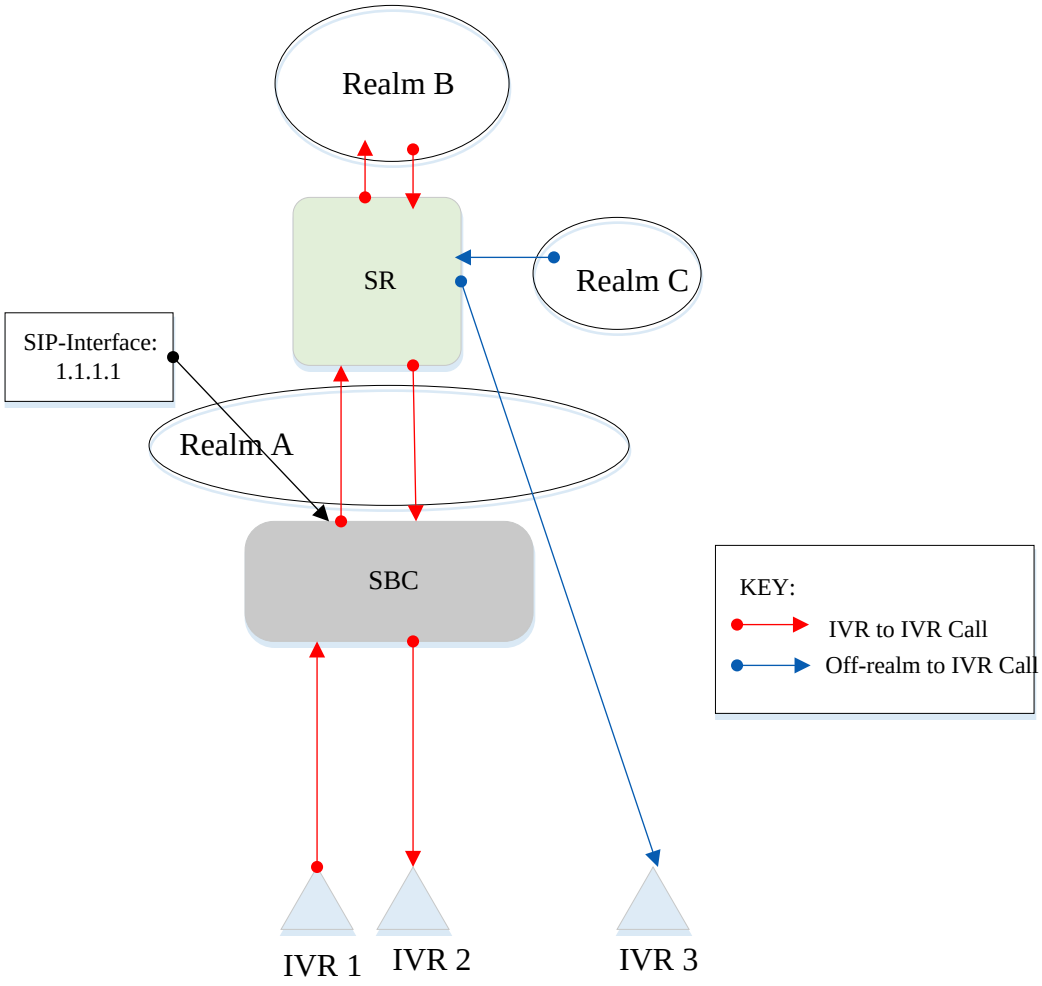
> **Note:**
>
> The comparison between actual value of the SIP header parameter and the match-value is an exact and case-sensitive match.

To use this functionality, the identifiable value must be present in all inbound SIP messages that you intend to associate with a particular session-agent. The identifiable value can be present in any header or parameter, even custom headers and parameters, within the SIP header of the SIP message. You can use the same session-agent-id-rules for multiple session agents if you intend to use the same SIP header or parameter to compare against the match values.

> **Note:**
>
> The session-agents found within the same associated-agents list must have a unique combination of match-identifiers (either identifier-rule or match-value) that distinguishes them from one another.

Session-Agent Identification Example

In the example below, the SBC has a session-agent configured for the SBC, which forwards traffic from three IVRs. IVR to IVR calls flow in the fashion depicted. The Session Router also has session-agents configured which identify the IVRs using DNS lookups. These agents are used when processing off-realm to IVR calls where the Session Router will send messages to the IVR directly. With this feature, the Session Router in the IVR to IVR call (shown in red) can apply IVR1's CAC despite SA-SBC being matched by IP address.




```
Configured Session-Agents on the Session Router

Hostname: SA-SBC
ip: 1.1.1.1
match-identifier: <blank>
associated-agents:
    IVR1.oracle.com
    IVR2.oracle.com
    IVR3.oracle.com

Hostname: IVR1.oracle.com
match-identifier:
    identifier-rule: SA-SBC-Rule
    match-value: ABC1
match-identifier:
    identifier-rule: SA-SBC-Rule2
    match-value: ABC1
associated-agents:
    <blank>

Hostname: IVR2.oracle.com
match-identifier:
```

```
    identifier-rule: SA-SBC-Rule
    match-value: ABC2
match-identifier:
    identifier-rule: SA-SBC-Rule2
    match-value: ABC2
associated-agents:
    <blank>

Hostname: IVR3.oracle.com
match-identifier:
    identifier-rule: SA-SBC-Rule
    match-value: ABC3
match-identifier:
    identifier-rule: SA-SBC-Rule2
    match-value: ABC3
associated-agents:
    <blank>
```

Legacy Configuration

The following is a sample configuration with the session-agent-id-rules and Session Agents. The configuration can be used to perform CAC on session-agent **IVR1.oracle.com** behind the session-agent **SA-SBC.IVR1.oracle.com**. It will be matched using the parameter “test” in P-Identifier in the SIP INVITE.

```
INVITE sip:123402232285116@1.1.1.1:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP 1.1.1.2:5060;branch=z9hG4bKk25du6204otrbrqbcil0.1
Max-Forwards: 69
From: <sip:3611101@1.1.1.4>;tag=as0bd34e4a
To: <sip:0223285116@1.1.1.3:5060;user=phone>
Contact: <sip:3611101@1.1.1.2:5060;transport=udp>
Call-ID: 6850461b04cf07193b480450247960ba@1.1.1.4:5060
CSeq: 102 INVITE
User-Agent: FPBX-2.11.0(11.17.1)
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, NOTIFY, INFO
Content-Type: application/sdp
Content-Length: 215
P-Identifier: temp=ABC;test=1234
TestHeader: sip:test User;id=Abc@oracle.com:1101;
P-Asserted-Identity: tel:2265
```

Session Agent Identifier Rules

Example 1 : Session Agent identifier rule without **uri-type**

```
session-agent-id-rule 1
name                SA-SBC-Rule1
match-header        P-Identifier
match-parameter     test
uri-type
```

This rule will search the SIP header and retrieve the value "1234" from the parameter "test" in match-header P-Identifier.

Example 2: Session Agent identifier rule with **uri-type**

```
session-agent-id-rule 2
name                SA-SBC-Rule2
match-header        TestHeader
match-parameter     id
uri-type            uri-user-param
```

This rule will search the SIP header and retrieve the value "Abc" from match-header TestHeader.

Session Agents

The Session Agent is configured with the session-agent-id-rules. You can add multiple **match-identifiers**. The following configuration contains sample settings for the purpose of providing an example. Your configuration may require different values.

```
<<Example associate-agent>>
hostname                                IVR1.oracle.com
ip-address
port                                    0
state                                   enabled
app-protocol                            SIP
app-type
transport-method                        StaticTCP
match-identifier
              identifier-rules          SA-SBC-Rule1
              match-value               1234
match-identifier
              identifier-rules          SA-SBC-Rule2
              match-value               Abc
associated-agents                       <blank>
…
constraints                             enabled
…


<<Example master-agent>>
hostname                                 SA-SBC
ip-address                               1.1.1.1
port                                     5060
state                                    enabled
app-protocol                             SIP
app-type
transport-method                         StaticTCP
match-identifier                         <blank>
<<This field will not appear if not configured>>
associated-agents                        IVR1.oracle.com
                                         IVR2.oracle.com
                                         IVR3.oracle.com
…
constraints                              enabled
…
```

In the above example, the sip-invite/registration is offered by session-agent SA-SBC. To perform CAC, SA-SBC's associated-agents list is looked up for a match between the match-identifiers and the incoming request. Once a match is found for a particular associated-agent, the existing CAC constraint is applied.

Override Alphanumeric Ordering of Session Agents with same IP address

The Oracle Communications Session Border Controller can associate an incoming call with a Session Agent based upon the **precedence** attribute for systems that have the same IP address, rather than the alphanumeric order of hostname.

To change inbound behavior from an outbound one, customers can configure several Session Agents (SAs) with the same IP address to different hostnames. By default, the system uses alphanumeric order to determine sorting. The attribute **precedence** provides a user-controlled mechanism to determine order for Session Agents. The rules of **precedence** are as follows:

- The default value is zero. This does not activate **precedence**.
- It is configurable with integer value from 1 to 4,294967,295
- The lowest value is considered first, followed by Session Agents with the same IP of increasing values
- After SAs with **precedence** set to a value higher than one are assigned, those with the default of zero are considered (e.g. alphanumeric order sorting is followed)
- Should two Session Agents have the same value of **precedence**, alphanumeric sorting rules apply

This attribute in the Session Agent configuration does not interact with the Session Agent Group construct as the latter is used for egress routing. For this same reason, **precedence** does not apply to surrogate registration, nor registration refresh.

For example, these settings cause the hosts abb123 and aaa456 to be preferred first and aaa123 last:

```
SA1: IP=192.168.139.5, hostname-aaa456, precedence=1
SA1: IP=192.168.139.5, hostname-abb123, precedence=1
SA1: IP=192.168.139.5, hostname-abc123, precedence=33
SA1: IP=192.168.139.5, hostname-aaa123, precedence=44
```

Override Alphanumeric Ordering of Session Agents with Same IP Configuration

The Oracle Communications Session Border Controller can associate an incoming call with a Session Agent based upon the **precedence** attribute for systems that have the same IP address, rather than just the alphanumeric ordering of hostname. Setting **precedence** to zero retains alphanumeric sorting

Enter the prerequisites here (optional).

Enter the context of your task here (optional).

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. Select the **session-agent** object to edit.

```
ORACLE(session-agent)# select
<hostname>:
1: 192.168.100.101:1813

selection: 1
ORACLE(session-agent)#
```

3. **precedence**—Set the importance level of this IP/hostname combination for Session Agents. To be considered for **precedence**, a value of 1 or more is required.
   - **Minimum: 0**
   - **Maximum: 4,294,967,295**
   - **Default: 0**
4. Type **done** to save your configuration.

H.323 Session Agents

H.323 session agents can include the following:

- Gatekeepers
- Gateways
- MCUs

Overlapping H.323 Session Agent IP Address and Port

You can now configure H.323 session agents to use overlapping IP addresses.

H.323 session agents continue are identified by their hostname when used in referencing configuration parameters—such as local policy next hops and session agent group destinations. This is why the hostname must be unique. However, when the Oracle Communications Session Border Controller selects a session agent to use, it chooses the appropriate realm and H.323 stack based on the hostname. This is the case even if there are other session agents with the same IP address and port. Likewise, incoming calls are matched to the session agent based on the incoming realm.

There are no specific parameters to configure in order to enable this feature. For it to work properly, however, each H.323 session agent must be configured with a unique hostname (still the primary index). Otherwise, session agents with non-unique hostnames will overwrite one another.

To create overlapping H.323 session agents, you give each of them a unique hostname, which only serves to identify each individually. The Oracle Communications Session Border Controller subsequently uses this label as the next hop destination in relevant local policy route entries.

Managing Session Agent Traffic

The SBC monitors availability, session load, and session rate for each session agent in real time. The session agent's state is determined by its performance relative to the constraints applied to it and its availability.

The following table lists the conditions that cause the SBC to suspend the routing of traffic to a session agent, along with the criteria for restoring the route.

Constraint Condition	SIP Criteria	H.323 Criteria	Action	Criteria for Resuming
Maximum sessions exceeded	Maximum concurrent SIP sessions exceeded.	Maximum concurrent H.323 sessions exceeded. If the session agent is a gatekeeper and gatekeeper routed mode is not used, this constraint is an aggregate of all the destination gateways. Only maximum outbound sessions are measured.	Session agent is declared in constraint violation state.	Concurrent sessions drop below the maximum sessions value.
Maximum outbound sessions exceeded	Maximum concurrent outbound SIP sessions exceeded.	Maximum concurrent outbound H.323 sessions exceeded. If the session agent is a gatekeeper and gatekeeper routed mode is not used, this constraint is an aggregate of all the destination gateways. Only maximum outbound sessions are measured.	Session agent is declared in constraint violation state.	Concurrent sessions drop below the maximum outbound sessions value.
Maximum burst rate exceeded	Maximum burst rate exceeded in current window.	Maximum burst rate exceeded in current window. If the session agent is a gatekeeper and gatekeeper routed mode is not used, this constraint is an aggregate of all the destination gateways. Only maximum outbound sessions are measured.	Session agent is declared in constraint violation state.	Burst rate in subsequent window drops below maximum burst rate.

Constraint Condition	SIP Criteria	H.323 Criteria	Action	Criteria for Resuming
Maximum sustained rate exceeded	Maximum sustained rate exceeded in current window.	Maximum burst rate exceeded in current window. If the session agent is a gatekeeper and gatekeeper routed mode is not used, this constraint is an aggregate of all the destination gateways. Only maximum outbound sessions are measured.	Session agent is declared in constraint violation state.	Sustained rate in subsequent window drops below the maximum sustained rate.
Session agent unavailable or unresponsive	SIP transaction expire timer expires for any out-of-dialogue request. For example, INVITE, REGISTER, or ping.	Response timer expires. The default is T301=4 seconds. Connect timer expires. The default is T303=32 seconds. If the session agent is a peer gatekeeper, the LRQ response time is used to determine availability. The RAS response timer is 4 seconds.	Session agent is declared in constraint violation state or out-of-service. The time to resume timer starts.	Time to resume timer expires and the Oracle Communications Session Border Controller declares the session agent in-service. or Session agent responds to subsequent pings (SIP only).

Session Agent Groups

Session agent groups contain individual session agents. Members of a session agent group are logically equivalent (although they might vary in their individual constraints) and can be used interchangeably. You can apply allocation strategies to session agent groups.

Examples of session agent groups include the following:

- application server cluster
- media gateway cluster
- softswitch redundant pair
- SIP proxy redundant pair
- gatekeeper redundant pair

Session agent group members do not need to reside in the same domain, network, or realm. The Oracle Communications Session Border Controller can allocate traffic among member session agents regardless of their location. It uses the allocation strategies configured for a SAG to allocate traffic across the group members.

Allocation strategies include the following:

Allocation Strategy	Description
Hunt	Oracle Communications Session Border Controller selects the session agents in the order in which they are configured in the SAG. If the first agent is available, and has not exceeded any defined constraints, all traffic is sent to the first agent. If the first agent is unavailable, or is in violation of constraints, all traffic is sent to the second agent. And so on for all session agents in the SAG. When the first agent returns to service, the traffic is routed back to it.
Round robin	Oracle Communications Session Border Controller selects each session agent in the order in which it is configured, routing a session to each session agent in turn.
Least busy	Oracle Communications Session Border Controller selects the session agent with the least number of active sessions, relative to the maximum outbound sessions or maximum sessions constraints (lowest percent busy) of the session agent.
Proportional distribution	Session agents are loaded proportionately based upon the respective maximum session constraint value configured for each session agent.
Lowest sustained rate	Oracle Communications Session Border Controller routes traffic to the session agent with the lowest sustained session rate, based on observed sustained session rate.

You apply allocation strategies to select which of the session agents that belong to the group should be used. For example, if you apply the Hunt strategy session agents are selected in the order in which they are listed.

Request URI Construction as Sent to SAG-member Session Agent

The Oracle Communications Session Border Controller constructs the request URI for a session agent selected from a session agent group by using the **session-agent**, **hostname** value of the selected **session-agent** target. This default behavior enables features such as trunk groups and ENUM to work properly. However, care must be given when the **hostname** parameter is not a resolvable FQDN. The **sag-target-uri**=<value> option can be used to overcome the default behavior.

The value is either

- **ip** – request URI constructed from **session-agent**, **ip-address**
- **host** – request URI constructed from **session-agent**, **hostname**

This option is global and is configured in the **sip-config** configuration element.

Request URI Construction as Forwarded to SAG-member Session Agent

The Oracle Communications Session Border Controller constructs the request URI for a session agent selected from a session agent group by using the **session-agent**, **hostname** value of the selected **session-agent** target. This default behavior enables features such as trunk groups and ENUM to work properly. However, care must be given when the **hostname** parameter is not a resolvable FQDN. Use the **sag-target-uri=**<value> option to override the default behavior.

The value can be set to either:

- **ip** - request URI constructed from **session-agent**, **ip-address**
- **host** - request URI constructed from **session-agent**, **hostname**

This option is global and is configured in the **sip-config** configuration element.

SIP Session Agent Group Recursion

You can configure a SIP session agent group (SAG) to try all of its session agents rather than to the next-best local policy match if the first session agent in the SAG fails.

With this feature disabled, the Oracle Communications Session Border Controller performs routing by using local policies, trunk group URIs, cached services routes, and local route tables. Local policies and trunk group URIs can use SAGs to find the most appropriate next-hop session agent based on the load balancing scheme you choose for that SAG: round robin, hunt, proportional distribution, least busy, and lowest sustained rate. When it locates a SAG and selects a specific session agent, the Oracle Communications Session Border Controller tries only that single session agent. Instead of trying other members of the SAG, the Oracle Communications Session Border Controller recurses to the local policy that is the next best match. This happens because the Oracle Communications Session Border Controller typically chooses a SAG based on the fact that it has not breached its constraints, but the Oracle Communications Session Border Controller only detects failed call attempts (due to unreachable next hops, unresolved ENUM queries, or SIP 4xx/5xx/6xx failure responses) after it has checked constraints. So the Oracle Communications Session Border Controller only re-routes if there are additional matching local policies.

When you enable SIP SAG recursion, the Oracle Communications Session Border Controller will try the additional session agents in the selected SAG if the previous session agent fails. You can also set specific response codes in the SAG configuration that terminate the recursion. This method of terminating recursion is similar to the Oracle Communications Session Border Controller's ability to stop recursion for SIP interfaces and session agents.

Session agents are selected according to the strategy you set for the SAG, and these affect the way that the Oracle Communications Session Border Controller selects session agents when this feature enabled:

- Round robin and hunt—The Oracle Communications Session Border Controller selects the first session agent according to the strategy, and it selects subsequent session agents based on the order they are entered into the configuration.
- Proportional distribution, least busy, and lowest sustained rate—The Oracle Communications Session Border Controller selects session agents based on the list of session agents sorted by the criteria specified.

You can terminate recursion based on SIP response codes that you enter into the SAG configuration. You can configure a SAG with any SIP response code in the 3xx, 4xx, and 5xx groups. Since you can also set such a list in the session agent configuration, this list is additive to that one so that you can define additional codes for a session agent group with out having to repeat the ones set for a session agent.

Session Agent Group Naming Support

In order to support underscore (_) in Session Agent Group Naming, enable the **support-legacy-hostnames** option under **sip-config**.

Enabling the support-legacy-hostnames option

To enable this option:

```
ORACLE(sip-config)# options +support-legacy-hostnames
```

About Local Policy

This section explains the role of local policy. Local policy lets you indicate where session requests, such as SIP INVITES, should be routed and/or forwarded. You use a local policy to set a preference for selecting one route over another. The local policy contains the following information that affects the routing of the SIP and H.323 signaling messages:

- information in the From header
  Information in the message's From header is matched against the entries in the local policy's from address parameter to determine if the local policy applies.
- list of configured realms
  This list identifies from what realm traffic is coming and is used for routing by ingress realm. The source realms identified in the list must correspond to the valid realm IDs you have already configured
- local policy attributes
  The attributes serve as an expression of preference, a means of selecting one route over another. They contain information such as the next signaling address to use (next hop) or whether you want to select the next hop by codec, the realm of the next hop, and the application protocol to use when sending a message to the next hop. You can also use the attributes to filter specific types of traffic.

Routing Calls by Matching Digits

Local policy routing of a call can be based on matching a sequence of digits against what is defined in the local policy. This sequence refers to the first digits in the (phone) number, matching left to right.

The following examples show how the Oracle Communications Session Border Controller matches an area code or number code against configured local policies.

- If the number or area code being matched is 1234567 (where 123 is an area code), and the from address value in one local policy is 123, and the from address value in another local policy is 12, the Oracle Communications Session Border Controller forwards the call to the server that is defined as the next hop in the local policy with 123 as the from address value.
- If the number or area code being matched is 21234, and the from address value in one local policy is 123, and the from address value in another local policy is 12, the Oracle Communications Session Border Controller will not find a match to either local policy because the first character of the number or area code must match the first character in a from address or to address field.

The following examples show how the Oracle Communications Session Border Controller matches an area or number code against different local policies: the first one has a From address value of 12 and the second has a From address value of 123. The Oracle Communications Session Border Controller chooses the route of the local policy that is configured with the most digits matching the area or number code in its From address and To address fields.

- When the two different local policies route to two different servers, and the area or number code being matched is 123, the Oracle Communications Session Border Controller selects the second local policy based on the From address value of 123.
- When the two different local policies route to two different servers, and the area or number code being matched is 124, the Oracle Communications Session Border Controller selects the first local policy based on the From address value of 12.

SIP and H.323 Interworking

You need to configure local policies, including the requisite local policy attributes, to use the H.323<—>SIP interworking (IWF). Flow progression in H.323<—>SIP traffic depends heavily on the local policies configured for the Oracle Communications Session Border Controller, which determine what protocol is used on the egress side of a session.

You set the application protocol (an local policy attribute option) to instruct the Oracle Communications Session Border Controller to interwork the protocol of an ingress message into a different protocol (H.323<—>SIP or SIP—>H.323) upon its egress to the next hop.

For example, if the application protocol is set to SIP, an inbound H.323 message will be interworked to SIP as it is sent to the next hop. An inbound SIP message would pass to the next hop unaffected. If the application protocol is set to H323, an inbound SIP message will be interworked to H.323 before being sent to the next hop.

Route Preference

The Oracle Communications Session Border Controller builds a list of possible routes based on the source realm and the From-address (From-URI) and To-address (Request-URI), which forms a subset from which preference then decides. Any local policy routes currently outside of the configured time/day are not used, if time/day are set. Also, any local policy routes not on the list of carriers (if carriers is set and the requests has a Carrier header) are not used.

> **Note:**
>
> Source realm is used in the local policy lookup process, but it is not used in route preference calculations.

The Oracle Communications Session Border Controller applies preference to configured local policies in the following order:

1. Cost (cost in local policy attributes) is always given preference.
2. Matching media codec (media profiles option in local policy attributes).
3. Longest matching To address (to address list in local policy).
4. Shortest matching To address (to address list in local policy).
5. Longest matching From address (from address list in local policy).
6. Shortest matching From address (from address list in local policy).
7. Narrowest/strictest day of week specification (days of week option in local policy attributes).
8. Narrowest/strictest time of day specification (start time and end time options in local policy attributes).
9. Wildcard matches (use of an asterisk as a wildcard value for the from address and to address lists in local policy).
10. Wild card matches are given the least preference. A prefix value of 6 is given a higher preference than a prefix value of * even though both prefix values are, in theory, the same length.

DTMF-Style URI Routing

The Oracle Communications Session Border Controller supports the alphanumeric characters a-d, A-D, the asterisk (*), and the ampersand (#) for local policy matching purposes. The Oracle Communications Session Border Controller handles these characters as standards DN (POTS) or FQDN when found in the to-addr (req-uri username) or from-addr (from0uri username for SIP, SIPS, and TEL URIs.

In addition, before performing the lookup match, the Oracle Communications Session Border Controller strips characters that provide ease-of-reading separation. For example, if the Oracle Communications Session Border Controller were to receive a req-uri containing tel:a-#1-781-328-5555, it would treat it as tel:a#17813285555.

SIP Routing

This section describes SIP session routing. When routing SIP call requests, the Oracle Communications Session Border Controller communicates with other SIP entities, such as SIP user devices, other SIP proxies, and so on, to decide what SIP-based network resource each session should visit next. The Oracle Communications Session Border Controller processes SIP call requests and forwards the requests to the destination endpoints to establish, maintain, and terminate real-time multimedia sessions.

Certain items in the messages are matched with the content of the local policy, within constraints set by the previous hop session agent, and the SIP configuration information (for example, carrier preferences) to determine a set of applicable next hop destinations.

The sending session agent is validated as either a configured session agent or a valid entry in a user cache. If the session INVITATION does not match any registering user, the SIP proxy determines the destination for routing the session INVITATION.

Limiting Route Selection Options for SIP

You can configure the local policy to use the single most-preferred route. And you can configure the SIP configuration max routes option to restrict the number of routes which can be selected from a local policy lookup:

- A **max-routes=1** value limits the Oracle Communications Session Border Controller to only trying the first route from the list of available preferred routes.
- A **max-routes=0** value or no **max-routes** value configured in the options field allows the Oracle Communications Session Border Controller to use all of the routes available to it.

A Oracle Communications Session Border Controller configured for H.323 architectures will have access to all of the routes it looks up by default.

About Loose Routing

According to RFC 3261, a proxy is loose routing if it follows the procedures defined in the specification for processing of the Route header field. These procedures separate the destination of the request (present in the Request-URI) from the set of proxies that need to be visited along the way (present in the Route header field).

When the SIP NAT's route home proxy field is set to enabled, the Oracle Communications Session Border Controller looks for a session agent that matches the home proxy address and checks the loose routing field value. If the loose routing is:

- **enabled**—A Route header is included in the outgoing request in accordance with RFC 3261.
- **disabled**—A Route header is not included in the outgoing request; in accordance with the route processing rules described in RFC 2543 (referred to as strict routing). That rule caused proxies to destroy the contents of the Request-URI when a Route header field was present.

Whether loose routing field is enabled is also checked when a local policy ‘s next hop value matches a session agent. Matching occurs if the hostname or the session agent’s IP address field value corresponds to the next hop value. If loose routing is enabled for the matching session agent, the outgoing request retains the original Request-URI and Route header with the next hop address.

About the Ingress Realm

You can create a list of realms in your local policy that is used by the Oracle Communications Session Border Controller to determine how to route traffic. This list determines from which realm traffic is coming and is used for routing by ingress realm.

The source realm values must correspond to valid identifier entered when the realm was configured.

About the Egress Realm

An egress realm allows SIP signaling to travel out of the Oracle Communications Session Border Controller through a network other than the home realm. The Oracle Communications Session Border Controller uses egress realms for signaling purposes (when matching flows). When a packet arrives at the Oracle Communications Session Border Controller with a destination address that does not match any defined session agents, the Oracle Communications Session Border Controller uses the address associated with the realm that is, in turn, associated with the SIP configuration’s egress realm ID, as the outgoing network. With the use of the egress realm ID, it is possible to define a default route for SIP requests addressed to destinations outside the home realm. If no egress realm is defined, the home realm (default ingress realm) is used as the default egress realm.

With session agent egress realm configured, the Oracle Communications Session Border Controller adds a default egress realm to the session agent to identify the signaling interface used for ping requests. The Oracle Communications Session Border Controller also uses the default egress realm when the normal routing request does not yield an egress realm—for example, when a local policy does not specify the next hop’s realm.

When you configure session agents, you can define them without realms or you can wildcard the realm value. These are global session agents, and multiple signaling interfaces can reach them. Then, when you use session agent pinging, the Oracle Communications Session Border Controller sends out ping requests using the signaling interface of the default egress realm defined in the global SIP configuration. The global session agents in certain environments can cause problems when multiple global session agents residing in multiple networks, some of which might not be reachable using the default SIP interface egress realm.

The Oracle Communications Session Border Controller uses the session agent egress realm for ping messages even when the session agent has a realm defined. For normal request routing, the Oracle Communications Session Border Controller uses the egress realm for global session agents when local policies or SIP-NAT bridge configurations do not point to an egress realm.

Ping Message Egress Realm Precedence

For ping messages, the egress realm precedence occurs in the following way (in order of precedence):

- Egress realm identified for the session agent.
- Session agent realm (set in the realm-id parameter) or the wildcarded value
- Global SIP configuration egress realm, when configured in the egress-realm parameter
- Global SIP configuration home realm

Normal Request Egress Realm Precedence

For normal request routing, the egress realm precedence occurs in the following way (in order of precedence):

- Egress SIP-NAT realm, when the **route-home-proxy** parameter is set to **forced** and no local policy match is found
- Matching local policy realm, when configured in the local policy attributes
- Session agent realm (set in the realm-id parameter) or the wildcarded value
- Session agent egress realm, when configured in the egress-realm-id parameter
- Global SIP configuration egress realm, when configured in the egress-realm parameter
- Global SIP configuration home realm

Session Agent Egress Realm Configuration

Configuring a session agent egress realm is optional.

To configure a session agent egress realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are adding this feature to an existing configuration, you need to select the configuration (using the ACLI **select** command) before making your changes.

4. **egress-realm-id**—Enter the name of the realm you want defined as the default egress realm used for ping messages. The Oracle Communications Session Border Controller will also use this realm when it cannot determine the egress realm from normal routing. There is no default value for this parameter.

5. Save and activate your configuration.

About SIP Redirect

SIP redirect involves proxy redirect and tunnel redirect.

Proxy Redirect

You can configure the SIP proxy mode to define how the SIP proxy will forward requests coming from the session agent. This value is used if the session agent's proxy mode has no value (is empty).

Tunnel Redirect

You can use tunnel redirect when requests are routed to a server behind a SIP NAT that sends redirect responses with addresses that should not be modified by the SIP NAT function. For example, a provider might wish to redirect certain calls (like 911) to a gateway that is local to a the UA that sent the request. Since the gateway address is local to the realm of the UA, it should not be modified by the SIP NAT of the server's realm. Note that the server must have a session agent configured with the redirect-action field set to the proxy option in order to cause the redirect response to be sent back to the UA.

SIP Method Matching and To Header Use for Local Policies

For SIP, this feature grants you greater flexibility when using local policies and has two aspects: basing local policy routing decisions on one or more SIP methods you configure and enabling the Oracle Communications Session Border Controller to use the TO header in REGISTER messages for routing REGISTER requests.

SIP Methods for Local Policies

This feature allows the Oracle Communications Session Border Controller to include SIP methods in routing decisions. If you want to use this feature, you set a list of one or more SIP methods in the local policy attributes. These are the SIP methods you can enter in the list: INVITE, REGISTER, PRACK, OPTIONS, INFO, SUBSCRIBE, NOTIFY, REFER, UPDATE, MESSAGE, and PUBLISH.

After the Oracle Communications Session Border Controller performs a local policy look-up for SIP, it then searches for local policy attributes that have this methods list configured. If it finds a a set of policy attributes that matches a method that matches the traffic it is routing, the Oracle Communications Session Border Controller uses that set of policy attributes. This means that the Oracle Communications Session Border Controller considers first any policy attributes with methods configured before it considers those that do not have methods. In the absence of any policy attributes with methods, the Oracle Communications Session Border Controller uses the remaining ones for matching.

In cases where it finds neither matching policy attributes with methods or matching policy attributes without them, the Oracle Communications Session Border Controller either rejects the calls with a 404 No Routes Found (if the request calls for a response) or drops the call.

You configure local policy matching with SIP methods in the local policy attributes parameter calls **methods**. This parameter is a list that takes either one or multiple values. If you want to enter multiple values, you put them in the same command line entry, enclosed in quotation marks and separated by spaces.

To configure SIP methods for local policy matching:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

```
ORACLE(session-router)# local-policy
ORACLE(local-policy)#
```

4. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

```
ORACLE(local-policy))# policy-attributes
ORACLE(policy-attributes)#
```

5. **methods**—Enter the SIP methods you want to use for matching this set of policy attributes. Your list can include: INVITE, REGISTER, PRACK, OPTIONS, INFO, SUBSCRIBE, NOTIFY, REFER, UPDATE, MESSAGE, and PUBLISH.

   By default, this parameter is empty—meaning that SIP methods will not be taken into consideration for routing based on this set of policy attributes.

   If you want to enter more than one method, you entry will resemble the following example.

```
ACMEPACKET(local-policy-attributes)# methods "PRACK INFO REFER"
```

6. Save and activate your configuration.

Routing Using the TO Header

For the Oracle Communications Session Border Controller's global SIP configuration, you can enable the use of an ENUM query to return the SIP URI of the Registrar for a SIP REGISTER message. Without this feature enabled, the Oracle Communications Session Border Controller uses the REQUEST URI. This ability can be helpful because REGISTER messages only have the domain in the REQUEST URI, whereas the SIP URI in the To header contains the user's identity.

There are two parts to enabling this feature. First, you must enable the **register-use-to-for-lp** parameter in the global SIP configuration. Then you can set the next-hop in the applicable local policy attributes set to **ENUM**.

To enable your global SIP configuration for routing using the TO header:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter. If you are adding this feature to a pre-existing SIP configuration, you will need to select and edit it.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **register-use-to-for-lp**—Set this parameter to enabled if you want the Oracle Communications Session Border Controller to use, for routing purposes, an ENUM query to return the SIP URI of the Registrar for a SIP REGISTER message. This parameters defaults to **disabled**.

   To set up your local policy attributes for routing using the TO header:

5. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

6. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

7. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

```
ORACLE(session-router)# local-policy
ORACLE(local-policy)#
```

8. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

```
ORACLE(local-policy))# policy-attributes
ORACLE(policy-attributes)#
```

9. **next-hop**—This is the next signaling host. Set this parameter to ENUM if you want to use SIP methods in local policy attribute information for routing purposes.
10. Save and activate your configuration.

H.323 Routing

This section describes H.323 routing.

Egress Stack Selection

Egress stack selection includes static stack selection and policy-based stack selection

Static Stack Selection

In static stack selection, the outgoing stack is determined though the establishment of associated stacks in the h323 stack.

The incoming stack (configured in the h323 stack) uses its associated stack value to determine the associated outgoing stack. The associated stack value corresponds to the name of an h323 stack. This type of selection is referred to as static because the incoming stack always uses the stack specified in the associated stack as the outgoing stack; no other stacks are considered.

Policy-Based Stack Selection

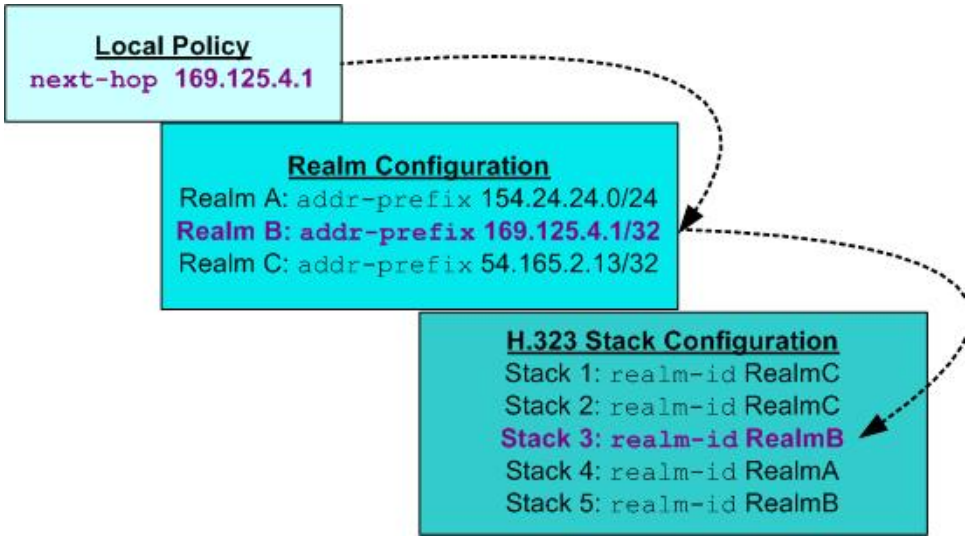
The Oracle Communications Session Border Controller performs dynamic, policy-based stack selection when an H.323 call arrives at the Oracle Communications Session Border Controller and a configured associated outgoing stack cannot be found.

For policy-based stack selection, the Oracle Communications Session Border Controller refers to local policies that contain address information that corresponds to incoming traffic. This information is contained in the local policy's To address and From address fields. For the source, this information is matched with the Q.931 calling party number; if there is no calling party number, the H.323 source address is used. For the destination, this information is matched with the called party number; if there is no called party number, then the H.323 destination address is used.

After a local policy corresponding to the incoming traffic has been found, the Oracle Communications Session Border Controller looks at the next hop value (a local policy attribute) and selects a local policy for the basis of stack selection. If the local policy look-up yields multiple local policies with the same next hop values, but with different cost values, the local policy with the lowest cost value is selected.

If a realm is not defined in the local policy, the next hop address is then matched against the address prefix values for the realms that are configured for the system. Thus, the Oracle Communications Session Border Controller discovers the realm for this traffic. Using this realm information, the Oracle Communications Session Border Controller performs stack selection. It uses the first configured H.323 stack in the Oracle Communications Session Border Controller's configuration that has a realm ID value matching the identifier field of the realm with the appropriate address prefix.

In the following example, the local policy matching yields a local policy with a next hop value of 169.125.4.1, which corresponds to RealmB. The outgoing stack selected is Stack 3 because it is the first stack to have been configured with RealmB as the realm ID.



Policy-Based Stack Selection

Registration Caching

The Oracle Communications Session Border Controller can cache and proxy an H.225 RegistrationRequest (RRQ) between an H.323 endpoint and a gatekeeper. Registration caching serves two functions:

- It allows aggregation of RRQs sent to a gatekeeper stack and proxies those requests through the gateway stack. If the external gatekeeper associated with the gatekeeper stack supports additive registration, the requests will be consolidated. Furthermore, if the gatekeeper supports additive registration, the Oracle Communications Session Border Controller will register in an additive manner, meaning that will send additive RRQs.
- It allows the gatekeeper stack to use the registration information to route calls from other realms to endpoints in its realms.

To perform registration caching, the Oracle Communications Session Border Controller must be configured with at least two stacks. One of these stacks will receive registrations (gatekeeper stack), and one stack will proxy registrations (gateway stack). The Oracle Communications Session Border Controller caches all successful registrations and uses the cache to route calls to the endpoints.

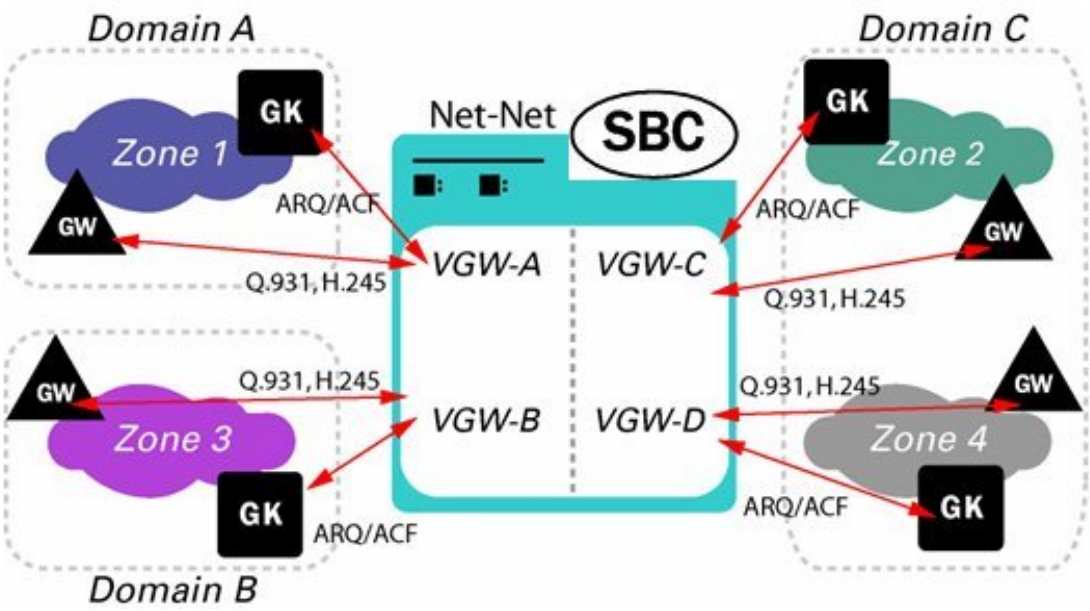
Gatekeeper Provided Routes

Gatekeeper provided routes includes back-to-back gateways, back-to-back gatekeeper and gateway, and interworking gatekeeper/gateway.

Back-to-Back Gateway

When the SBC is functioning as a back-to-back gateway (B2BGW), it appears as multiple H.323 gateways to multiple networks. Each SBC virtual gateway discovers and registers with a gatekeeper in its respective domain. Each gateway relies on its gatekeeper for admission and location services through the ARQ/ACF exchange. H.225 call control and H.245 messages are exchanged directly with the terminating gateway or gatekeeper. Routing policies are used to associate one virtual gateway with another.

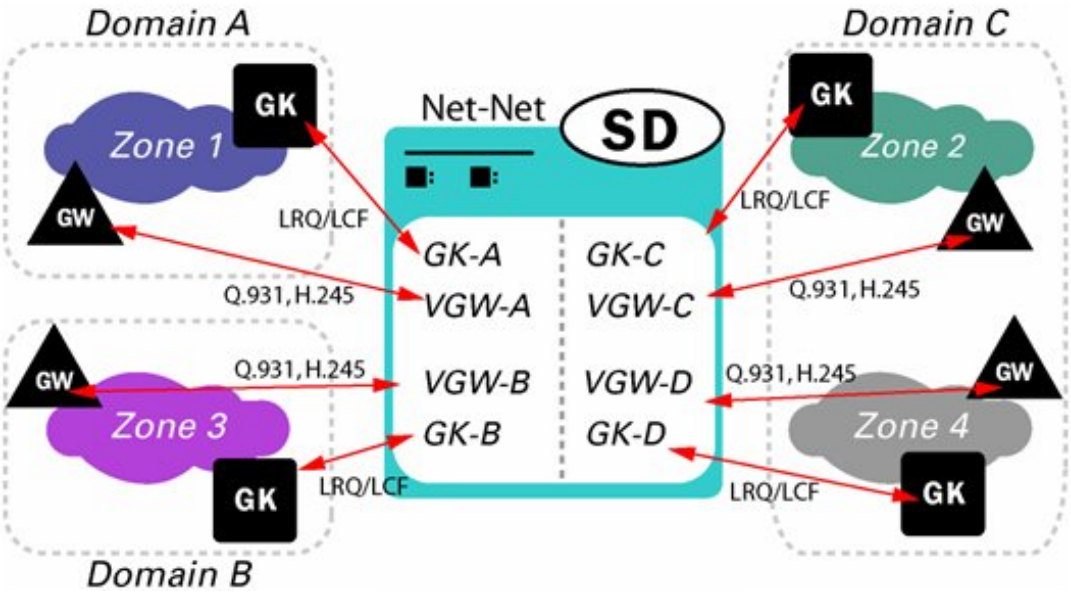
The following diagram illustrates the back-to-back gateway.



Back-to-Back Gatekeeper and Gateway

For peering connections where both networks use inter-domain gatekeeper signaling, the SBC is configured as a back-to-back gatekeeper proxy and gateway mode of operation. The SBC responds and issues LRQs and LCFs/LRJs acting as a routed gatekeeper. Peered gatekeepers send LRQ to the RAS address of one of the SBC's virtual gatekeepers and it responds by providing its call signaling address that performs the gateway functions. Routing policies are used to determine the egress virtual gatekeeper that then exchanges LRG/LCF to determine the call signaling address of the terminating gateway.

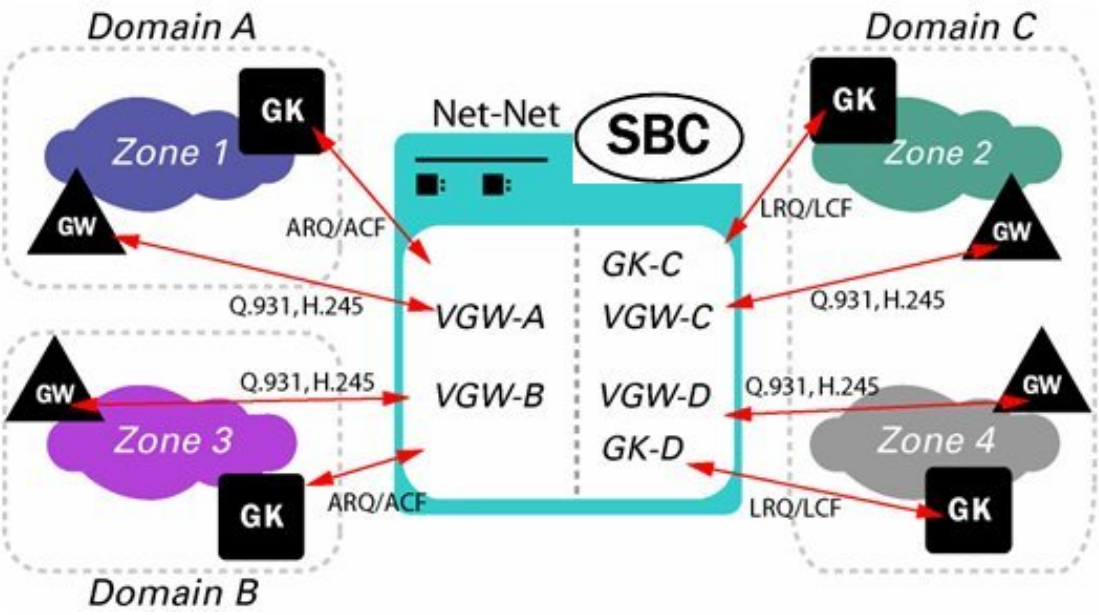
The following diagram illustrates the back-to-back gatekeeper and gateway.



Interworking Gatekeeper Gateway

In the interworking gatekeeper/gateway signaling mode of operation, the SBC interworks between the other two modes; presenting a routed gatekeeper interface to one zone and a gateway to the other.

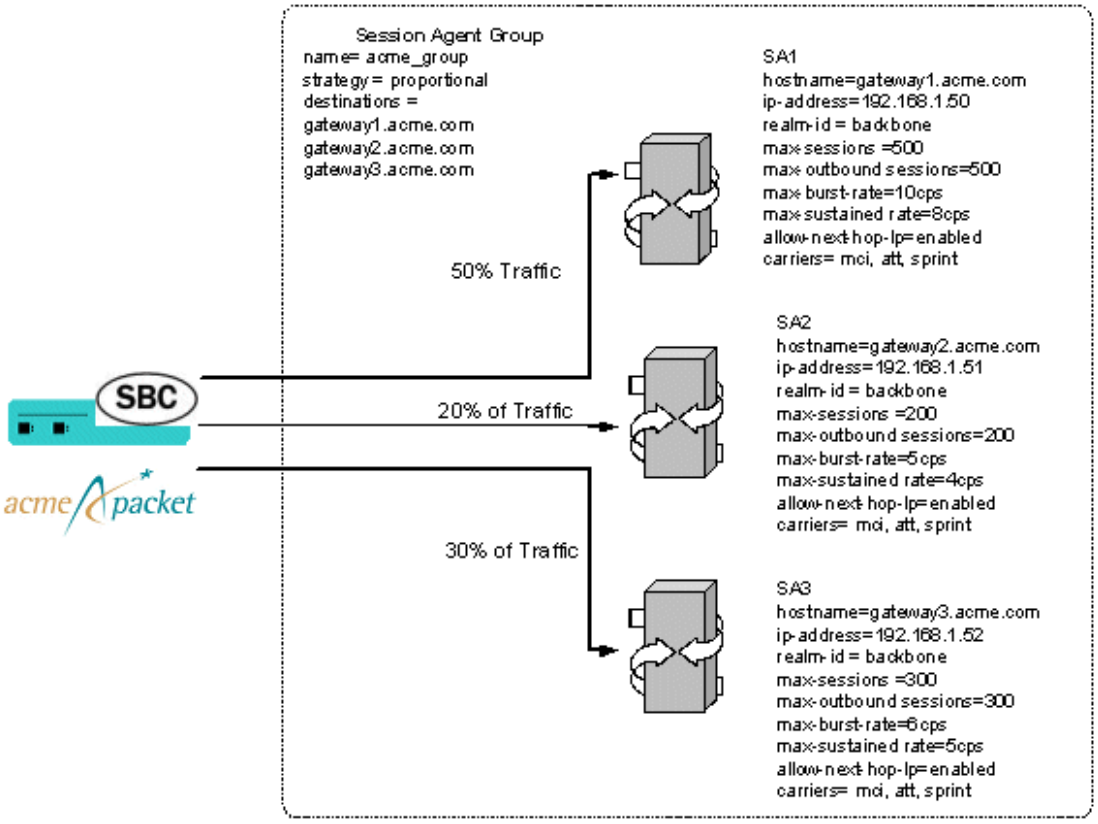
The following diagram illustrates the interworking gatekeeper/gateway.



Load Balancing

This section describes Oracle Communications Session Border Controller load balancing. You can use session agent groups to assist in load balancing among session agents. You define concurrent session capacity and rate attributes for each session agent and then define the session agent group. Next, you select the allocation strategy you want applied to achieve the load balancing you want.

The following example shows a configuration for load balancing gateways based on a proportional allocation strategy.



Routing and load balancing capabilities include the following:

- least cost, which includes cost-based and time-based routing
- customer preference
- traffic aggregation
- routing by media (codec) type
- capacity-based, by destination
- service element load balancing
- service element failure detection and re-route
- session agent failure
- routing by codec

Parallel Call Forking

You can configure the SBC to direct calls to targets simultaneously using parallel forking. You establish parallel forking behavior by enabling the parallel-forking parameter on one or more **local-policy** elements and configuring the **cost** within each applicable **policy-attribute**.

This feature relies on the system's route selection process to establish a list of routes from which it determines targets for parallel forking. The selection process includes referring to one or more **local-policy** elements triggered by the initial INVITE for best match and other standard criteria within the applicable **policy-attribute** sub-elements. Having determined route order, the system then identifies those **local-policy** elements that have **parallel-forking** enabled.

Having determined which **local-policy** elements apply, the system then refers to the **cost** of each **policy-attribute** within each applicable **local-policy**. The SBC then groups attributes that have the same **cost** together for forwarding in parallel. The final route list includes batches of routes for parallel forwarding, which may or may not be intermingled with routes the system uses serially.

Having issued a set of INVITEs in parallel, the SBC waits for all parallel endpoints to error or timeout before proceeding to the next routes, which may or may not be another group of parallel routes. When any endpoint accepts the call, the system issues CANCELs to the any other endpoints forked in parallel and stops trying to reach any further parallel or serial endpoints. These CANCEL messages use the **sip-locally-generated-cancel-for-parallel-fork** entry in the **local-reponse-map** to signal the cancel. Unless you configure it otherwise, the system sends the default reason.

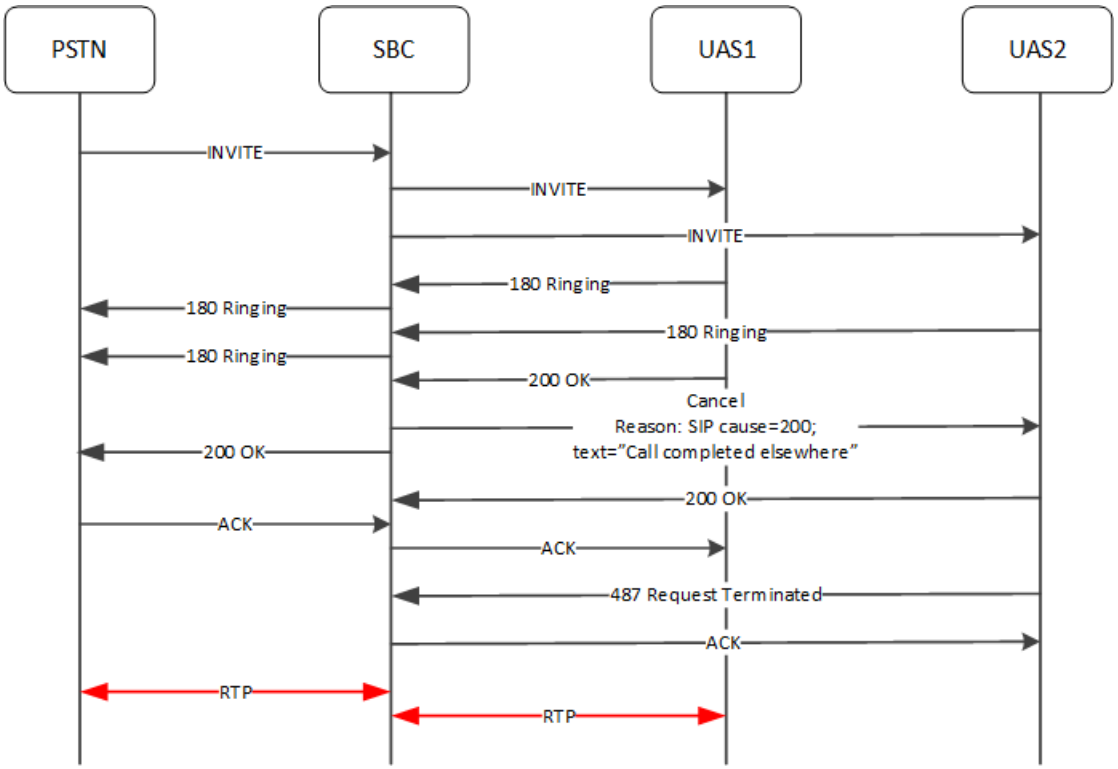
```
Reason: SIP; cause=0; text="Call completed elsewhere"
```

> **Note:**
>
> This feature supports HMR and Session Translation tools you use to change To and FROM numbers and formats, such as E.164 and National, in INVITEs.

You configure parallel call forking based on the state of the **parallel-forking** and **cost** parameters. The system can perform parallel-forking with this **local-policy** if:

- You have enabled **parallel-forking**
- You have set the same **cost** in the **policy-attribute** of the targets

The diagram below depicts the SBC supporting a simple parallel call forking scenario. The CANCEL includes your configured reason-header.



The SBC may also perform serial forking within the context of parallel forking call flows when it encounters:

- A **local-policy** in its sequence that targets a single station
- Multiple DNS resolutions to an FQDN
- A 302 responses to the INVITE indicating the target has moved

The system attempts to reach serial targets when it reaches them in the route list or when serial targets are generated by a current route target. If any target responds with a 302 response, the system attempts to reach those targets serially. If it receives a 302 during an active parallel forking batch, the system attempts to reach the 302 targets serially when all parallel targets fail. If there is no response or 503 errors from a serial target the system proceeds with the next targets in the route list, which may or may not include parallel forking targets.

If it receives no responses or 503 error responses from all targets, meaning no stations have accepted the call, the system ends the call after all timeouts expire.

The parallel forking feature works in conjunction with the **merge-early-dialogs** feature. It is mandatory that you enable **merge-early-dialogs** on the ingress **realm-config** for parallel forking. Without it, media does not flow correctly. Refer to the Merge Function within Early Dialog Support topic in this guide for operational information, including limitations.

This implementation also supports parallel forking for early media flows.

The SBC can perform parallel call forking to **local-policy** destinations configured as:

- IP address
- MSISDN
- ENUM
- LRT—When forking to routes in an LRT, the system ignores the LRT entries' weight and cost. Instead, the system parallel forks to all destinations.
- **session-agent**—The SBC can fork to a session agent whether it targets an IP address or an FQDN, performing a DNS dip for FQDNs. If the FQDN, however, produces multiple resolutions, the system attempts to reach each resolution serially.
- **session-agent-group** (SAG)—The SBC performs parallel forking to all agents in a session agent group if you have enabled **sag-recursion** on that group. If **sag-recursion** is disabled, the SBC can perform parallel forking with equal cost endpoints and the first agent in the group only.
  A SAG may include session agents configured as FQDNs. If a call flow triggers this SAG, the system obtains resolutions to the FQDN, but forwards to those resolutions serially.

  For example, consider the system forwarding in parallel to an endpoint named UAS1 and a SAG that includes two session agents configured as FQDNs. Next, assume these DNS lookups result in 2 IPs (IP1, IP2) for FQDN1 and 2 IPs (IP3, IP4) for FQDN2. In this case, the SBC:

  1. Forks the INVITE to UAS1 and IP1 in parallel.
  2. If both UAS1 and IP1 timeout, the SBC forwards the INVITE to IP2.
  3. If IP2 times out, the SBC forwards the INVITE to IP3.
  4. If IP3 times out, the SBC forwards the INVITE to IP4.
  5. If IP4 times out, the SBC terminates the call with the UAC.

  If any endpoint accepts the call, the SBC sends CANCELs to outstanding INVITEs and stops attempting to reach any further endpoints.

- FQDN—The SBC performs parallel forking with only the first target in any DNS response. Parallel forking scenarios wherein the SBC targets an FQDN include a targeted **local-policy** as a **session-agent** with a **hostname** that is an FQDN.
  The system uses serial forking to cycle through all of a DNS lookup's resolutions before resuming with the original route list.

Configuration

You enable **parallel-forking** on a **local-policy** by enabling the feature and disabling **terminate-recursion**.

```
ORACLE(local-policy)# parallel-forking enabled
ORACLE(local-policy)# terminate-recursion disabled
```

If desired, you can configure a custom response that the system sends to the endpoints it does not select for a these calls. This configuration uses the **sip-locally-generated-cancel-for-parallel-fork** error within the **local-response-map**, which allows you to configure a custom **sip-status** and **sip-reason**. This does not affect functionality and is typically used to specify custom reason text.

```
ORACLE(local-response-map)#
entries
      local-error            sip-locally-generated-cancel-for-parallel-
fork
      sip-status             200
      sip-reason             Call Completed at other place
```

You must also manage the number of steering-pools ports available within parallel forking scenarios. The SBC reserves ports for all forked destinations and does not release them until it receives a 200 OK for the call. This process can temporarily consume a large number of ports.

Related Configuration

This **parallel-forking** feature interacts with other SBC configuration that can change or affect its behavior:

- **terminate-recursion**—Disable the **terminate-recursion** parameter in **local-policy** under **policy-attribute** if you want to use forking, either parallel or serial. If **terminate-recursion** is enabled, the SBC does not try to reach the next route.
- **merge-early-dialogs**—Enable the **merge-early-dialogs** parameter in conjunction with **parallel-forking** to merge the early dialogs so that the applicable UACs only need to maintain a single dialog. When using **merge-early-dialogs** with **parallel-forking** causes the SBC to send only a single final response to the applicable UAC. If one forking destination returns a 200 OK and the others return CANCEL, 487, ACK and so forth, the system only sends the 200 OK to that UAC.
- **sag-recursion**—Enable the **sag-recursion** parameter on any SAG you use as a parallel forking target, if you want to parallel fork INVITE to all **session-agents** in that SAG.
- **prevent-duplicate-attrs**—Enable this parameter in the **account-config** to properly update ACME FlowIDs and FlowType AVPs in ACRs for CDR generation.
- **policy-priority**—If the route sequence includes routes of equal cost from multiple **local-policy** elements, and all of those local policies have the same **policy-priority**, the system forwards to all routes in parallel. If, however, a **local-policy** has a different **policy-priority**, the system does not include those routes in the parallel forking batch.

A **policy-priority** setting does not affect route order, but when it reaches those routes in the list order, the system handles the routes in that **local-policy** as its own batch.

Forking Operation

When you enable parallel forking on a **local-policy**, the SBC can use it to establish a list of routes to which it can simultaneously forward an INVITE. There can be multiple sets of these parallel routes as well as routes for serial forking ordered by cost in these route lists. In addition, the system can change the contents and order of the list based on external replies, such as DNS responses or 3xx replies. When attempting to reach an endpoint, the system uses timeouts and errors to trigger attempts to reach the next route(s). During these timeouts, the system waits to receive a 200 OK from one of the endpoints. The SBC ends this search when it receives the 200 OK and cancels any incomplete session setups.

The SBC refers to your cost and priority configuration to establish "batches" of routes to use for parallel forking. Batches are typically endpoints that have the same cost and priority. The SBC collects these together to create a routing sequence. When triggered by an INVITE, the SBC uses parallel forking to signal all endpoints within a batch simultaneously. If there is only one endpoint in a batch, the SBC signals that endpoint only.

For example, consider a configuration that results in the following "batches".

UAC-1, Cost-1	Batch-1
UAC-2, Cost-1	Batch-1
UAC-3, Cost-2	Batch-2
UAC-4, Cost-3	Batch-3
UAC-5, Cost-3	Batch-3

This configuration results in the SBC attempting to reach:

1. The endpoints in Batch-1 simultaneously
2. UAC3 alone (Batch-2)
3. The endpoints in Batch-3 simultaneously

This next example demonstrates how responses can affect this routing. Consider a configuration that results in the following two batches:

- Route1 – cost 5 – parallel-forking enabled
- Route2 – cost 5 – parallel-forking enabled
- Route3 – cost 10 – parallel-forking enabled
- Route4 – cost 10 – parallel-forking enabled

Based on configuration, the SBC route procedure includes:

1. Parallel forking to Route1 and Route2 (two INVITEs sent out).
2. Parallel forking to Route3 and Route4 (two INVITEs sent out).

If Route2, however, returns a 302 response that includes 2 contacts, the SBC route procedure changes to include serial forking to the two contact provided by Route2:

1. Parallel forking to Route1 and Route2 (two INVITEs sent out - Route1 fails - Route2 sends 302)
2. Forward to the first Route2 contacts.

3. If the first contact fails, such as 18x timeout/error response, try the next 302 contact.
4. If the second contact fails, parallel fork to Route3 and Route4.

In this next example, consider the configuration of the applicable routes wherein Route 2 has parallel forking disabled:

- Route1 – cost 5 – parallel-forking enabled
- Route2 – cost 5 – parallel-forking disabled
- Route3 – cost 10 – parallel-forking enabled
- Route4 – cost 10 – parallel-forking enabled

This configuration results in three batches with parallel forking used only for Route3 and Route4, if the process reaches them.

Behaviors in Response to 302 Messages

Another simple scenario has the SBC responding to a 302 response with multiple contacts from a route.

- Route1 – cost 5 – parallel-forking enabled
- Route2 – cost 5 – parallel-forking enabled

1. The SBC performs parallel-forking with Route1 and Route2.
2. If Route1 destination returns a 302 with 3 Contacts, IP1, IP2 and IP3, the SBC performs serial forking to these IPs.

Consider the scenario wherein the SBC forks the INVITE to Route1 and IP1, and both Route1 and IP1 reply with a 302 with the following contacts:

- Route1 302 has uas1 and uas2 as Contact IPs.
- IP1 302 has uas3 and uas4 as Contact IPs.

In this case, the SBC uses the following procedure.

1. Upon receiving the 302 from Route1, the system forks the INVITE to uas1.
2. Upon receiving the 302 from IP1, the system forks the INVITE to uas3.
3. After the uas1 timeout, the system forks the INVITE to uas4, because, upon receiving the first 302, the system's redirect list includes uas1 and uas2.
4. Upon receiving the second 302, the system's redirect list includes uas1, uas3, uas4 and uas2.
   The SBC sends INVITEs using the same sequence. Also in case either uas1 or uas3 or both of them respond with some provisional response, then next redirect from the list will be tried only after the timeout of all pending transactions, that is when timeouts or error-responses come from uas1 and uas2.

Behavior Using DNS Resolutions

There are cases wherein the SBC receives an INVITE that uses an FQDN as a target, requiring a DNS dip before it can determine its routing. Generally speaking, the SBC only uses the first IP address in the DNS response as a target for a parallel forking scenario. Similar to 3xx redirect scenarios, however, the SBC performs serial forking on the remaining resolutions if the first resolution results in no response or a 503 response.

Consider the following configuration, and assume the DNS response to the FQDN includes 3 targets, including IP1, IP2, IP3:

- Route1 – cost 5 – parallel-forking enabled
- Route2 (FQDN) – cost 5 – parallel-forking enabled
- Route3 – cost 5 – parallel-forking enabled

A simple scenario has the SBC sending INVITEs and taking the following steps in response to 503 or no response:

1. Performs parallel-forking to Route1 and IP1, the first resolution from DNS
2. IP2 as a serial fork
3. IP3 as a serial fork
4. Route3

> **Note:**
>
> If Route 2 gets multiple DNS-resolutions that the SBC reaches through a session-agent, you must have enabled **ping-all-addresses** for the system to use serial forking on the DNS-resolved targets.

When not configured for **parallel-forking**, the SBC will use serial forking for the configuration that includes the FQDN above when it sends an INVITE to IP1 in response to the following scenarios:

- If the INVITE times out or IP1 responds with 503 response, the SBC sends the INVITE to IP2. If the IP2 INVITE times out, the SBC sends the INVITE to IP3.
- If IP1 sends a 1xx response and then times out, the SBC does not try to reach other DNS-resolved targets. Instead, the SBC routes using the next **policy-attribute**.
- If IP1 sends 302 with 2 contacts, the SBC sends INVITES to those 2 contacts serially. If these contacts timeout, the SBC ends the call.
- If IP1 sends a 4xx or 5xx error response, with the exception of 503, the SBC does not try to reach other DNS-resolved targets. Instead, the SBC routes using the next **policy-attribute**.
- If IP1 sends a 6xx error response, the SBC forwards the 6xx message to the UAC and then ends the call.

When configured for **parallel-forking**, the SBC may use parallel or serial forking for the configuration that includes FQDN above:

- If your configuration causes the SBC to fork INVITEs to Route1 and IP1, and Route1 sends 180 Ringing then times out, and IP1 sends no response:
  1. The system tries to reach IP2 after 32 seconds.
  2. If the IP2 contact fails, the system tries to reach IP3.

Scenario Including Both FQDNs and 302 Responses

Consider the following configuration, and assume the DNS response to the FQDN includes 3 targets, including IP1, IP2, IP3:

- Route1 – cost 5 – parallel-forking enabled
- Route2 (FQDN) – cost 5 – parallel-forking enabled

In this case, the SBC:

1. Forks in parallel to Route1 and IP1 in parallel. Assume Route1 responds with 302 with 2 Contacts (UAS1, UAS2).
2. If there is no response/error response from IP1, the system tries UAS1 serially.
3. If there is no response/503 response from UAS1, the system tries UAS2 serially.
4. If there is no response/503 response from UAS2, the system tries IP2 serially.
5. If there is no response/503 response from IP2 the system ends the call.

Parallel Forking Call Flows

The SBC supports parallel forking within multiple contexts. These context include adjusting target lists based on FQDN responses, 302 responses as well as LDAP and LRT dips. These contexts provide a framework within which the SBC invokes parallel forking when triggered. The flow diagrams presented here, therefore, have similar structures with the triggers and results of parallel forking residing within the flows at similar points within each flow, and with similar denotation.

For example, consider the simple call flow presented at the beginning. The basic configuration that produces this flow is below. This same call flow would apply for multiple equivalent functions by changing the **next-hop** value.

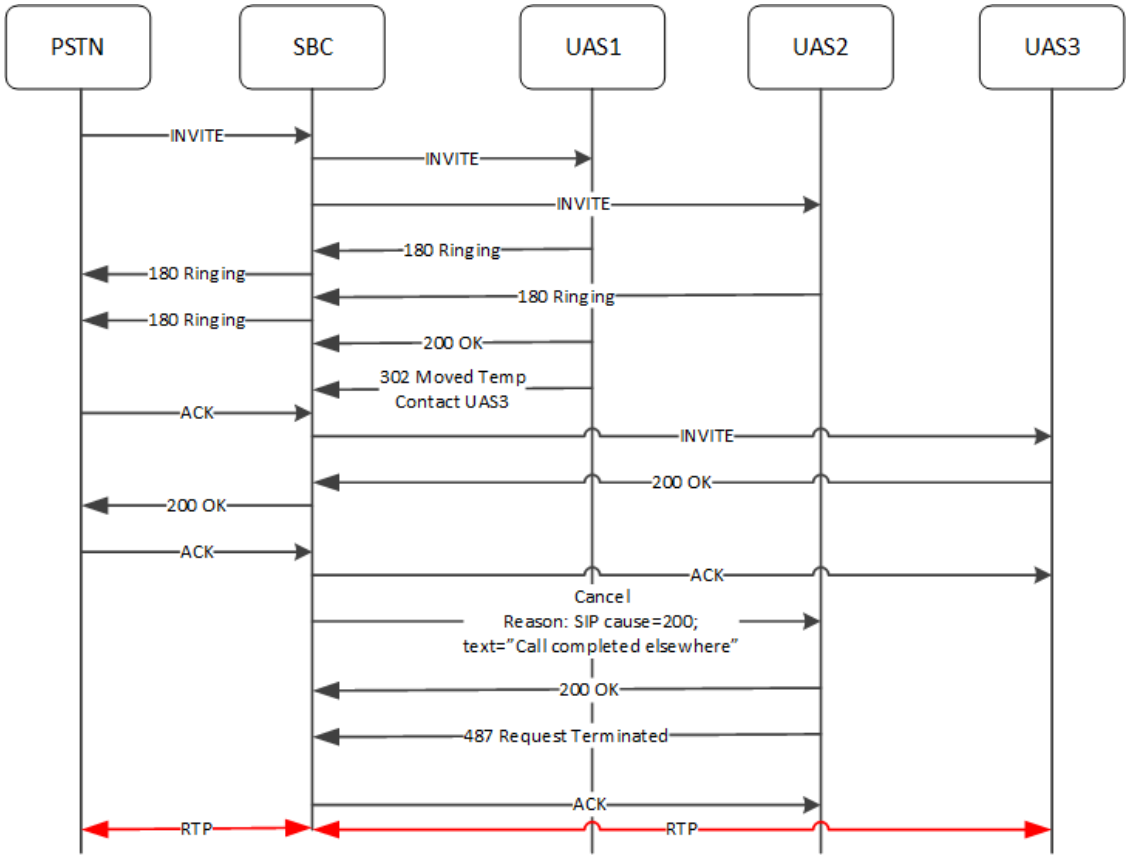
```
local-policy
from-address                *
to-address                  2222
source-realm                sip192
parallel-forking            enabled
policy-attribute
next-hop                    UAS1
realm                       sip172
cost                        10
policy-attribute
next-hop                    UAS2
realm                       core2
cost                        10
```

The same call flow would result when using:

- A DNS dip, such as **sip.pstnhub.microsoft.com**
- An LDAP dip, such as **ldap:default-query**
- An LRT dip, such as **lrt:test1**
- A multi-stage lookup, such as:
  - **next-hop LRT:Test1;key=$TO**
  - **lookup multi**

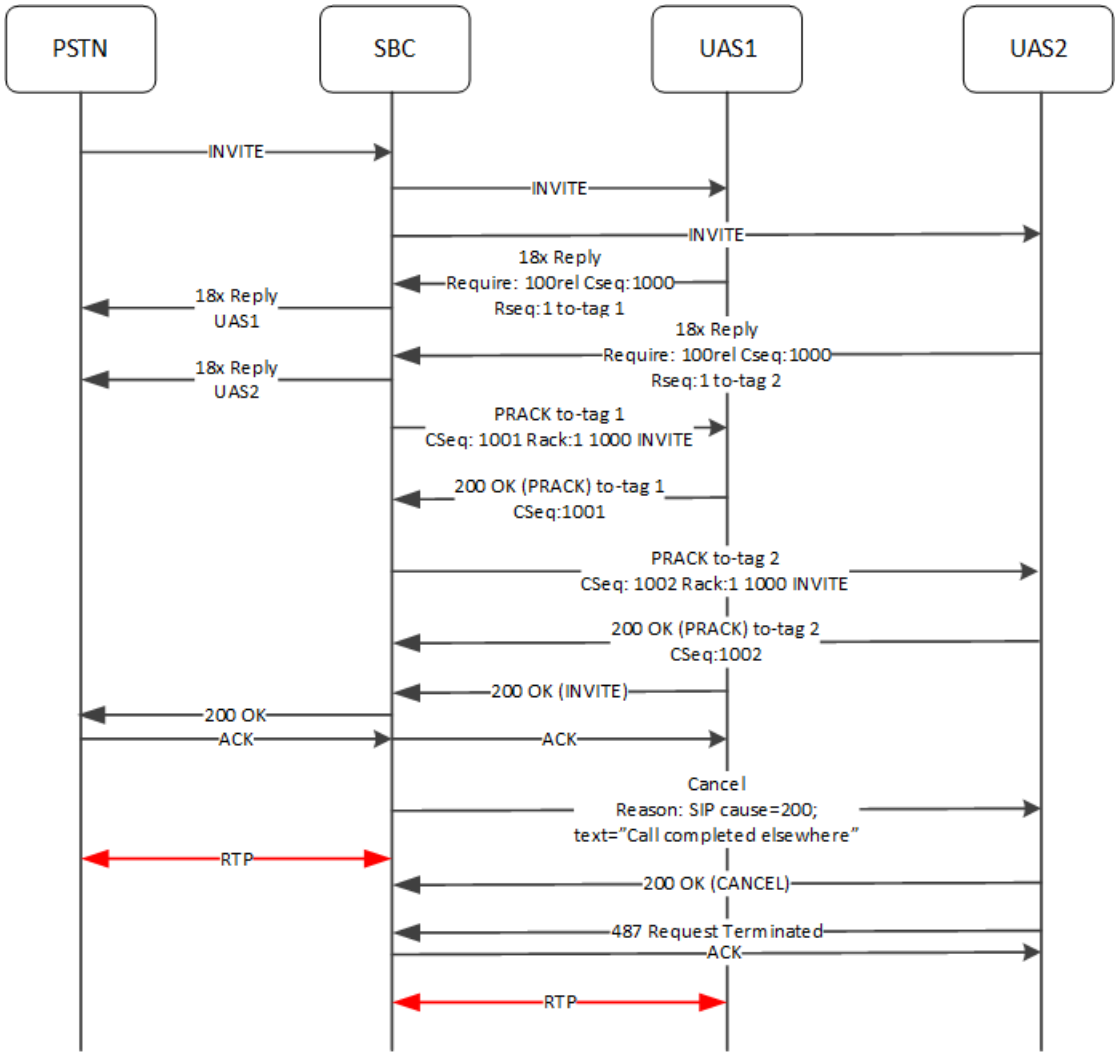
Parallel Forking and 302 Response

Consider the scenario wherein the SBC engages parallel-forking with one of the target stations replying with a 302: Moved. In this case the SBC responds to the 302 by using the new station within the parallel forking step. The system removes UAS2 from consideration and add UAS3 as a replacement.



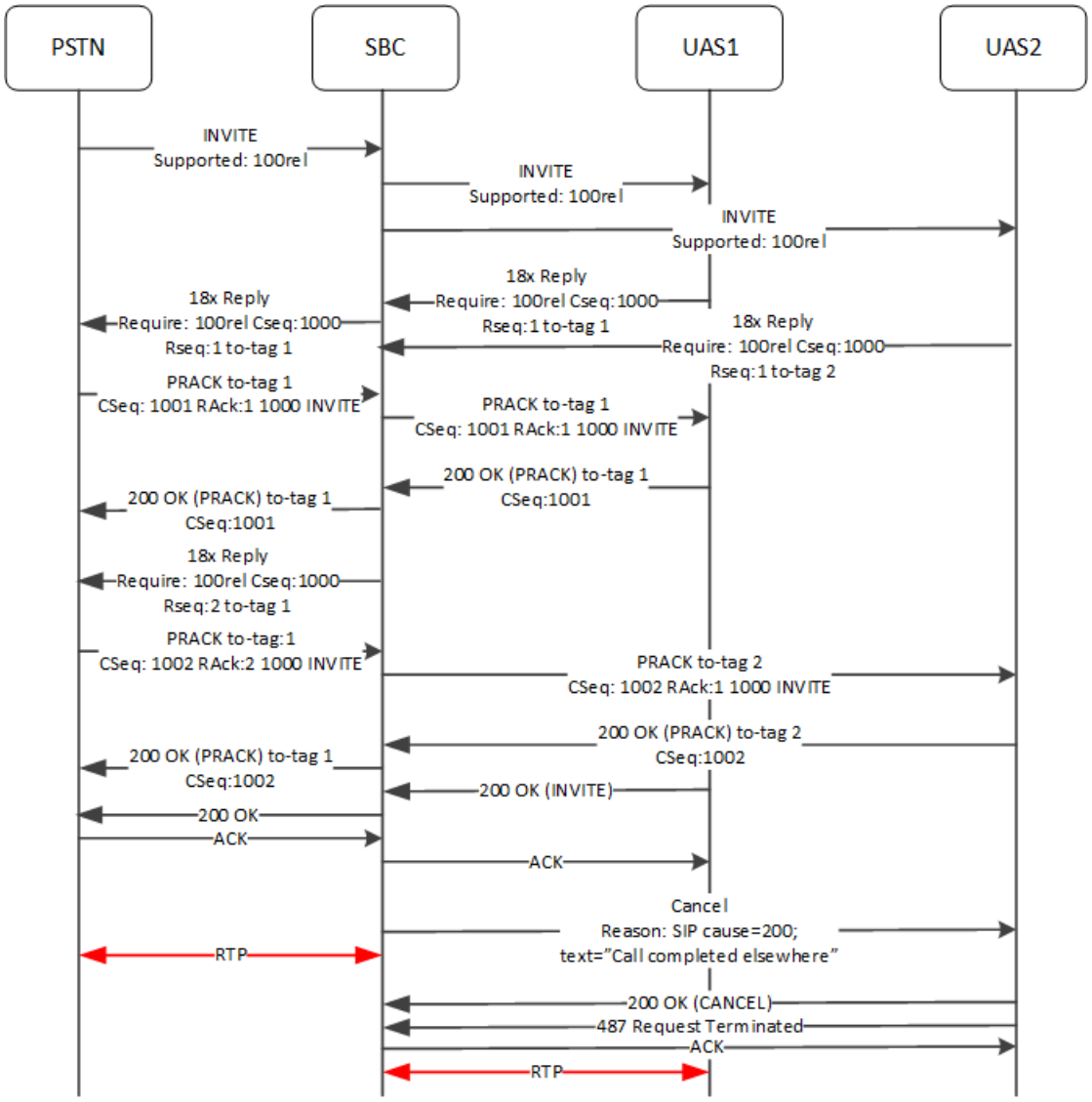
Parallel Forking and PRACK

In this scenario, the SBC and both UAS end points support PRACK, but the PSTN does not. The SBC supports the requirements the end points send for the PSTN and supports the PRACK transactions.



Parallel Forking and PRACK2

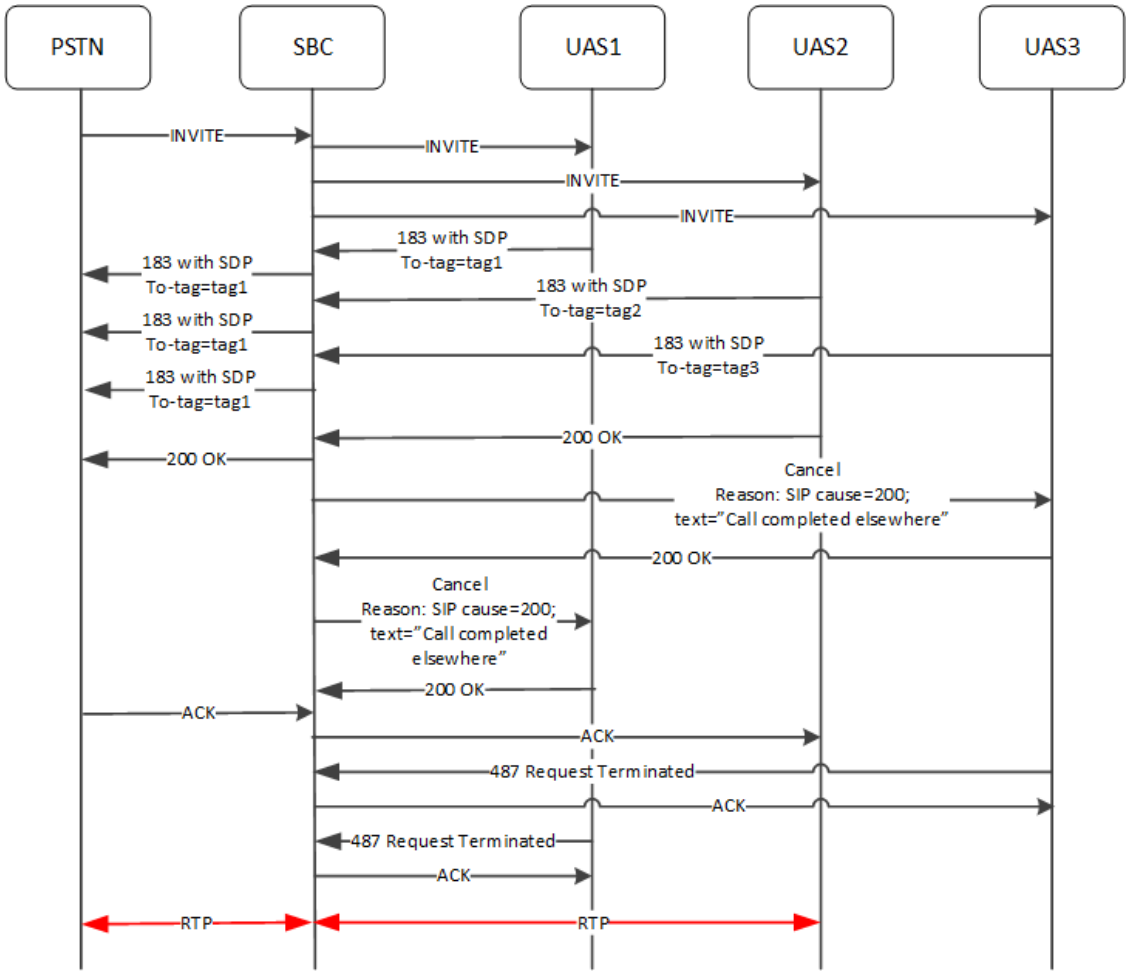
In this scenario, the SBC and the PSTN support PRACK, but both UAS end points do not. The PSTN generates the 100-rel requirement and the SBC supports the requirement and the PRACK transactions on behalf of the end points.



Parallel Forking and Early Media

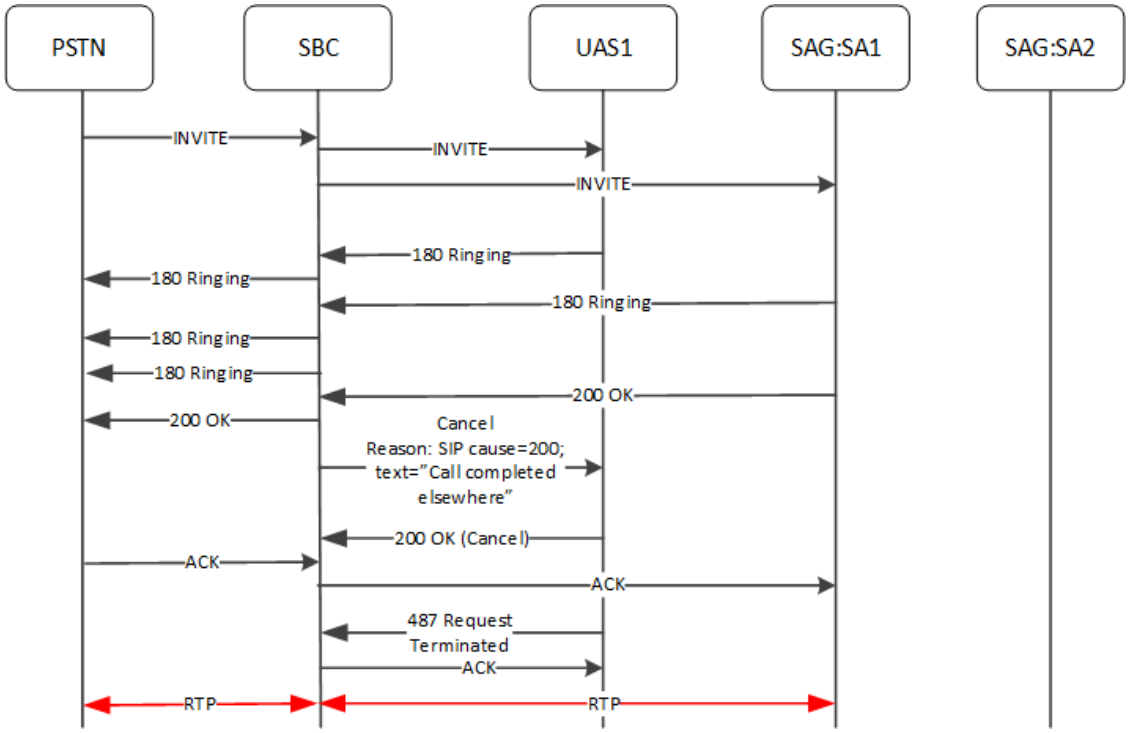
An important feature to support within parallel forking is early media. This next call flow shows all three destinations UAS1,UAS2 and UAS3 sending early media. The SBC is supporting this media, relaying it to the PSTN while the scenario finds an end station, UAS2 in this case, to answer the call. After the 200 OK comes from UAS2, the SBC cancels the setups to UAS1 and UAS3.

Key to this feature is the SBC latching to the most recent early (eg, the last) SDP/media flow when multiple endpoints (UAS below) send early SDP/media. The SBC makes any previous early media inactive, preventing the PSTN from having to handle multiple early media flows simultaneously.



Parallel Forking to a Session Agent Group

In this scenario, the SBC receives an INVITE that triggers a local policy to UAS1 and a local policy to a Session Agent Group. Although a session agent group can participate as a parallel forking target, it's members do not, with the SBC instead cycling through all other SAG members serially. Therefore, the SBC attempts to reach UAS1 and SAG:SA1 first, in parallel. In this flow, SAG:SA1 accepts the call and sends a 200 OK. So the SBC CANCELs UAS1 and never contacts SAG:SA2.



Configuring Routing

This section explains how to configure routing on the Oracle Communications Session Border Controller.

Configuration Prerequisite

You should have already configured the realms for your environment before you configure the routing elements. You need to know the realm identifier when configuring session agents and local policy.

You can use an asterisk (*) when the session agent exists in multiple realms.

Configuration Order

Recommended order of configuration:

- realm
- session agent
- session agent group
- local policy

Routing Configuration

You can enable, then configure, individual constraints that are applied to the sessions sent to the session agent. These constraints can be used to regulate session activity with the session agent. In general, session control constraints are used for session agent groups or SIP proxies outside or at the edge of a network. Some individual constraints, such as maximum sessions and maximum outbound sessions are not applicable to core proxies because they are

transaction stateful, instead of session stateful. Other constraints, such as maximum burst rate, burst rate window, maximum sustained rate, and sustained rate are applicable to core routing proxies.

Configuring Session Agents

To configure session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type session-agent and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **host-name**—Enter the name of the host associated with the session agent in either hostname or FQDN format, or as an IP address.

   If you enter the host name as an IP address, you do not have to enter an IP address in the optional IP address parameter. If you enter the host name in FQDN format, and you want to specify an IP address, enter it in the optional IP address parameter. Otherwise you can leave the IP address parameter blank to allow a DNS query to resolve the host name.

   If the initial DNS query for the session agent fails to get back any addresses, the session agent is put out-of-service. When session agent is pinged, the DNS query is repeated. The ping message is not sent until the DNS query gets back one or more IP addresses. After the query receives some addresses, the ping message is sent. The session agent remains out of service until one of the addresses responds.

   > **Note:**
   >
   > The value you enter here must be unique to this session agent. No two session agents can have the same hostname.

   The hostnames established in the session agent populate the corresponding fields in other elements.

5. **ip-address**—Optional. Enter the IP address for the hostname you entered in FQDN format if you want to specify the IP address. Otherwise, you can leave this parameter blank to allow a DNS query to resolve the host name.

6. **port**—Enter the number of the port associated with this session agent. Available values include:

   - zero (0)—If you enter zero (0), the Oracle Communications Session Border Controller will not initiate communication with this session agent (although it will accept calls).
   - 1025 through 65535

     The default value is **5060**.

> **Note:**
>
> If the transport method value is TCP, the Oracle Communications Session Border Controller will initiate communication on that port of the session agent.

7. **state**—Enable or disable the session agent by configuring the state. By default, the session agent is **enabled**.
   - enabled | disabled
8. **app-protocol**—Enter the protocol on which you want to send the message. The default value is **SIP**. Available values are:
   - SIP | H.323
9. **app-type**—If configuring H.323, indicate whether the application type is a gateway or a gatekeeper. Available values include:
   - **H.323-GW**—gateway
   - **H.323-GK**—gatekeeper
10. **transport-method**—Indicate the IP protocol to use (transport method) to communicate with the session agent. **UDP** is the default value. The following protocols are supported:
    - **UDP**—Each UDP header carries both a source port identifier and destination port identifier, allowing high-level protocols to target specific applications and services among hosts.
    - **UDP+TCP**—If the system determines that routes over TCP and UDP are both available, it initially uses UDP as the transport method. If a failure or timeout occurs in response to the UDP INVITE, the system uses TCP. When you select this transport method, the system always sends INVITEs via UDP as long as it receives a response.

      If the available route(s) only use one transport, either UDP or TCP, the system uses that transport method.
    -
    - **DynamicTCP**—Dynamic TCP connections are the transport method for this session agent. A new connection must be established for each session originating outbound from the SBC to the session agent. This connection is torn down at the end of a session.
    - **StaticTCP**—Static TCP connections are the transport method for this session agent. Once a connection is established, it will remain and not be torn down.
    - **StaticSCTP**—SCTP is used as the transport method.
    - **DynamicTLS**—Dynamic TLS connections are the transport method for this session agent. A new connection must be established for each session originating outbound from the SBC to the session agent. This connection is torn down at the end of a session.
    - **StaticTLS**—Static TLS connections are the transport method for this session agent. Once a connection is established, it will remain and not be torn down.
    - **ANY**—support all transport methods .
11. **realm-id**—Optional. Indicate the ID of the realm in which the session agent resides.

    The realm ID identifies the realm for sessions coming from or going to this session agent. For requests coming from this session agent, the realm ID identifies the ingress realm. For

requests being sent to this session agent, the realm ID identifies the egress realm. In a Oracle Communications Session Border Controller, when the ingress and egress realms are different, the media flows must be steered between the realms.

- no value: the egress realm is used unless the local policy dictates otherwise
- asterisk (*): keep the egress realm based on the Request URI

> **Note:**
>
> The realm ID you enter here must match the valid identifier value entered when you configured the realm.

12. **description**—Optional. Enter a descriptive name for this session agent.
13. **carriers**—Optional. Add the carriers list to restrict the set of carriers used for sessions originating from this session agent.

    Carrier names are arbitrary names that can represent specific service providers or traditional PSTN telephone service providers (for sessions delivered to gateways). They are global in scope, especially if they are exchanged in TRIP. Therefore, the definition of these carriers is beyond the scope of this documentation.

    You could create a list using carrier codes already defined in the North American Numbering Plan (NANP); or those defined by the local telephone number or carrier naming authority in another country.

    > **Note:**
    >
    > If this list is empty, any carrier is allowed. If it is not empty, only local policies that reference one or more of the carriers in this list will be applied to requests coming from this session agent.

14. **constraints**—Enable this parameter to indicate that the individual constraints you configure in the next step are applied to the sessions sent to the session agent. Retain the default value of **disabled** if you do not want to apply the individual constraints. Valid values are:

    - enabled | disabled

    > **Note:**
    >
    > In general, session control constraints are used for SAGs or SIP proxies outside or at the edge of a network.

15. Enter values for the individual constraints you want applied to the sessions sent to this session agent. The following table lists the available constraints along with a brief description and available values.

maximum sessions	Maximum number of sessions (inbound and outbound) allowed by the session agent. The range of values is: • minimum: zero (0) is the default value and means there is no limit • maximum: 4294967295

maximum outbound sessions	Maximum number of simultaneous outbound sessions (outbound from the Oracle Communications Session Border Controller) that are allowed from the session agent. The range of values is: • minimum: zero (0) is the default value and means there is no limit • maximum: 4294967295 The value you enter here cannot be larger than the maximum sessions value.
maximum burst rate	Number of session invitations allowed to be sent to or received from the session agent within the configured burst rate window value. SIP session invitations arrive at and leave from the session agent in intermittent bursts. By entering a value in this field, you can limit the amount of session invitations that are allowed to arrive at and leave from the session-agent. For example, if you enter a value of 50 here and a value of 60 (seconds) for the burst rate window constraint, no more than 50 session invitations can arrive at or leave from the session agent in that 60 second time frame (window). Within that 60-second window, any sessions over the limit of 50 are rejected. The range of values is: • minimum: zero (0) session invitations per second • maximum: 4294967295 session invitations per second Zero is the is the default value.
maximum sustain rate	Maximum rate of session invitations (per second) allowed to be sent to or received from the session agent within the current window. The current rate is determined by counting the number of session invitations processed within a configured time period and dividing that number by the time period. By entering a value in this field, you can limit the amount of session invitations that are allowed to arrive at and leave from the session agent over a sustained period of time. For the sustained rate, the Oracle Communications Session Border Controller maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes. For example, if you enter a value of 5000 here and a value of 3600 (seconds) for the sustain rate window constraint, no more than 5000 session invitations can arrive at or leave from the session agent in any given 3600 second time frame (window). Within that 3600-second window, sessions over the 5000 limit are rejected. The range of values is: • minimum: zero (0) invitations per second • maximum: 4294967295 invitations per second Zero is the is the default value. The value you set here must be larger than the value you set for the maximum burst rate constraint.

time to resume	Time in seconds after which the SIP proxy resumes sending session invitations to this session agent. This value only takes effect when the SIP proxy stops sending invitations because a constraint is exceeded. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295 seconds Default is zero.
time to resume (ttr) no response	Delay in seconds that the SIP proxy must wait from the time that it sends an invitation to the session agent and gets no response before it tries again. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295 seconds Default is zero. The value you enter here must be larger than the value you enter for the time to resume constraint.
in service period	Amount of time in seconds the session agent must be operational (once communication is re-established) before the session agent is declared as being in-service (ready to accept session invitations). This value gives the session agent adequate time to initialize. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295 seconds Default is zero.
burst rate window	Burst window period (in seconds) that is used to measure the burst rate. The term window refers to the period of time over which the burst rate is computed. Refer to the maximum burst rate information. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295seconds Zero is the is the default value. The value you set here must be smaller than the value you set for the maximum burst rate constraint.
sustain rate window	Sustained window period (in seconds) that is used to measure the sustained rate. Refer to the maximum sustain rate information. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295seconds Zero is the is the default value. The value you set here must be larger than the value you set for the maximum sustain rate constraint.

16. **req-uri-carrier-mode**—SIP only. Set whether you want the selected carrier (determined by a value in the local policy) added to the outgoing message by configuring the request uri carrier mode parameter.

    You can set this parameter to let the system perform simple digit translation on calls sent to gateways. A 3-digit prefix is inserted in front of the telephone number (the Request-URI) that the gateway will use to select a trunk group. Most often, the Oracle Communications Session Border Controller needs to insert the carrier code into the signaling message that it sends on.

    The default value is **none**. The following lists the available modes.

    - **none**—Carrier information will not be added to the outgoing message.
    - **uri-param**—Adds a parameter to the Request-URI. For example, cic-XXX.
    - **prefix**—Adds the carrier code as a prefix to the telephone number in the Request-URI (in the same manner as PSTN).

17. **proxy-mode**—SIP only. Indicate the proxy mode to use when a SIP request arrives from this session agent.

    If this field is empty (upon initial runtime or upgrade), it's value is set to the value of the SIP configuration's proxy mode by default. If no proxy mode value was entered for the SIP configuration, the default for this field is **proxy**.

    The following are valid proxy modes:

    - **proxy**—If the Oracle Communications Session Border Controller is a Session Router, the system will proxy the request coming from the session agent and maintain the session and dialog state. If the Oracle Communications Session Border Controller is a Session Director, the system behaves as a B2BUA when forwarding the request.
    - **redirect**—The system sends a SIP 3xx reDIRECT response with contacts (found in the local policy) to the previous hop.

18. **redirect-action**—SIP only. Indicate the action you want the SIP proxy to take when it receives a Redirect (3XX) response from the session agent.

    If the response comes from a session agent and this field is empty (upon initial runtime or upgrade), the redirect action will be recurse. If no session agent is found (for example, if a message comes from an anonymous user agent), the redirect action is set to proxy. If the Redirect (3xx) response does not have any Contact header, the response will be sent back to the previous hop.

    The following table lists the available proxy actions along with a brief description

    - **proxy**—The SIP proxy passes the response back to the previous hop; based on the pfoxy mode of the original request.
    - **recurse**—The SIP proxy serially sends the original request to the list of contacts in the Contact header of the response (in the order in which the contacts are listed in the response). For example, if the first one fails, the request will be send to the second, and so on until the request succeeds or the last contact in the Contact header has been tried.
    - **Recurse-305-only**—Recurses on the Contacts only in a 305 response.

19. **loose-routing**—SIP only. Enable this parameter if you want to use loose routing (as opposed to strict routing). The default is **enabled**. Valid values are:

    - enabled | disabled

      When the SIP NAT route home proxy parameter is enabled, the Oracle Communications Session Border Controller looks for a session agent that matches the home proxy address and checks the loose routing value. If loose routing is enabled, a

Route header is included in the outgoing request in accordance with RFC 3261. If loose routing is disabled, the Route header is not included in the outgoing request (in accordance with strict routing procedures defined in RFC 2543).

The loose routing value is also checked when the local policy's next hop value matches a session agent. If loose routing is set to enabled, the outgoing request retains the original Request-URI and Route header with the next hop address.

20. **send-media-session**—SIP only. Enable this parameter if you want to include a media session description (for example, SDP) in the INVITE or REINVITE message sent by the Oracle Communications Session Border Controller. Setting this field to **disabled** prevents the Oracle Communications Session Border Controller from establishing flows for that INVITE message.

    The default is **enabled**. Valid values are:

    - enabled | disabled

    > **Note:**
    >
    > Only set send media session to disabled for a session agent that always redirects requests. It returns an error or 3xx response instead of forwarding an INVITE message. In addition, do not disable send media session on session agents that support SIP-to-H.323 IWF call flows. This can cause call failure.

21. **response-map**—Optional and for SIP only. Enter the name of the response map to use for this session agent. The mappings in each SIP response map is associated with a corresponding session agent. You can also configure this value for individual SIP interfaces.

22. **ping-method**—SIP only. Indicate the SIP message/method to use to ping a session agent. The ping confirms whether the session agent is in service. If this field is left empty, no session agent will be pinged.

    Setting this field value to the OPTIONS method might produce a lengthy response from certain session agents and could potentially cause performance degradation on your Oracle Communications Session Border Controller.

23. **ping-interval**—SIP only. Indicate how often you want to ping a session agent by configuring the ping interval parameter. Enter the number of seconds you want the Oracle Communications Session Border Controller to wait between pings to this session agent. The default value is **0**. The valid range is:

    - Minimum: 0
    - Maximum: 999999999

      The Oracle Communications Session Border Controller only sends the ping if no SIP transactions (have occurred to/from the session agent within the time period you enter here.

24. **trunk-group**—Enter up to 500 trunk groups to use with this single session agent. Because of the high number of trunk groups you can enter, the ACLI provides enhanced editing mechanisms for this parameter:

    - You use a plus sign (+) to add single or multiple trunk groups to the session agent's list.

When you add a single trunk group, simply use the plus sign (+) in front of the trunk group name and context. Do not use a Space between the plus sign and the trunk group name and context.

For example, you might have already configured a list of trunk groups with the following entries: **tgrpA:contextA, tgrpB:contextB,** and **tgrpC:contextC**. To add **tgrp1:context1**, you would make the following entry:

```
ORACLE(session-agent)# trunk-group +tgrp1:context1
```

Your list would then contain all four trunk groups.

When you add multiple trunk groups, simply enclose your entry in quotation marks () or in parentheses (()). While you put spaces between the trunk group name and context entries, you do not use spaces with the plus sign, parentheses or quotation marks.

```
ORACLE(session-agent)# trunk-group +tgrp1:context1 tgrp2:context2
tgrp3:context3
```

- You use a minus sign (-) to delete single or multiple trunk groups from the session agent's list.

  When you remove a single trunk group, simply use the minus sign (-) in front of the trunk group name and context. Do not use a Space between the minus sign and the trunk group name and context.

  For example, you might have already configured a list of trunk groups with the following entries: tgrpA:contextA, tgrpB:contextB, tgrpC:contextC, and tgrp1:context1. To delete tgrp1:context1 from the list, you would make the following entry:

  ```
  ORACLE(session-agent)# trunk-group -tgrp1:context1
  ```

  Your list would then contain: tgrpA:contextA, tgrpB:contextB, and tgrpC:contextC.

  When you add multiple trunk groups, simple enclose your entry in quotation marks () or in parentheses (()). While you put spaces between the trunk group name and context entries, you do not use spaces with the plus sign, parentheses or quotation marks.

  ```
  ORACLE(session-agent)# trunk-group -tgrp1:context1 tgrp2:context2
  ```

- You overwrite (replace) the entire list of a session agent's trunk groups by entering a list that does not use either the plus (+) or the minus (**-**) sign syntax.

25. **ping-in-service-response-codes**—SIP only. Enter the list of response codes that keep a session agent in service when they appear in its response to the Oracle Communications Session Border Controller's ping request. The Oracle Communications Session Border Controller takes the session agent out of service should a response code be used that does not appear on this list. Default is **none**.
26. **out-service-response-codes**—SIP only. Enter the list defines the response codes that take a session agent out of service when they appear in its response to the Oracle Communications Session Border Controller's ping request or any in-dialog creating request (such as an INVITE, SUBSCRIBE, etc.). The Oracle Communications Session Border Controller ignores this list if an in-service list exists.
27. **options**—Optional. You can add your own features and/or parameters by using the options parameter. You enter a comma-separated list of either or both of the following:

- feature=<value feature>

For example:

You can include the original address in the SIP message from the Oracle Communications Session Border Controller to the proxy in the Via header parameter by entering the following option:

via-origin=<parameter-name>

The original parameter is included in the Via of the requests sent to the session agent. The via origin feature can take a value that is the parameter name to include in the Via. If the value is not specified for via origin, the parameter name is origin.

> **Note:**
>
> If the feature value itself is a comma-separated list, enclose it within quotation marks.

28. **media-profiles**—Optional and for H.323 only. You can enter a list of media profiles to open logical channels when starting an outgoing call as a Fast Start H.323 call.

    Values you enter here must start with either an alphabetical character from A through Z (A-Za-z) or with an underscore (_). After the first character, each list entry can contain any combination of alphabetical or numerical characters (0-9A-Za-z), as well as the period (.), the dash (-), and the underscore (_). For example, netnet_mediaprofile1.

    You can enter 1 to 24 characters.

    > **Note:**
    >
    > The values you enter here must correspond to a valid name you entered when you configure the media profile.

29. **in-translationid**—Optional. Enter the In Translation ID for a configured session translation (group of address translation rules with a single ID) if you want to apply session translation to incoming traffic.

30. **out-translationid**—Optional. Enter the Out Translation ID for a configured session translation (group of address translation rules with a single ID) if you want to apply session translation to outgoing traffic.

    Address translations attached to session agents take precedence over address translations attached to realms. If no address translation is applied to a session agent, then the Oracle Communications Session Border Controller will use the address translation applied to a realm. If an address translation is applied to both a realm and session agent, the translation attached to the session agent will apply. If the applicable session agent and realm have no associated translations, then the addresses will remain in their original forms and no address translations will be performed.

31. **trust-me**—Indicate whether this session agent is a trusted source, which the Oracle Communications Session Border Controller checks when it receives a message to determine if the source is trusted. The default value is **disabled**. The valid values are:

    - enabled | disabled

The following example shows a session agent with an IP address used for the hostname.

```
session-agent
        hostname                        192.168.1.10
        ip-address                      192.168.1.10
        port                            5060
        state                           enabled
        app-protocol                    SIP
        app-type
        transport-method                UDP
        realm-id                        realm-1
        description                     englab
        carriers
                                        carrier1
        constraints                     disabled
        max-sessions                    355
max-inbound-sessions            4
        max-outbound-sessions           355
        max-burst-rate                  0
        max-inbound-burst-rate          10
        max-outbound-burst-rate         1
        max-sustain-rate                3000
        max-inbound-sustain-rate        0
        max-outbound-sustain-rate       0
        min-seizures                    5
        min-asr                         0 time-to-resume                60
        ttr-no-response                 0
        in-service-period               30
        burst-rate-window               60
        sustain-rate-window             3600
        req-uri-carrier-mode            None
        proxy-mode                      Proxy
        redirect-action                 Recurse
        loose-routing                   enabled
        send-media-session              enabled
        response-map
        ping-method
        ping-interval                   0
        media-profiles
        in-translationid
        out-translationid
        trust-me                        disabled
        request-uri-headers
        stop-recurse
        local-response-map
        ping-to-user-part
        ping-from-user-part
        li-trust-me                     disabled
        in-manipulationid
        out-manipulationid
        p-asserted-id
        trunk-group
        max-register-sustain-rate       0
```

32. **static-tcp-source-port**—If using TCP or TLS transport, you may need to specify the source port of the SBC. If so, use the **static-tcp-source-port** parameter to specify the port

that the SBC uses as a source port while making an outbound TCP connection to that session agent.

Some environments do not support ephemeral port assignment and require that the SBC use a fixed/static TCP port to make a connection to a **session-agent**. Configuring this parameter allows the SBC to make connections to those session agents using the specified port number.

```
ORACLE(session-agent)# static-tcp-source-port 5061
```

- Values: 0 is default
- Min: 1025 / Max: 65535
  The system performs error checking on your values and prevents you from entering an invalid port number. Enabling this feature requires that you reboot the SBC.

Session Agent Group Configuration

To configure session agent groups:

1. Access the **session-agent-group** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-group
   ORACLE(session-agent-group)#
   ```

2. **group-name**—Enter a unique name for the session agent group in Name format.
3. **description**—Optional. Enter descriptive information about the session agent group.
4. **state**—Enable or disable the session agent group on the Oracle Communications Session Border Controller. The default value is **enabled**. Valid values are:
   - enabled | disabled
5. **application-protocol**—Indicate the signaling protocol you want to use with the session agent group. The default value is **SIP**. The valid values are:
   - SIP | H.323
6. **strategy**—Indicate the session agent allocation strategy you want to use. The strategy you chose selects the session agents that will be made available by this session agent group. The default value is **hunt**. The valid values are:
   - **hunt**—Selects session agents in the order in which they are listed. For example, if the first agent is online, working, and has not exceeded defined constraints, all traffic is sent to the first agent. If the first agent is offline or if it exceeds a defined constraint, the second agent is selected. If the first and second agents are offline or exceed defined constraints, the third agent is selected. And so on through the list of session agents.
   - **roundrobin**—Selects each session agent in the order in which they are listed in the destination list, selecting each agent in turn, one per session.
   - **leastbusy**—Selects the session agent that has the fewest number of sessions relative to the maximum sessions constraint (for example, lowest percent busy) of the session agent element. The Least Busy Calculation is the result of dividing the number of active calls for a session agent by the max-sessions parameter within the session-agent element configuration. If the default max-session parameter value issued for a session agent (0), the result of the Least Busy Calculation will be 0. The Least Busy

SAG Strategy will route a session to the session agent with the lowest resulting Least Busy Calculation percentage. If multiple session agents have the lowest percentage, the foremost session agent in the Session Agent Group dest parameter will be used.

- **propdist**—Based on programmed, constrained session limits, the Proportional Distribution strategy proportionally distributes the traffic among all of the available session agents. Sessions are distributed among session agents based on the max-outbound-sessions value in each session agent. The sum of max-outbound-sessions for every session-agent within a session group equates to 100% and the max-outbound-sessions value for each session-agent represents a % that total. Sessions are proportionally allocated to session agents based on their individual session agent max-outbound-sessions value, as a % of the total max-outbound-sessions for the group.
- **lowsusrate**—The Lowest Sustained Rate strategy routes to the session agent with the lowest sustained rate of session initiations/invitations (based on observed sustained session request rate).

7. **destination**—Identify the destinations (session agents) available for use by this session agent group.

   A value you enter here must be a valid IP address or hostname for a configured session agent.

8. **trunk-group—**Enter trunk group names and trunk group contexts to match in either IPTEL or custom format. If left blank, the Oracle Communications Session Border Controller uses the trunk group in the realm for this session agent group. Multiple entries are surrounded in parentheses and separated from each other with spaces.

   Entries for this list must one of the following formats: trgp:context or trgp.context.

9. **sag-recursion**—Enable this parameter if you want to use SIP SAG recursion for this SAG. The default value is **disabled**. Valid values are:

   - enabled | disabled

10. **stop-sag-recurse**—Enter the list of SIP response codes that terminate all further recursions, including those external to the SAG. Upon receiving one of the specified response codes, such as 401 unauthorized, or upon generating one of the specified response codes internally, such as 408 timeout, the Oracle Communications Session Border Controller returns a final response to the UAC and stops trying to route the message. This includes not attempting to contact higher-cost SAs.

    You can enter the response codes as a comma-separated list or as response code ranges.

11. Type **done** to save your configuration.

SAG Matching for LRT and ENUM

When this feature is enabled and a match is found, the Oracle Communications Session Border Controller uses the matching SAG for routing. When there is no match for the SAG, the Oracle Communications Session Border Controller processes the result as it would have if this feature had not been enabled: either matching to a session agent hostname, or performing a DNS query to resolve it.

Note that you set the state of this feature in the SIP configuration.

To configure a SAG for ENUM or LRT matching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

If you are adding support for this feature to a pre-existing SIP configuration, then you must select (using the ACLI **select** command) that configuration to edit it.

4. **enum-sag-match**—Set this parameter to enabled so the Oracle Communications Session Border Controller will match session agent group (SAG) names with the hostname portion in the naming authority pointer (NAPTR) from an ENUM query or LRT next-hop entry. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

Configuring Local Policy

To configure local policy:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
```

3. Type **local-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ACMEPACKET(session-router)# local-policy
ACMEPACKET(local-policy)#
```

4. **from-address**—Indicate the originating address information by entering a From address value. You can use the asterisk (*) as a wildcard to indicate this policy can be used with all originating addresses.

   You can also use complete or partial E.164 addresses (strings that contain telephone keypad characters) here. Number matching works from left to right. Formats include the following:
   - SIP From address
   - FQDNs
   - IP addresses
   - H.323 CallingPartyAddress
     The Oracle Communications Session Border Controller also supports the asterisk as part of the From address you configure in your local policies.

This means that for the **from-address** parameters of a local policy configuration, you can enter values in which an asterisk appears and match them accordingly. You might enter a value that resemble the following example:

- 123*456

After entering the from-address value, the Oracle Communications Session Delivery Manager automatically saves it to the configuration when exiting from local policy.

5. **to-address**—Indicate the destination address by entering a To address value. You can use the asterisk (*) as a wildcard to indicate all this policy can be used for any destination address.

   You can also use E.164 addresses (strings that contain telephone keypad characters) here. Number matching works from left to right. Formats include the following:

   - SIP Request-URI
   - FQDNs
   - IP addresses
   - H.323 CalledPartyAddress
     The system also supports the asterisk as part of the To address you configure in your local policies.

     This means that for the **to-address** parameters of a local policy configuration, you can enter values in which an asterisk appears and match them accordingly. You might enter a value that resembles the following example:
   - 123*456

   After entering the to-address value, the Oracle Communications Session Delivery Manager automatically saves it to the configuration when exiting from local policy.

6. **source-realm**—Enter the realm, or list of realms, you want the Oracle Communications Session Border Controller to use to determine how to route traffic. This list identifies from what realm traffic is coming and is used for routing by ingress realm by the local policy.

   You can use the asterisk (*) as a wildcard to indicate this local policy can be used with all realms. The default value is *****.Or you can enter a value that corresponds to the identifier of an already configured realm. Formats include the following:

   - realm ID
   - customer name
   - peer name
   - subdomain name
   - VPN identifier

7. **activate-time**—Set the time you want the local policy to be activated using the following syntax:

   ```
   yyyy-mm-dd hh:mm:ss
   yyyy-mm-dd-hh:mm:ss
   ```

8. **deactivate-time**—Set the time you want the local policy to be deactivated using the following syntax:

   ```
   yyyy-mm-dd hh:mm:ss
   yyyy-mm-dd-hh:mm:ss
   ```

9. **state**—Indicate whether you want the local policy to be enabled or disabled on the system. The default value is **enabled**. The valid values are:
   - enabled
   - disabled
10. **parallel-forking**—Enable if you want the local policy to support parallel forking or disabled on the system. The default value is **disabled**. The valid values are:
   - enabled
   - disabled
11. **policy-priority**—Specify the priority for this local policy. The default value is **none**. The valid values are:
   - none
   - normal
   - non-urgent
   - urgent
   - emergency
12. **policy-attribute**—Configure local policy attributes by following steps 8 through 21.
13. **next-hop**—Identify the next signaling host by entering the next hop value. You can use the following as next hops:
   - IPv4 address or IPv6 address of a specific endpoint
   - Hostname or IPv4 address or IPv6 address of a configured session agent
   - Group name of a configured session agent group

   > **Note:**
   >
   > The group name of a configured session agent group must be prefixed with SAG. For example:
   >
   > – next-hop SAG:appserver
   > – next-hop lrt:routetable
   > – next-hop enum:lerg

   You can also configure a next hop that has an address of 0.0.0.0, thereby creating a null route. Different from not having a local policy configured (which would trigger Oracle Communications Session Border Controller local policy recursion), this terminates local policy recursion and immediately fails the request. In these cases, the system responds a request with a 404 Not Found.
14. **realm**—Identify the egress realm (the realm used to reach the next hop) if the system must send requests out from a specific realm.

   The value you enter here must correspond to a valid identifier you enter when you configured the realm. If you do not enter a value here, and the next hop is a session agent, the realm identified in the session agent configuration is used for egress. In H.323, the next hop address is matched against the realm's address prefix to determine the realm.
15. **replace-uri**—Indicate whether you want to replace the Request-URI in outgoing SIP requests with the next hop value.

16. **carrier**—Optional. Enter the name of the carrier associated with this route. The value you enter here must match one or more of the carrier names in the session agent configuration.

    Entries in carrier fields can be from 1 to 24 characters in length and can consist of any alphabetical character (Aa-Zz), numerical character (0-9), or punctuation mark (! ” # $ % ^ & * ( ) + - = < > ? ‘ | { } [ ] @ / \ ‘ ~ , . _ : ;) or any combination of alphabetical characters, numerical characters, or punctuation marks. For example, both 1-0288 and acme_carrier are valid carrier field formats.

17. **start-time**—Indicate the time of day (from the exact minute specified) the local policy attributes go into effect. Enter only numerical characters (0-9) and follow the 4-digit military time format. For example:

    ```
    1400
    ```

    The default value of **0000** implies that the defined policy attributes can be considered in effect any time after 00:00:00. The valid range is:

    - Minimum—0000
    - Maximum—2400

18. **end-time**—Indicate the time of day (from the exact minute specified) the local policy attributes are no longer in effect. Enter only numerical characters (0-9) and follow the 4-digit military time format. For example:

    ```
    2400
    ```

    The default value of **2400** implies that the defined policy attributes can be considered in effect any time before midnight. The valid range is:

    - Minimum—0000
    - Maximum—2400

19. **days-of-week**—Enter any combination of days of the week (plus holidays) you want the local policy attributes to be in effect. You must enter at least one day or holiday here. A holiday entry must correspond with a configured holiday established in the Session Router.

    The default is **U-S**. The valid values are:

    - U (Sunday)
    - M (Monday)
    - T (Tuesday)
    - W (Wednesday)
    - R (Thursday)
    - F (Friday)
    - S (Saturday)
    - H (Holiday)
      You can enter a range of values separated by a hyphen, for example U-S. And you can enter multiple values separated by commas, for example M,W,F. You cannot use spaces as separators.

20. **cost**—Enter a cost value that acts as a unitless representation of the cost of a route relative to other routes reaching the same destination (To address). This value is used as a way of ranking policy attributes.

The default value is zero (**0**). The valid values are:

- minimum—zero (0)
- maximum—999999999

21. **app-protocol**—Enter the signaling protocol to use when sending messages to the next hop. The valid values are:
    - H.323
    - SIP
22. **state**—Indicate whether you want to enable or disable the local policy. The default value is **enabled**. The valid values are:
    - enabled
    - disabled
23. **media-profiles**—Configure a list of media profiles if you want the local policy to route SIP and H.323 traffic by the codecs specified in the SDP. The list of media profiles entered here are matched against the SDP included in SIP or H.323 requests and the next hop is selected by codec.

    The values in this list are matched against the rtpmap attribute of passed SDP, and preference weight for route selection is based on the order in which the matching payload type appears in the SDP's media (m=) line.
    For example when the following SDP arrives:

    ```
    m=audio 1234 RTP/AVP 0 8 18
    ```

    that contains the following attributes that correspond to three configured local policies with the same cost:

    - a=rtpmap:**0** PCMU/8000
    - a=rtpmap:**8** PCMA/8000
    - a=rtpmap:**18** G729/8000
      The following route selection action occurs:

      The local policy route that corresponds to the a=rtpmap:**0** PCMU/8000 attribute is selected because the payload type of **0** in the attribute line matches the first payload type of 0 listed in the m= line. The codec value of PCMU indicated in this selected attribute is used to find the local policy with the media profiles attribute that includes PCMU in the list.

      Because the value you enter here is matched against the codec values included in the actual passed SDP, it must correspond to accepted industry-standard codec values.

      The following example shows a local policy with a next hop value of the session agent group called gw-sag2.

      ```
      local-policy
              from-address
                                              *
              to-address
                                              192.168.1.10
              source-realm                    *
              activate-time                   2005-01-20 20:30:00
              deactivate-time                 N/A
              state                           enabled
      ```

```
        last-modified-date              2005-01-10 00:36:29
policy-attribute
        next-hop                        SAG:gw-sag2
        realm
        replace-uri                     enabled
        carrier
        start-time                      0000
        end-time                        2400
        days-of-week                    U-S
        cost                            0
        app-protocol
        state                           enabled
        media-profiles
```

Local Policy Matching for Parent Realms

For SIP and H.323, you can configure the Oracle Communications Session Border Controller to use the parent realm for routing purposes even when the source realm for an incoming message is a child realm.

With this feature disabled (default), the Oracle Communications Session Border Controller uses the specific source realm to perform a local policy look-up. When the source realm is a child realm and any relevant local policies are configured with the parent realm, there will be no matches and the local policy look-up will fail. To avoid this issue and ensure successful look-ups, you must configure multiple local policies if you want to use a configuration with nested realms.

The Oracle Communications Session Border Controller examines the source realm to determine if it is a parent realm with any child realms when you enable this feature. If the parent, source realm does have child realms, then the Oracle Communications Session Border Controller creates local policy entries for the parent and all of its child realms. This operation is transparent and can save time during the configuration process.

It is possible, then, for a local policy look-up to match the same child realm in two ways:

- Through a match via the parent realm
- Through a direct match for a local policy configured with that specific child realm

In such a case, the child realm must have different costs for each type of match to avoid collisions.

This feature is enabled on a global basis in the session router configuration. Because it applies system-wide, all source realms will use this form of matching when enabled.

To enable local policy matching for parent realms:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-router** and press Enter.

```
ORACLE(session-router)# session-router
ORACLE(session-router-config)#
```

4. **match-lp-source-parent-realms**—If you want the Oracle Communications Session Border Controller to perform local policy realm matching based on the parent realm (so that there are local policy entries for parent and child realms), set this parameter to **enabled**. The default value is **disabled**. The valid values are:

   - enabled | disabled

```
ORACLE(session-router-config)# match-lp-src-parent-realms enabled
```

5. Save and activate your configuration.

SIP Session Agent DNS-SRV Load Balancing

The Oracle Communications Session Border Controller provides the ability to specify an FQDN (fully qualified domain name) for a destination session-agent. During DNS lookup the FQDN can resolve to multiple SRV (Resource Record for Servers) records. Each SRV can resolve to a single IP address via A-Record query.

The Oracle Communications Session Border Controller also supports load balancing behavior as described in RFC 3263, Session Initiation Protocol (SIP): Locating SIP Servers.

The **ping-all-addresses** parameter in session-agent configuration mode enables internal load balancing and RFC 3263 compliance. The Oracle Communications Session Border Controller monitors the availability of the dynamically resolved IP addresses obtained from DNS using OPTIONS pings (ping-per-DNS entry). The ping-method and ping-interval for each resolved IP address is copied from the original session-agent.

Status of Session-Agent:

In Service – if any of dynamically resolved IP addresses is in service

Out of service – if all dynamically resolved IP addresses is out of service.

The default of **ping-all-addresses** is disabled, in which case the Oracle Communications Session Border Controller only pings the first available resolved IP addresses.

With status of each resolved IP addresses above, the Oracle Communications Session Border Controller recurses through the list of these in-service IP addresses dynamically resolved from DNS server on 503 response, and stop recursion based upon a configured list of response values specified by the **stop-recurse** parameter in sip-interface configuration mode. With internal load balancing enabled in the session-agent, the Oracle Communications Session Border Controller provides the ability to select routing destinations based on SRV weights. The priority/weight algorithm is based on RFC 2782, *A DNS RR for specifying the location of services (DNS SRV)*.

The Oracle Communications Session Border Controller provides the similar functionality as that listed above for A-records, the Oracle Communications Session Border Controller selects the first available routing destinations because there is no priority/weight contained in A-records.

Statistics and Traps on Agents

You can configure the SBC to track activity on individually resolved agents by enabling the **sa-routes-stats** and **sa-routes-traps** parameters in the **sip-config**. This feature also requires

that the session agent's **ping-all-addresses** function be active. This functionality requires the status checks on all agents enabled by the **ping-all-addresses** parameter. Extended functionality includes:

- **sa-routes-stats**—Extends the statistics collection function on DNS resolved session-agents
  - Extends the **show sipd agents** command to support additional arguments and output, including agent FQDN and specific SIP methods.
  - Enables HDR to generate records for each DNS resolved **session-agent** route.
- **sa-routes-traps**—Extends operation on DNS resolved **session-agent** to issue traps when a session agent route changes state.

Session Agent DNS-SRV Load Balancing Configuration

To configure the Oracle Communications Session Border Controller to perform Session-Agent DNS-SRV load balancing:

1. From superuser mode, use the following command sequence to access sip-config configuration mode. While in this mode, you configure SAG-based address resolution.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

2. Use the **ping-all-addresses** parameter to enable Session-Agent DNS-SRV load balancing.
3. Use **done**, **exit**, and **verify-config** to complete Session-Agent DNS-SRV load balancing configuration.

   The **show sip agents** ACLI command displays the availability of dynamically resolved IP addresses

   ```
   ORACLE# show sip agents acme.engr.com
   21:46:05-51-router
   Session Agent acme.engr.com(core) [In Service] NO ACTIVITY

   ... Statistics of ALL IPs associated with this SA ....

   Destination: 192.168.200.235 In Service
   Destination: 192.168.200.231 In Service
   ```

Answer to Seizure Ratio-Based Routing

New SIP session agent constraints set a threshold for Answer to Seizure Ratio (ASR) has been implemented. ASR is considered when determining whether session agents are within their constraints to route calls (in addition to session and rate constraints).

The new session agent constraints indicate the minimum acceptable ASR value and computes the ASR while making routing decisions. ASR is calculated by taking the number of successfully answered calls and dividing by the total number of calls attempted (which are known as seizures).

If the ASR constraints are exceeded, the session agent goes out of service for a configurable period of time and all traffic is routed to a secondary route defined in the local policy (next hop with higher cost).

The two session agent constraints are:

- minimum seizure: determines if the session agent is within its constraints. When the first call is made to the session agent or the if calls to the session agent are not answered, the minimum seizure value is checked.
  For example, if 5 seizures have been made to the session agent and none of them have been answered, the sixth time, the session agent is marked as having exceeded its constraints and the calls will not be routed to it until the time-to-resume has elapsed.
- minimum ASR: considered when make routing decisions. If some or all of the calls to the session agent have been answered, the minimum ASR value is considered to make the routing decisions.
  For example, if the you set the minimum ASR at 50% and the session agent's ASR for the current window falls below 50%, the session agent is marked as having exceeded its constraints and calls will not be routed to it until the time-to-resume has elapsed.

ASR Constraints Configuration

To configure ASR constraints:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. If configuring an existing session agent, enter the select command to select the session agent.
5. **min-seizures**—Enter the minimum number of seizures that when exceeded, cause the session agent to be marked as having exceeded its constraints. Calls will not be routed to the session agent until the time-to-resume has elapsed. The default value is **5**. The valid range is:
   - Minimum—1
   - Maximum—999999999
6. **min-asr**—Enter the percentage you want as the minimum. If the session agent's ASR for the current window falls below this percentage, the session agent is marked as having exceeded its constraints and calls will not be routed to it until the time-to-resume has elapsed. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—100
7. Save and activate your configuration.

The following example shows a session agent configuration.

```
session-agent
        hostname                        192.168.1.6
        ip-address
        port                            1720
        state                           enabled
        app-protocol                    H323
        app-type                        H323-GW
        transport-method
        realm-id                        external
        description
        carriers
        constraints                     enabled
        max-sessions                    0
max-inbound-sessions            4
        max-outbound-sessions           5
        max-burst-rate                  0
        max-inbound-burst-rate          10
        max-outbound-burst-rate         1
        max-sustain-rate                0
        max-inbound-sustain-rate        0
        max-outbound-sustain-rate       0
min-seizures                    5
        min-asr                         50
        time-to-resume                  30
        ttr-no-response                 0
        in-service-period               0
        burst-rate-window               0
        sustain-rate-window             0
        req-uri-carrier-mode            None
        proxy-mode
        redirect-action
        loose-routing                   enabled
        send-media-session              enabled
        response-map
        ping-method
        ping-interval                   0
        media-profiles
        in-translationid
        out-translationid
        trust-me                        disabled
        request-uri-headers
        stop-recurse
        local-response-map
        ping-to-user-part
        ping-from-user-part
        li-trust-me                     disabled
        in-manipulationid
        out-manipulationid
        p-asserted-id
        trunk-group
        max-register-sustain-rate       0
        early-media-allow
```

```
invalidate-registrations       disabled
last-modified-date             2006-05-12 19:48:06
```

SIP Recursion Policy

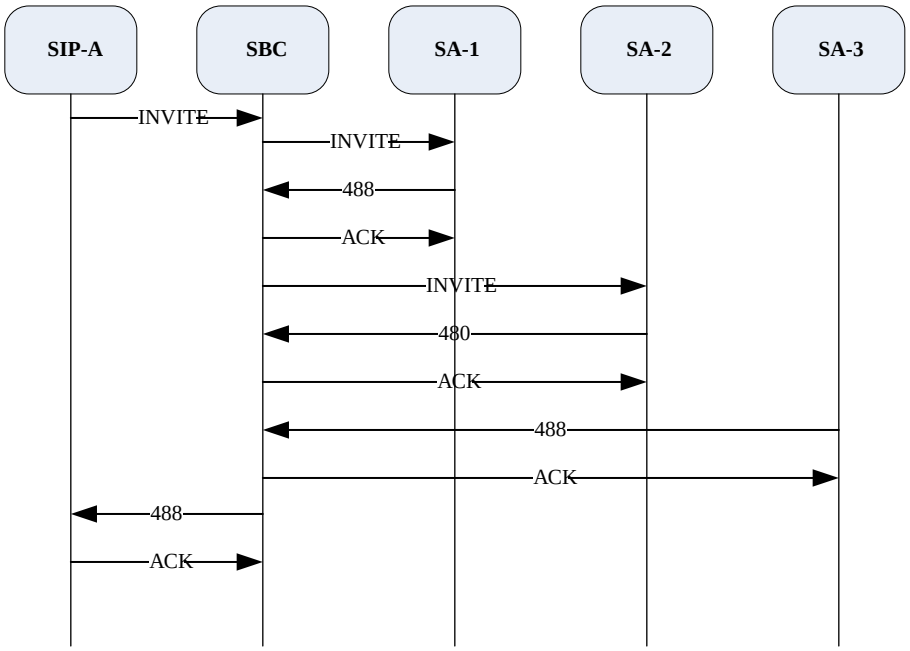
Session Agents (and Session agent groups) can utilize a SIP Recursion policy to customize the Oracle Communications Session Border Controller (SBC) behavior when recursing through a list of target SIP peers. These policies are useful for networks with large numbers of SIP peers that wish to customize recursive routing behavior for individual session agents or session agent groups, based on number of recursion attempts or the returned SIP response code.

SIP recursion policy provides the SBC with rules to indicate when to resend messages to the next SIP peer based on the previous response before terminating recursion and forwarding the final peer's response back to the initial requester.

The system terminates recursing among session agents (or session agent groups) when one of two criteria is first met. The first criteria is reaching a maximum number of recursions among all targets to a given SIP request for all configured response codes. The second criteria is receiving a maximum number of a configured response code from all targets as recursed for a given SIP request.

To set a maximum number of recursions before forwarding the final response back to the ingress-side requester, you set the **sip-recursion-policy**, **global-count** parameter and the desired **response-code**s. Once the SBC exceeds retrying to send a message to valid targets this number of times, it stops recursing and forwards the final response back to the requester. To disable maximum recursion hops per-call, set the **sip-recursion-policy**, **global-count** to 0.

When **sip-recursion-policy**, **global-count** is set to 3, with no additional configuration, the behavior shown in the following diagram occurs; after the SBC receives the 3rd response code, recursion stops and the final response is forwarded to the requestor.



SIP recursion policy can also terminate recursion based on receiving one or more of the individually configured 3xx, 4xx, or 5xx response messages received from all targets in

response to the request. The SBC can consider the number of responses per response code that were received before terminating recursion.

In absolute mode, the SBC stops recursing after the total configured number of responses (**sip-response-code**, **attempts**) for a configured **sip-response-code** subelement has been received. When the number of attempts per response code is received, recursion stops and the final response is forwarded to the requestor. Considering the previous call flow, in the following configuration example, after the SBC receives the 2nd 488 response from among all targets, it terminates recursion and forwards the last 488 to the requester:

```
sip-response-code
                response-code                        488
                attempts                             2
```

Consecutive mode sets the SBC to stop recursing after the total configured number of responses (**sip-response-code**, **attempts**) for a configured **sip-response-code** subelement has been received consecutively. When the number of attempts per response code is received, recursion stops and the final response is forwarded to the requestor. Considering the previous example's configuration set to consecutive mode, the SBC will need to receive two 488 responses in a row before recursion terminates and the last 488 is forwarded back to the requester.

Final Configuration

In order to use a **sip-recursion-policy**, it must be configured within a session agent or session agent group. Populate a **session-agent-group**, **sip-recursion-policy** or **session-agent**, **sip-recursion-policy** parameter with the **name** of the SIP recursion policy you wish to apply.

Feature Interactions

The existing SIP Configurable Route Recursion feature takes higher precedence than this feature. If the **stop-recurse** parameter is configured in the SIP interface, session agent, or SAG, the SBC checks each response against that list, before any SIP recursion policy.

SIP Recursion Policy Configuration

1. Access the **sip-recursion-policy** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-recursion-policy
   ORACLE(sip-recursion-policy)#
   ```

2. name - Set a name for this SIP Recursion Policy. This value will be referenced by individual session agents' or session agent groups' **sip-recursion-policy** parameter.
3. description - Enter a textual description of this SIP Recursion Policy instance. If the description includes spaces, enclose all words within double quotes.
4. global-count - Enter the maximum number of recursions to take before terminating recursion and sending the response back to the requester. Entering 0 here disables a maximum recursion counter.
5. mode - If terminating recursion based on responses received, enter whether to terminate on absolute responses or consecutive responses. This value is set if also configuring **sip-response-code** subelements. Valid values are:
   - absolute

- consecutive

6. Type **done** to save your configuration.

If configuring the Oracle Communications Session Border Controller to terminate recursion on received response, continue this procedure to create one or more **sip-response-code** subelement.

7. Access the **sip-response-code** subelement.

```
ORACLE(sip-recursion-policy)# sip-resp-code-attempts
ORACLE(sip-response-code)#
```

8. response-code - Enter the SIP response code number to associate with an attempt number through this configuration element for when to terminate recursion.
   - Default: 503
   - Range: 300 - 599
9. attempts - Enter the number of times the system must receive a message of the **response-code** before terminating recursion. This value is handled according to the **sip-recursion-policy**, **mode** parameter.
   - Default: 1
   - Range: 1 - 1000
10. Type **done** to save your configuration.

Configure the appropriate **session-agent**, **sip-recursion-policy** or **session-agent-group**, **sip-recursion-policy** parameter with the **name** of the SIP Recursion policy you just created.

ENUM Lookup

Telephone Number Mapping (ENUM from TElephone NUmber Mapping) is a suite of protocols used to unify the telephone system with the Internet by using E.164 addresses with the Domain Name System (DNS). With ENUM, an E.164 number can be expressed as a Fully Qualified Domain Name (FQDN) in a specific Internet infrastructure domain defined for this purpose (e164.arpa). E.164 numbers are globally unique, language independent identifiers for resources on Public Switched Telecommunication Networks (PSTNs). ITU-T recommendation E.164 is the international public telecommunication telephony numbering plan.

How ENUM Works

ENUM uses DNS-based architecture and protocols for mapping a complete international telephone number (for example, +1 202 123 1234) to a series of Uniform Resource Identifiers (URIs).

The protocol itself is defined in the document E.164 number and DNS (RFC 3761) that provides facilities to resolve E.164 telephone numbers into other resources or services on the Internet. The syntax of Uniform Resource Identifiers (URIs) is defined in RFC 2396. ENUM uses Naming Authority Pointer (NAPTR) records defined in RFC 2915 in order to identify available ways or services for contacting a specific node identified through the E.164 number.

Translating the Telephone Number

A telephone number is translated into an Internet address using the following steps:

1. The number is first stored in the following format, +1-202-555-1234. 1 is the country code for the United States, Canada, and the seventeen other countries that make up the North American Numbering Plan (NANP). The + indicates that the number is a complete, international E.164 telephone number.
2. All characters are removed except for the digits. For example, 12025551234.
3. The order of the digits is reversed. For example, 43215552021. The telephone number is reversed because DNS reads addresses from right to left, from the most significant to the least significant character. Dots are placed between each digit. Example: 4.3.2.1.5.5.5.2.0.2.1. In DNS terms, each digit becomes a zone. Authority can be delegated to any point within the number.
4. A domain (for example, e164.arpa) is appended to the end of the numbers in order to create a FQDN. For example,4.3.2.1.5.5.5.2.0.2.1.e164.arpa.
5. The domain name is queried for the resource records that define URIs necessary to access SIP-based VoIP.

   Once the authoritative name server for that domain name is found, ENUM retrieves relevant records and uses that data to complete the call or service. For example, the number 12025551234 returns sip:my.name@bigcompany.com.

About NAPTR Records

ENUM uses NAPTR records for URI resource records. NAPTR records are used to translate E.164 addresses to SIP addresses. An example of a NAPTR record is:

```
$ORIGIN 4.3.2.1.5.5.5.2.0.2.1.e164.arpa.
IN NAPTR 100 10 "u" "sip+E2U"  "!^.*$!sip:phoneme@example.net!"
```

This example specifies that if you want to use the "sip+E2U" service, you should use sip:phoneme@example.net as the address.

The regular expression can be used by a telephone company to easily assign addresses to all of its clients. For example, if your number is +15554242, your SIP address is sip:4242@555telco.example.net; if your number is +15551234, your SIP address is sip:1234@555telco.example.net.

About the Oracle Communications Session Border Controller ENUM Functionality

The ENUM functionality lets the Oracle Communications Session Border Controller make an ENUM query for a SIP request. The ENUM lookup capability lets the Oracle Communications Session Border Controller transform E.164 numbers to URIs during the process of routing (or redirecting) a call. During the routing of a SIP call, the Oracle Communications Session Border Controller uses a local policy attribute to determine if an ENUM query is required and if so which ENUM server(s) need to be queried. A successful ENUM query results in a URI that is used to continue routing or redirecting the call.

Configurable Lookup Length

You can configure a lookup length in the ENUM configuration that provides for more efficient caching of URI lookup results; in it, you can specify the length of the string for the DNS request starting from the most significant digit. This provides more flexibility for length matching, which

is useful given the amount of wild card matching available in ENUM services. Specific ENUM groups might only be intended to provide NPANXX or wild card results.

UDP Datagram Support for DNS NAPTR Responses

The Oracle Communications Session Border Controller's default behavior is to conform to the DNS standard defined in RFC 1035 Domain Names: Implementation and Specification, which sets a maximum size for UDP responses of 512 bytes. This limitation means that responses larger than 512 bytes are truncated (set with the TC, or truncation, bit). In addition, this limitation protects network and system resources because using TCP consumes an undesirable amount of both.

However, you can configure support ENUM queries that manage larger UDP DNS responses as set out in RFC 2671, Extension Mechanisms for DNS (EDNS0), enabling your Oracle Communications Session Border Controller to manage responses beyond 512 bytes. According to RFC 2671, senders can advertise their capabilities using a new resource record (OPT pseudo-RR), which contains the UDP payload size the sender can receive. When you specify a maximum response size over 512 bytes, then the Oracle Communications Session Border Controller add the OPT pseudo-RR to the ENUM query—without which the ENUM server will truncate the response.

Custom ENUM Service Type Support

You can configure the ENUM service type that you want to use for an ENUM group. The Oracle Communications Session Border Controller has always supported E2U+sip and sip+E2U by default, and still does. With Release S-C6.1.0, however, you are also able to configure the service type to those supported in RFCs 2916 and 3721.

For example, you can now set the service type in the ENUM configuration to support E2U+sip and E2U+voicemsg:sip. When you configure customer ENUM service types on your system, however, you should note the following:

- New entries in the **service-type** parameter overwrite pre-existing values, including the default values.
- Because of the overwriting noted above, you must include the defaults (if you want them configured) when you are adding additional ENUM service type support. That is, you have to also type in E2U+sip and sip+E2U if you want them to be used in addition to the customized types you are setting.

ENUM Failover and Query Distribution

ENUM Query Distribution

The Oracle Communications Session Border Controller can intelligently distribute ENUM queries among all configured ENUM servers. By setting the enum config's **query method** parameter to round robin, the Oracle Communications Session Border Controller will cycle ENUM queries, sequentially, among all configured ENUM servers. For example, query 1 will be directed to server 1, query 2 will be directed to server 2, query 3 will be directed to server 3, and so on.

The default query method, hunt, directs all ENUM queries toward the first configured ENUM server. If the first server is unreachable, the Oracle Communications Session Border Controller directs all ENUM queries toward the next configured ENUM server, and so on.

Failover to New enum-config

When an enum-config's configured servers are unreachable via the network, i.e., no response is received on a query, the Oracle Communications Session Border Controller can failover to a defined ENUM config that contains different enum servers to query. This failover behavior works when all servers in an enum config are unreachable, rather than when the Oracle Communications Session Border Controller receives not-found type responses.

The Oracle Communications Session Border Controller queries each ENUM server once before trying the next configured server, and then ultimately trying the servers listed in the **failover-to** enum config. If the failover-to servers also are unreachable, the Oracle Communications Session Border Controller fails the call; the failover-to behavior does not recurse among enum-configs, it only checks the first, linked enum-config.

ENUM Server Operation States

After 5 consecutive failed attempts, an ENUM server is considered Out of Service (OOS). All subsequent queries which would be directed to the OOS servers are immediately directed to the first non-OOS server. ENUM servers return to in-service after 600 seconds. If all configured ENUM servers are OOS, the Oracle Communications Session Border Controller fails the call.

After the first failed attempt to reach an ENUM server, it is placed in a Time Out state, which it stays in for 30 seconds. Within this 30 seconds it will not be contacted when an ENUM query is made. After the 30 seconds pass, the ENUM server goes back to an in-service state.

Server Availability Monitoring

The Oracle Communications Session Border Controller can probe an ENUM server's health by sending it a standard ENUM NAPTR query and receiving a valid answer. The query is for the phone number defined in the **health query number** parameter, which should be one that the ENUM servers can positively resolve. As long as the query succeeds, that ENUM server maintains its in-service state and is available for ENUM queries. Any lack of response, whether network based (time-outs), or application based (DNS error or not found response) is considered a query failure and the server is set to OOS and unavailable for ENUM queries.

The Oracle Communications Session Border Controller continuously checks the health of all configured ENUM servers to determine their current state and monitor for failed servers' return to service. All servers are checked for availability at the **health query interval** parameter, as defined in seconds.

> **Note:**
>
> When ENUM server availability monitoring is enabled, ENUM servers can only exist in an in-service or out-of-service states; Without the health query interval defined, server availability monitoring is disabled, and ENUM servers exist in three service states.

ENUM Server IP Address and Port

You can configure an IP address and port for each enum server listed in the enum-servers parameter. IP address and port are specified in XXX.XXX.XXX.XXX:YYYY format with a port value range of 1024-65535. If the port number is not specified, 53 is assumed.

The Oracle Communications Session Border Controller supports IPv6 ENUM configurations in IPv6 realms. The enumservers parameter in the enum-config configuration parameter can be configured IPv6 addresses in addition to IPv4 addresses. When IPv6 Addresses are used, the realm configured in the realm-id parameter must be an IPv6 realm.

Unapplicable SNMP Traps and Objects

When only IPv4 ENUM servers are configured, all legacy SNMP object and trap functionality remains the same. When IPv6 addressing is used for ENUM servers, these existing SNMP objects are obsoleted.

```
apSysMgmtENUMStatusChangeTrap           NOTIFICATION-TYPE
apENUMServerStatusTable OBJECT-TYPE
```

IPv6 ENUM SNMP Traps and Objects

New SNMP trap notifies operators of ENUM Server Status change.

```
apAppsENUMServerStatusChangeTrap           NOTIFICATION-TYPE
        OBJECTS         { apAppsENUMConfigName,
                          apAppsENUMServerInetAddressType,
                          apAppsENUMServerInetAddress,
                          apAppsENUMServerStatus }
        STATUS          current
        DESCRIPTION
        " The trap will be generated if the reachability status of an ENUM
        server changes."
        ::= { apAppsNotifications 1 }
```

The following objects support this trap.

```
apAppsENUMServerStatusTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF ApAppsENUMServerStatusEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "A read-only table to hold the status of configured ENUM servers,
indexed by the name of the enum server, server address type and server IP.
            Please note this table is the replacement of
apENUMServerStatusTable defined in ap-smgmt.mib, where the table was
obsoleted."
        ::= { apAppsMIBTabularObjects 1 }
apAppsENUMServerStatusEntry OBJECT-TYPE
        SYNTAX          ApAppsENUMServerStatusEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "An entry designed to hold the status of a single ENUM server"
        INDEX { apAppsENUMConfigName,
                apAppsENUMServerInetAddressType,
                apAppsENUMServerInetAddress }
        ::= { apAppsENUMServerStatusTable 1 }
ApAppsENUMServerStatusEntry ::= SEQUENCE {
        apAppsENUMConfigName                DisplayString,
```

```
        apAppsENUMServerInetAddressType     InetAddressType,
        apAppsENUMServerInetAddress         InetAddress,
        apAppsENUMServerStatus              INTEGER
        }
apAppsENUMConfigName    OBJECT-TYPE
        SYNTAX          DisplayString
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The name of the enum-config element that contains this
            ENUM server."
        ::= { apAppsENUMServerStatusEntry 1 }
apAppsENUMServerInetAddressType   OBJECT-TYPE
        SYNTAX          InetAddressType
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The IP address of this ENUM server."
        ::= { apAppsENUMServerStatusEntry 2 }
apAppsENUMServerInetAddress   OBJECT-TYPE
        SYNTAX          InetAddress
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The IP address of this ENUM server."
        ::= { apAppsENUMServerStatusEntry 3 }
apAppsENUMServerStatus  OBJECT-TYPE
        SYNTAX          INTEGER {
                                inservice(0),
                                lowerpriority(1),
                                oosunreachable(2)
                                }
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The status of this ENUM server."
        ::= { apAppsENUMServerStatusEntry 4 }
apAppsENUMServerStatusGroup  OBJECT-GROUP
        OBJECTS {
                apAppsENUMConfigName,
                apAppsENUMServerInetAddressType,
                apAppsENUMServerInetAddress,
                apAppsENUMServerStatus
                }
        STATUS          current
        DESCRIPTION
                "Report the status of configured ENUM servers."
        ::= { apAppsObjectGroups 1  }
apAppsEnumServerNotificationsGroup NOTIFICATION-GROUP
      NOTIFICATIONS {
                        apAppsENUMServerStatusChangeTrap
                    }
                STATUS          current
                DESCRIPTION
                "A collection of traps to extend reporting capabilities."
      ::= {  apAppsNotificationGroups 1 }
```

Caching ENUM Responses

As DNS responses often lead to further DNS queries, a DNS server can send additional multiple records in a response to attempt to anticipate the need for additional queries. The Oracle Communications Session Border Controller can locally cache additional NAPRT, SRV, and A records returned from an ENUM query to eliminate the need for unnecessary external DNS requests by enabling the **cache addl records** parameter. These cached records can then be accessed by internal ENUM and DNS agents.

The unprompted NAPTR, SRV, or A record returned to the Oracle Communications Session Border Controller must include complete information to resolve a call to be added to the local DNS/ENUM cache, otherwise the Oracle Communications Session Border Controller will preform an external query to fine the address it is looking to resolve.

Cached entries are per ENUM config. That means if one ENUM config has a number of cached entries, and an ENUM request is directed through a different ENUM config, the second configuration is not privy to what the first configuration has cached.

The Oracle Communications Session Border Controller uses the shorter lifetime of the DNS response's TTL or the server dns attribute's transaction-timeout to determine when to purge a DNS record from the local cache.

Source URI Information in ENUM Requests

ENUM queries can be configured to include the source URI which caused the ENUM request by enabling the **include source info** parameter. The Oracle Communications Session Border Controller can add the P-Asserted-ID URI (only if not in an INVITE) or the From URI into an OPT-RR Additional Record to be sent to the ENUM server. It can be useful to specify the originating SIP or TEL URI from a SIP request which triggered the ENUM query, so the ENUM server can provide a customized response based on the caller.

This feature implements the functionality described in the Internet Draft, DNS Extension for ENUM Source-URI, draft-kaplan-enum-source-uri-00.

When a P-Asserted-ID is blocked or removed before the ENUM query is made, the Oracle Communications Session Border Controller only sends the URI in the From header.

Note that to support this feature, according to the Internet draft, ENUM clients must support 1220 bytes in UDP responses. Therefore, if this feature is enabled, and the max response size parameter is not set i.e., with a 512 byte default, the Oracle Communications Session Border Controller will set the size to 1200 on the OPT-RR records sent.

Operation Modes

There are four modes of ENUM operation that are selected on a global basis:

- stateless proxy
- transaction stateful proxy
- session stateful proxy
- B2BUA with or without media

Stateless Proxy Mode

The stateless proxy mode is the most basic form of SIP operation. The stateless proxy mode:

- Has the least number of messages per call. No record route header is added and there are no 100 Trying or BYEs.
- Does not keep transaction state (timers and retransmission). There are no session counters and no session stop time. No session stop time means no RADIUS STOP records.
- Has no limits on session state.
- Can restrict functionality by specification. This can mean no media management, limited potential for RADIUS accounting, and no CALEA (no Release/BYE messages for CDC).
- Acts primarily as a routing device, with local policy routing and ENUM routing.

Transaction Stateful Proxy

In the transaction stateful proxy mode:

- Adds state to the proxy (not dialogs).
- Has lower number of messages per call. No Record Route header added and no BYES.
- Keeps transaction state (timers and retransmissions.
- Enforces session restrictions (32k) because of state management. These restrictions can be increased.
- Can restrict functionality by specification. This can mean no media management, limited potential for RADIUS accounting, and no CALEA (no Release/BYE message for CDC).
- Acts as routing device with transaction timers, with local policy routing and ENUM routing.
- Can off-load some transactions across unreliable links.

Session Stateful Proxy

The session stateful proxy mode:

- Maintains dialog state as a proxy.
- Includes BYES (though cannot be inserted)
- Keeps transaction state (timers and retransmission)
- Provides per-session information such as session counters per session agent, RADIUS STOP accounting record generation, CALEA CDC generation.
- Enforces session restrictions (32k) because of state management.
- Does not provide media management. There is no CALEA CCC.
- Routes full sessions with transaction timers with local policy routing and ENUM routing.

B2BUA

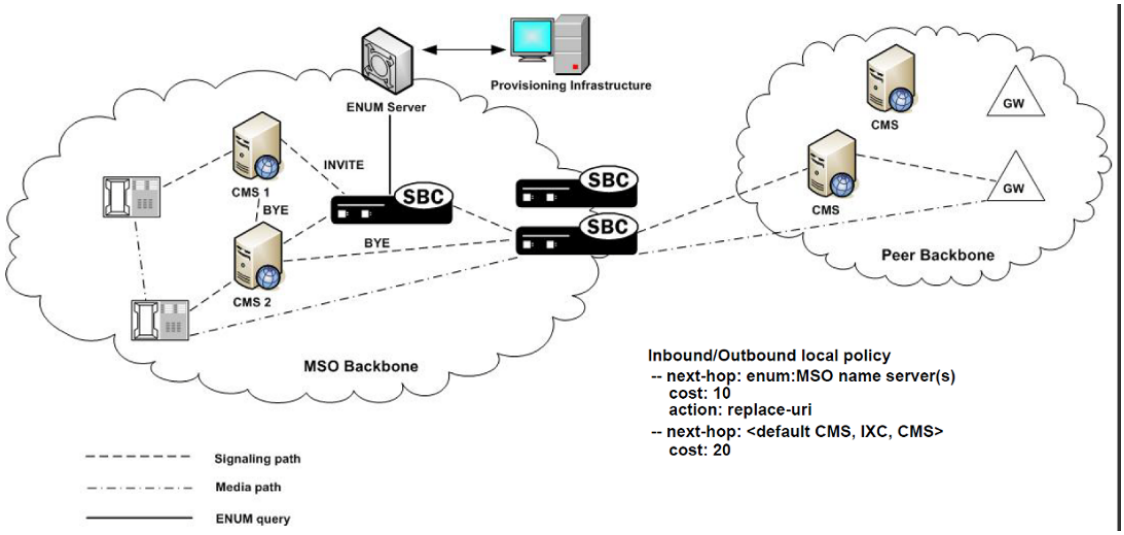
The B2BUA mode:

- Acts as UAS and UAC within call flow.
- Includes BYES (can be inserted).
- Keeps transaction state (timers and retransmissions)
- Provides per-session information such as session counters per session agent, RADIUS STOP accounting record generation, CALEA CDC generation.

- Enforces session restrictions (32k) because of state management.
- Can provide media management, including media routing through a single IP address with topology masking, CALEA CCC, media watchdogs for state management.
- Routes full sessions with topology masking. Includes rewriting Via, Route, Contact headers, full NATing with SIP NAT or header manipulation, direct bridging, local policy routing, and ENUM routing.

Example ENUM Stateless Proxy

The following diagram shows the Oracle Communications Session Border Controller using ENUM to query a local subscriber database. The Oracle Communications Session Border Controller serves as the inbound and outbound routing hub and performs media management. Calls are routed throughout the MSO network using ENUM lookup results.



ENUM Configuration

This section shows you how to configure ENUM on your Oracle Communications Session Border Controller.

> **NOT_SUPPORTED:**
>
> ACMECSR-1660 Hardware based datapath SBC platforms, including 46xx, 61xx and 63xx support a maximum of 4 ENUM servers.

To configure ENUM:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **enum-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# enum-config
ORACLE(enum-config)#
```

4. **name**—Enter a string that uniquely identifies this ENUM configuration. You use this name in other areas of the Oracle Communications Session Border Controller configuration to refer to this ENUM configuration. For example, in the local policy attributes.
5. **top-level-domain**—Enter the domain extension to be used when querying the ENUM servers for this configuration. For example, e164.arpa. The query name is a concatenation of the number and the domain.

   For example the number is +17813334444 and the domain is e164.arpa, the query name would be 4.4.4.4.3.3.3.1.8.7.1.e164.arpa.com.
6. **realm-id**—Enter the realm where the ENUM servers can be reached. The realm ID is used to determine on which network interface to issue the ENUM query.
7. **enum-servers**—Enter the list of ENUM servers (an ENUM server and corresponding redundant servers) to be queried. Separate each server address with a space and enclose list within parentheses.

   The first server on this list is the first one to be queried. If the query times out (including retransmissions) without getting a response, the next server on the list is queried and so on.
8. **service-type**—Enter the ENUM service types you want supported in this ENUM configuration. Possible entries are E2U+sip and sip+E2U (the default), and the types outlines in RFCs 2916 and 3721.

   This parameter defaults to the following service types: E2U+sip and sip+E2U.

   You can enter multiple services types in the same entry, as in this example:

```
ORACLE(enum-config)# service-type E2U+sip,sip+E2U,E2U+voicemsg
```

9. **query-method**—Set the strategy the Oracle Communications Session Border Controller uses to contact ENUM servers. Valid values are:
   - hunt—Directs all ENUM queries toward the first configured ENUM server. If the first server is unreachable, the Oracle Communications Session Border Controller directs all ENUM queries toward the next configured ENUM server, and so on.
   - round-robin—Cycles all ENUM queries, sequentially, among all configured in-service ENUM servers. Query 1 will be directed to server 1, query 2 will be directed to server 2, query 3 will be directed to server 3.
10. **timeout**—Enter the total time in seconds that should elapse before a query sent to a server (and its retransmissions) will timeout. If the first query times out, the next server is queried and the same timeout is applied. This process continues until all the servers in the list have timed out or until one of the servers responds.

    The retransmission of ENUM queries is controlled by three timers. These timers are derived from this timeout value and from underlying logic that the minimum allowed retransmission interval should be 250 milliseconds; and that the Oracle Communications Session Border Controller should retransmit 3 times before timing out to give the server a chance to respond. The valid values are:

- **Init-timer**—Is the initial retransmission interval. If a response to a query is not received within this interval, the query is retransmitted. To safeguard from performance degradation, the minimum value allowed for this timer is 250 milliseconds.
- **Max-timer**—Is the maximum retransmission interval. The interval is doubled after every retransmission. If the resulting retransmission interval is greater than the value of max-timer, it is set to the max-timer value.
- **Expire-timer**—Is the query expiration timer. If a response is not received for a query and its retransmissions within this interval, the server will be considered non-responsive and the next server in the list will be tried.

  The following examples show different timeout values and the corresponding timers derived from them.

  timeout >= 3 seconds

  ```
  Init-timer = Timeout/11
  Max-Timer = 4 * Init-timer
  Expire-Timer = Timeout
  ```

  timeout = 1 second

  ```
  Init-Timer = 250 ms
  Max-Timer = 250 ms
  Expire-Timer = 1 sec
  ```

  timeout = 2 seconds

  ```
  Init-Timer = 250 ms
  Max-Timer = 650 ms
  Expire-Timer = 2sec
  ```

11. **cache-inactivity-timer**—Enter the time interval in seconds after which you want cache entries created by ENUM requests deleted, if inactive for this interval. If the cache entry gets a hit, the timer restarts and the algorithm is continued until the cache entry reaches its actual time to live.

    Setting this value to zero disables caching. For optimal performance, set this to one hour. Rarely used cache entries are purged and frequently used entries are retained. The default value is **3600**. The valid range is:

    - Minimum—0
    - Maximum—999999999

12. **lookup-length**—Specify the length of the ENUM query, starting from the most significant digit. The default is **0**. The valid range is:

    - Minimum—1
    - Maximum—255

13. **max-response-size**—Enter the maximum size in bytes for UDP datagrams in DNS NAPTR responses. This parameter takes values from 512 (default) to 65535. Although the maximum value you can set is 65535, Oracle recommends configuring values that do not exceed 4096 bytes.

14. **health-query-number**—Set this parameter to a standard ENUM NAPTR query that will consistently return a positive response from the ENUM server.

15. **health-query-interval**—Set this parameter to the number of seconds to perpetually probe ENUM servers for health.
16. **failover-to**—Set this parameter to the name of another ENUM-config which to failover to under appropriate conditions.
17. **cache-addl-records**—Set this parameter to **enabled** for the Oracle Communications Session Border Controller to add additional records received in an ENUM query to the local DNS cache.
18. **include-source-info**—Set this parameter to enabled for the Oracle Communications Session Border Controller to send source URI information to the ENUM server with any ENUM queries.
19. Save your work.

Example

The following example shows an ENUM configuration called enumconfig.

```
enum-config
        name                            enumconfig
        top-level-domain
        realm-id                        public
        enum-servers                    10.10.10.10:3456
                                        10.10.10.11
        service-type                    E2U+sip,sip+E2U
        query-method                    hunt
        timeout                         11
        cacheInactivityTimer            3600
        max-response-size               512
        health-query-number             +17813245678
        health-query-interval           0
        failover-to                     enumconfig2
        cache-addl-records              enabled
        include-source-info             disabled
```

Configuring the Local Policy Attribute

You can specify that an ENUM query needs to be done for the routing of SIP calls. You do so by configuring the local policy's next-hop attribute with the name of a specific ENUM configuration, prefixed with the enum: tag. For example: enum:test

You can configure multiple next-hops with different ENUM servers or server groups (possibly with different top-level-domains). If the first ENUM server group you enter as the next hop is not available, one of the others can be used.

> **Note:**
>
> A new parameter called action has replaced the policy attribute's replace-uri parameter available prior to build 211p19.
> To configure local policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. **next-hop**—Enter the name of the ENUM configuration with the prefix enum:. For example, enum:test.
5. **action**—Set to **redirect** if you want to send a REDIRECT message back to the calling party with the information returned by ENUM in the Contact. The calling party then needs to send a REDIRECT using that information. The default value is **none**. Valid values are:
   - **none**—No specific actions requested.
   - **replace-uri**—To replace the next Request-URI with the next hop.
   - **redirect**—To send a redirect response with this next hop as contact.
6. Save and activate your configuration.

Local Policy Example

The following example shows one local policy with the next-hop configured to use enum:test and a second with the next-hope configured to use enum:test_alternate.

```
local-policy
     from-address            *
     to-address              *
     source-realm            public
     activate-time           N/A
     deactivate-time         N/A
     state                   enabled
     last-modified-date      2006-03-09 09:18:43
     policy-attribute
       next-hop                enum:test
       realm                   public
       action                  none
       terminate-recursion     disabled
       carrier
       start-time              0000
       end-time                2400
       days-of-week            U-S
       cost                    1
       app-protocol            SIP
       state                   enabled
       media-profiles
     policy-attribute
       next-hop                enum:test_alternate
       realm                   public
```

```
action                  none
terminate-recursion     disabled
carrier
start-time              0000
end-time                2400
days-of-week            U-S
cost                    2
app-protocol            SIP
state                   enabled
```

CNAM Subtype Support for ENUM Queries

CNAM, calling name, data is a string up to 15 ASCII characters of information associated with a specific calling party name. The *Internet-draft, draft-ietf-enum-cnam-08.txt*, registers the Enumservice 'pstndata' and subtype 'cnam' using the URI scheme 'pstndata:' to specify the return of CNAM data in ENUM responses. The Oracle Communications Session Border Controller recognizes CNAM data returned via this mechanism. CNAM data is then inserted into the display name of the From: header in the original Request. If a P-Asserted-ID header is present in the original request, the CNAM data is inserted there as well.

CNAM data is identified by an ENUM response with service-type: E2U+pstndata:cnam

CNAM support is configured in the sip profile configuration element, which can then be applied to either a session agent, realm, or SIP interface.

The Oracle Communications Session Border Controller can preform CNAM queries on the signaling message's ingress or egress from the system by setting the cnam lookup direction parameter to either ingress or egress. If the CNAM lookup direction parameters are configured on both the ingress and egress sides of a call, the Oracle Communications Session Border Controller will only preform the lookup on the ingress side of the call.

CNAM Unavailable Response

A CNAM response can include a Calling Name Privacy Indicator parameter ('unavailable=p') or Calling Name Status Indicator parameter ('unavailable=u') in responses. The Oracle Communications Session Border Controller can insert a custom reason string into the SIP message's From and P-Asserted-ID header in the original requires.

Configuring the **cnam unavailable ptype** parameter inserts the specified text into the From and P-Asserted-ID headers when a CNAM response contains the unavailable=p parameter.

Configuring the **cnam unavailable utype** parameter inserts the specified text into the From and P-Asserted-ID headers when a CNAM response contains the unavailable=u parameter.

SIP Profile Inheritance

CNAM features, via the SIP Profile configuration element can be applied to session agents, realms, and SIP interfaces. The more generalized object inherits the more specific object's values. For example, if CNAM support via a SIP profile is configured on a session agent, the expected processing will override any SIP profile configuration on the downstream realm or SIP interface. Likewise, if CNAM support is unconfigured on the receiving session agent, but configured in the realm, CNAM configuration on the SIP interface will be ignored.

CNAM Subtype Support Configuration

To enable the Oracle Communications Session Border Controller to preform CNAM subtype ENUM queries, you must configure a SIP profile with an enum-config object (that points to valid ENUM servers). The referenced enum-config configuration element lists the servers to contact for CNAM type queries (and other general ENUM server interaction parameters).

To configure CNAM subtype support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-profile
   ORACLE(sip-profile)#
   ```

4. **name**—Enter a string that uniquely identifies this SIP profile configuration. You use this name in other areas of the Oracle Communications Session Border Controller configuration to refer to this SIP profile in session agents, realms, or SIP interfaces.
5. **cnam-lookup-server**—Set this parameter to the name of an ENUM-config to that will query ENUM servers for CNAM data.
6. **cnam-lookup-dir**—Set this parameter to **ingress** or **egress** to identify where the Oracle Communications Session Border Controller performs a CNAM lookup with respect to where the call traverses the system. The default value is **egress**.
7. **cnam-unavailable-ptype**—Set this parameter to a string, no more than 15 characters, to indicate that the unavailable=p parameter was returned in a CNAM response.
8. **cnam-unavailable-utype**—Set this parameter to a string, no more than 15 characters, to indicate that the unavailable=u parameter was returned in a CNAM response.
9. Save your work.

Using the Local Route Table (LRT) for Routing

The LRT allows the Oracle Communications Session Border Controller to determine next hops and map E.164 to SIP URIs locally for routing flexibility.

The LRT uses a local route cache that is populated by a local XML file on the Oracle Communications Session Border Controller. Each local cache is populated from one defined XML file. For routing, the local route cache operates in a way similar to the ENUM model where a local policy next hop specifies the local route table that the Oracle Communications Session Border Controller references. For example, you can configure one next hop to use one table, and another next hop to use a different table.

Similar to the ENUM model, the Oracle Communications Session Border Controller typically performs a local route table lookup using the telephone number (TN) of the SIP Request-URI.

This is the user portion of the URI, and the Oracle Communications Session Border Controller ignores user parameters or non-digit characters. The local route table XML file defines the matching number and the resulting regular expression replacement value such as ENUM NAPTR entries do. The Oracle Communications Session Border Controller uses the resulting regular expression to replace the Request-URI, and it uses the hostname or IP address portion to determine the next hop. If the hostname or IP address matches a configured session agent, the request is sent to that session agent. If the Oracle Communications Session Border Controller does not find a matching session agent for the hostname/IP address, the Oracle Communications Session Border Controller either performs a DNS query on the hostname to determine its IP address or sends the request directly to the IP address.

> **Note:**
>
> RFC3261 explicitly excludes the use of the _ [underscore] character in a URI. Do not configure LRT entries with a URI that includes an underscore character.

When the next hop is defined as a user-parameter lookup key, such as a routing number (RN) or carrier identification code (CIC), the defined key is used for the local route table lookup.

The Oracle Communications Session Border Controller can attempt up to 10 next hops per LRT entry in the order in which they appear in the XML file. If the next hop is unsuccessful, the Oracle Communications Session Border Controller tires the next hop on list. An unsuccessful hop may occur when an out-of-service session agent or the next hop responds with a failure response.

> **Note:**
>
> Entering XML comments on the same line as LRT XML data is not supported.

The Oracle Communications Session Border Controller can perform local route table lookups for SIP requests and communicate the results to the SIP task. The new task processes the new local routing configuration objects.

When a SIP call is routed, the Oracle Communications Session Border Controller uses local policy attributes to determine if a local route table lookup is required. If a lookup is needed, the Oracle Communications Session Border Controller selects the local routing configuration to use. Successful local route table lookups result in URIs that can be used to continue routing and redirecting calls.

Local Route Table (LRT) Performance

Capabilities

- Loads approximately 500 LRT tables during boot time
- Loads 100,000 entries per LRT file
- Loads 2,000,000 LRT entries total per system

Constraints

- You cannot configure the Oracle Communications Session Border Controller with 500 LRT files each with 100,000 entries.

- Actual performance that affects the interaction among the three performance attributes varies with system memory and configuration.

Multi-Tiered LRT Route Selection

When routing though an LRT, the SBC normally attempts to reach next-hops using LRT entries in the order that they appear in the XML file. If a next-hop is unsuccessful, the SBC tries the next-hop on the list. You can, however, configure entries in LRTs that cause the SBC to gradually increase traffic for specific routes and control the distribution, while also monitoring usage. You can specify priorities and weights to favor route entries and use a preferred route instead of following the list order.

To establish the use of priority and weight, use the route tag syntax **<route format="weighted">** for a route entry in the LRT. Each next-hop entry can then include the following fields:

- **prio**—Specifies the relative preference in which the SBC uses this next-hop entry. You configure priority with zero as the highest priority, using the range from 0 to 65535.
- **weight**—Specifies the frequency of use for a given next-hop entry in a set of next-hop entries with the same priority; an entry with a higher weight is returned more frequently. You configure weight in multiples of 10 using the range of 0 to 65535.

A route set in an LRT file, without a specified priority or weight, uses the route tag syntax **<route>**. The SBC selects next-hop entries for these routes using the list order only. Using the <route format="weighted"> tag syntax allows you to have active routes with specified priority and weight in the same table as routes that do not.

When the route tag contains the format="weighted" attribute, the maximum number of next-hops allowed per route with the same priority value is 10. There is no next-hop limit when there is no format attribute specified.

The SBC treats any next-hop entry with weight=”0” as disabled. The SBC does not use these entries for routing and does not display them in **show lrt** commands. The SBC also treats invalid route-sets, which do not have any valid next-hop entries under a particular route-entry, as if they are disabled.

Operational Examples

The following table displays one route set using format, priority, and weight, and in contrast, one route set that uses none.

```
<?xml  version="1.0" encoding="UTF-8"?>
<localRoutes>
<route format=”weighted”>
      <user type="E164">370</user>
      <next prio="0" weight="40" type="regex">!^.*$!sip:\0@SAG-CarrierA!</
next>
      <next prio="0" weight="30" type="regex">!^.*$!sip:\0@SAG-CarrierB!</
next>
      <next prio="0" weight="20" type="regex">!^.*$!sip:\0@SAG-CarrierC!</
next>
      <next prio="1" weight="10" type="regex">!^.*$!sip:\0@SAG-CarrierD!</
next>
      <next prio="2" weight="10" type="regex">!^.*$!sip:\0@SAG-CarrierE!</
next>
</route>
<route>
```

```
      <user type="E164">371</user>
      <next type="regex">!^.*$!sip:\0@SAG-NoPrio1</next>
      <next type="regex">!^.*$!sip:\0@SAG-NoPrio2</next>
</route>
</localRoutes>
```

Consider the next hops with priority 0 and weights of 40, 30 and 20. In this case, the SBC distributes calls between these 3 entries using a ratio of 4:3:2. This means that, for 10 calls to example user “370”, the SBC distributes the first 9 calls using a 4:3:2 ratio:

- 4 calls will be routed to SAG-CarrierA
- 3 calls to SAG-CarrierB
- 2 calls to SAG-CarrierC

The SBC attempts to reach next-hops with higher prio values (lower priority, such as prio=1) in this example only if the next-hop with lower prio values (higher priority), such as prio=0, is in Out of service. Note that the weighted algorithm distributes calls randomly in the given ratios.

Consider the above example ,whenever a call attempts to route to SAG-CarrierA of prio=0, and if it is Out of Service,the call routes to SAG-CarrierD, which is from next priority prio=1.The same is the case for SAG-CarrierB, SAG-CarrierC from prio=0, if they are in Out of service, it routes the next priority prio=1.

The SBC ignores the weight value if a given next-hop is the only entry with that priority in a route set. In the example , the SBC ignores the weight values for the **prio="1"** and **prio="2"** entries as they are the only entries with those priority values. Furthermore:

1. If we need 4 next-hops in ratio of 2:3:5:7, then we can configure the weight values as either:
   - 20,30,50,70
   - 200,300,500,700, or
   - 2000,3000,5000,7000
2. If we configure weights as 10, 100, 1000, 10000, the SBC routes calls using the ratio 1:10:100:1000.

Configuration Dependencies

Recursion allows the call to break out of a SAG route and continue to the next route in the LRT. Continuing with the example above, assume the following:

- SAG-CarrierA contains 2 SAs, SA_M1 and SA_M2 and has sag-recursion=disabled
- SAG-CarrierB contains 2 SAs, SA_B1 and SA_B2 and has sag-recursion=enabled
- SAG-CarrierC contains 2 SAs, SA_T1 and SA_T2 and has sag-recursion=enabled

The route proceeds as SA_M1 overflows to SA_B1 on to SA_B2, and then on to SA_T1 and SA_T2. Notice SA_M2 was not attempted because sag recursion was disabled within that route.

If you have enabled **sag-recursion** for SAGs configured under a single route-entry, and each SAG has multiple SAs, the SBC tries to reach every SA in every SAG until it gets a response for its request. However, you can skip recursion between multiple SAs under one SAG by either configuring **stop-sag-recurse** with valid response values or by configuring a **sip-recursion-policy** on the SAG.

- If you configure **stop-sag-recurse** with valid response codes and the condition gets a hit, then the SBC terminates the call by sending the response code to UAC. In this case, the SBC does not try the next SAGs in the list.
- If you want the SBC to recurse through multiple SAG's in the list, even after receiving error response codes from one of the SA in SAG, you can configure a **sip-recursion-policy** at two levels; one at the SAG level and other at the **sip-interface** level.

> **Note:**
>
> SIP recursion between different route entries in LRT file works in same way with SAGs and SAs. This feature does not affect any SIP recursion behavior.

Creating XML Files

Refer to the following guidelines as you set up your multi-tiered LRT deployment:

- Format Attribute (XML file):
  - Specifying the wrong value to format attribute, format="weids" for example , causes the SBC to treat the entire route as invalid, not load any of the route entries under this route, and not display them in the **show lrt stats** output. However, by specifying the format attribute name incorrectly, form="weighted" for example, you can ensure the SBC treats routes as "route without format attribute", expecting next-hop entries to be present without any prio and weight attributes. If any <next> entries under this route have prio, weight or both, then the SBC treats the <next> entry as invalid. If all of the <next> entries under the <route> are invalid, then the <route> is invalid.
  - The format attribute in the xml file under route entry specifies that all the next-hop entries under this route entry should have both priority and weight fields defined. Any next-hop entry without both is considered invalid.
  - The SBC allows configuration of partial values for the format attribute. All partial strings matching the original string ("weighted") are treated as valid. Valid examples include "w", "we", "wei"… "weighted".
  - The SBC default behavior supports XML files without the format="weighted" attribute in the route tag. The SBC considers any route entry without format attribute and with next-hop entry contains either prio, weight, as invalid entries.
- Prio and Weight Attributes:
  - The SBC requires the Prio and weight attributes to contain numerical values, treating entries with alphanumeric, empty values or any other values are treated as invalid. The SBC refers to the "format" attribute, using the value "weighted" to differentiate between weighted entries and normal entries.
  - You can specify the prio, weight and type attributes in any order in a next-hop.
  - The maximum number of next-hop entries used under one route set with same priority value is 10, when the attribute format="weighted" is in the route tag. The SBC considers only the first 10 <next> entries as valid. In addition, routes with more than 10 <next> entries are valid, with entries after the tenth being ignored.

Monitoring Multi-Tiered LRT Operation

You can monitor LRT activity using the **show lrt** command to display priority and weight fields if they are part of the route entry output. The SBC always displays weight values in the output of **show lrt** commands. Applicable syntax includes:

- **show lrt route-table <lrt-config-name>**
- **show lrt route-entry <lrt-config-name> <user>**

The examples below assume an LRT table named LRT1 that contains one route and one user named 370.

```
<?xml version="1.0" encoding="UTF-8"?>
<localRoutes>
<route format="weighted">
      <user type="E164">370</user>
      <next prio="0" weight="40" type="regex">!^.*$!sip:\0@SA1!</next>
      <next prio="1" weight="10" type="regex">!^.*$!sip:\0@SA2!</next>
</route>
</localRoutes>
```

Example output when using the route-table argument is shown below.

```
ORACLESBC# show lrt route-table LRT1
UserName <370>
    Entry Type = E164
      Priority = 0
        Weight = 40
       NextHop = !^.*$!sip:\0@SA1!
  NextHop Type = regexp
      Priority = 1
        Weight = 10
       NextHop = !^.*$!sip:\0@SA2!
  NextHop Type = regexp

---------------------------------
Total: 1 routes
---------------------------------
```

When you add the user argument, the SBC changes the output as shown below.

```
ORACLESBC# show lrt route-entry LRT1 370
UserName <370>
    Entry Type = E164
      Priority = 0
        Weight = 40
       NextHop = !^.*$!sip:\0@SA1!
  NextHop Type = regexp
      Priority = 1
        Weight = 10
       NextHop = !^.*$!sip:\0@SA2!
  NextHop Type = regexp
```

Local Routing Configuration

This section shows you how to:

- Set up local route configuration

- Specify that a set of local policy attributes needs to use local routing

Configure Local Routing

The local routing configuration is an element in the ACLI session-router path, where you configure a name for the local route table, the filename of the database corresponding to this table, and the prefix length (significant digits/bits) to be used for lookup.

To configure local routing:

1. In Superuser mode, type **configure terminal**, and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router**, and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config,**and press Enter.

   ```
   ORACLE(session-router)# local-routing-config
   ORACLE(local-routing-config)#
   ```

4. **name**—Enter the name (a unique identifier) for the local route table; this name is used for reference in the local policy attributes when to specify that local routing should be used. There is no default for this parameter, and it is required.
5. **file-name**—Enter the name for the file from which the database corresponding to this local route table will be created. You should use the .gz format, and the file should be placed in the /code/lrt/ directory. There is no default for this parameter and it is required.
6. **prefix-length**—Enter the number of significant digits/bits to used for lookup and cache storage. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—999999999
7. Save and activate your configuration.

   The following example displays a typical local routing configuration.

   ```
   local-routing-config
           name                            lookup
           file-name                       abc.xml.gz
           prefix-length                   3
   ```

Applying the Local Routing Configuration

Apply the local routing configuration by calling it to use in the local policy attributes. You do this by setting a flag in the **next-hop** parameter along with the name of the local routing configuration that you want to use.

To apply the local routing configuration:

1. In Superuser mode, type **configure terminal**, and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router**, and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **local-policy,** and press Enter.

```
ORACLE(session-router)# local-policy
ORACLE(local-policy)#
```

4. Type **policy-attributes,** and press Enter.

```
ORACLE(local-policy)# policy-attributes
ORACLE(local-policy-attributes)#
```

5. **next-hop**—In the **next-hop** parameter, type in **lrt:** followed directly by the name of the local routing configuration to be used. The **lrt:** tag tells the Oracle Communications Session Border Controller that a local route table will be used.

```
ACMEPACKET(local-policy-attributes)# next-hop lrt:lookup
```

6. Save and activate the configuration.

Multiple Contact Handling in Redirect Action for LRT

When performing a redirect action triggered by local policy lookups, the Oracle Communications Session Border Controller (SBC) typically issues a 305 (Use Proxy) message with a single contact derived from the local policy. In some cases, however, it is preferred to issue a 300 (Multiple Choices) message and provide multiple contacts, providing the endpoint with, for example, fallback contacts. For these scenarios, you can configure the SBC with a **sip-interface** option that supercedes the lookup configuration's compliance with the RFC 3261 standard for issuing a proxy, and respond based on the number of local policy contacts.

Applicable scenarios include the SBC receiving an INVITE that triggers a lookup within a local-policy that has its **policy-attribute**, **action** parameter set to **redirect**. By default, the SBC replies with a single contact inside a 305. Enable the **redirect300ForMultipleContacts** option to have the SBC refer to the local-policy and send multiple contacts inside a 300 message if that local policy has multiple contacts. If the policy has a single contact, the SBC sends the 305.

To perform the desired behavior, configure the applicable **sip-interface** using the following **redirect300ForMultipleContacts** option syntax.

```
ORACLE(sip-interface)# options +redirect300ForMultipleContacts
```

If you type the option without the "plus" sign, you overwrite any previously configured options. To append the option to the **sip-interface** configuration's options list, prepend the option syntax with a "plus" sign, as shown in the previous example.

Save and activate your configuration.

Consider the configuration prior to deployment as it generates a behavior change for all applicable triggers on this **sip-interface**. Alternatively, you could set this behavior on a **sip-interface** implemented for this purpose.

Local Route Table Support for H.323 and IWF

Local Route Table (LRT) support for H.323 and IWF is compatible with that currently offered for SIP. LRT and ENUM provide the Oracle Communications Session Border Controller with the ability to perform routing based on ENUM queries to a DNS server or local to an onboard database.

For the LRT feature, this means that entries in the local routing table now include those prefixed with the h323: URI scheme, indicating that H.323 is the next hop protocol.

IWF Considerations

When the system performs a local policy lookup for an incoming SIP or H.323 call and determines an ENUM/LRT server is the next hop, it queries that ENUM/LRT server. The response will include the URI scheme, indicating the next hop protocol and the hostname/IP address representing the next hop. For cases where the incoming call signaling protocol and the URI scheme of the ENUM/LRT response are the same, the call requires no interworking. The Oracle Communications Session Border Controller can simply route the egress call leg to the specified next hop.

Interworking is required when the incoming signaling protocol and the URI scheme of the ENUM/LRT response do not match. When the responses do not match, the Oracle Communications Session Border Controller interworks between SIP and H.323 to route the call to the appropriate next hop.

The Oracle Communications Session Border Controller also compares the URI scheme returned in the ENUM/LRT response to the application protocol specified in the policy attributes. If the URI scheme is SIP, but the policy attributes indicate H.323, the route is deemed invalid. The same is true for an H.323 URI scheme and SIP route.

ENUM LRT Responses

No special configuration is required for LRT to work for H.323 and IWF calls. You can configure the system to match ENUM/LRT responses against session agent groups, and then use those SAGs for routing.

To enable matching ENUM/LRT responses for H.323 SAG routing:

1. In Superuser mode, type **configure terminal**, and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router,** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **h323-config,** and press Enter.

```
ORACLE(session-router)# h323-config
ORACLE(h323-config)#
```

4. **enum-sag-match**—Set this parameter to enabled if you want the Oracle Communications Session Border Controller to perform matching against the hostnames in ENUM/LRT

lookup responses and session agent groups. If there is a match, the Oracle Communications Session Border Controller uses the matching SAG for routing. If no match is found, normal ENUM/LRT routing proceeds.

LRT Entry Matching

When searching an LRT for a matching route, the Oracle Communications Session Border Controller can be configured with one of three match modes with the match mode parameter in the local routing config. These modes are:

- exact—When searching the applicable LRT, the search and table keys must be an exact match.
- best—The longest matching table key in the LRT is the chosen match.
- all—The all mode makes partial matches where the table's key value is a prefix of the lookup key. For example, a lookup in the following table with a key of 123456 returns entries 1, 3, and 4. The 'all' mode incurs a performance penalty because it performs multiple searches of the table with continually shortened lookup keys to find all matching entries. This mode also returns any exact matches too.

Entry#	Key	Result
1	1	<sip:\0@host1.example.com>
2	122	<sip:\0@host22.example.com>
3	123	<sip:\0@host3.example.com>
4	1234	<sip:\0@host4.example.com>
5	1234567	<sip:\0@host7.example.com>
6	1235	<sip:\0@host5.example.com>

LRT Entry Matching Configuration

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **local-routing-config** and press Enter.

```
ORACLE(session-router)# local-routing-config
ORACLE(local-routing-config)#
```

4. **match-mode**—Set this parameter to either **best**, **all**, or leave it as **exact** which is the default. This indicates to the Oracle Communications Session Border Controller how to determine LRT lookup matches.
5. Save your work using the **done** command.

LRT String Lookup

The Oracle Communications Session Border Controller can search an LRT for either E.164 or string table keys. This selection is on a global basis. When the string-lookup parameter is

disabled (default) in the local routing configuration, all lookups with be E.164 type, except when:

- If eloc-str-lkup is enabled in a matching local policy's policy-attribute, E-CSCF procedures are applied and the resulting lookup type is 'string'.
- The Oracle Communications Session Border Controller also performs string lookups exclusively when a compound lookup key is specified.

When the lookup type is 'E.164', the lookup is skipped if the lookup key is not a valid telephone number (i.e. it must contain only digits).

LRT String Lookup Configuration

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# local-routing-config
   ORACLE(local-routing-config)#
   ```

4. **string-lookup**—Set this parameter to **enabled** for the Oracle Communications Session Border Controller to perform LRT lookups on table keys of a string data type. Leave this parameter to its default as disabled to continue using E.164 type lookups.
5. Save your work using the **done** command.

Directed Egress Realm from LRT ENUM

A message can be sent into a specific egress realm identified in an ENUM query or LRT lookup. The egress realm is noted by a configurable parameter in the result URI. The Oracle Communications Session Border Controller is configured with the name of this parameter, that indicates an egress realm name, and looks for it in the returned URI.

To configure the parameter name, the egress-realm-param option is added to the sip config and/or the h323 config using the following format:

```
egress-realm-param=<name>
```

Where <name> is the parameter name to extract the egress realm name from.

When the egress realm param is defined, the ENUM and LRT results will always be checked for the presence of the URI parameter. The sip config options will apply for received SIP requests. The h323 config option will apply for received H.323 messages.

For example, if egress-realm-param=egress is added to the sip config, a matching entry in the LRT that specifies the egress realm core will look like this:

```
<route>
<user type="E164">+17815551212</user>
```

```
<next type="regex">!^.*$!sip:\0@core.example.com;egress=core!</next>
</route>
```

If the URI does not contain the parameter or the parameter identifies a realm that is not configured on the system, the egress realm that is normally applicable (from local policy, SIP-NAT, or session-agent data) will be used.

Directed Egress Realm Configuration

To add an egress parameter to look for in a sip-config:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. If you are adding this feature to a pre-existing SIP configuration, you will need to select and edit it.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **egress-realm-param**—Configure this option with the parameter to parse for in a returned ENUM or LRT result: For example

   - **options egress-realm-param=egress**

     In order to append the new option to the sip-config's options list, you must prepend the new option with a plus sign. For example:

     ```
     ORACLE(sip-config)# options +egress-realm-param=egress
     ```

5. Save your work using the ACLI **done** command.

> **Note:**
>
> The egress-realm-param option can be configured similarly in the h323-config.

SIP Embedded Route Header

The Oracle Communications Session Border Controller examines the ENUM and LRT lookup result for embedded Route headers. In the LRT or as returned in an ENUM query a URI including an embedded route header would look like:

```
<sip:user@example.com?Route=%3Csip:host.example.com;lr%3E>
```

Using embedded Route headers is the Oracle Communications Session Border Controller's default behavior. This can be overridden by adding the sip-config option use-embedded-route.

When the ENUM or LRT result becomes the top Route header, any embedded Route headers extracted are inserted just after that top Route (which will always be a loose route and include the "lr" URI parameter). In this case, the request will be sent to the top Route.

When the ENUM or LRT results become the Request-URI, any embedded Route headers extracted from the result are inserted before any Route headers already in the outgoing request. After that, if there are any Route headers in the outgoing request and the top Route header has an "lr" URI parameter, the request is sent to the top Route header. Otherwise, the request is sent to the Request URI.

SIP Embedded Route Header Configuration

To set the Oracle Communications Session Border Controller's default behavior of using embedded route headers from ENUM queries or LRT lookups:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter. If you are adding this feature to a pre-existing SIP configuration, you will need to select and edit it.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **use-embedded-route**—Configure this as an option with one of the following arguments:
   - **all** = use embedded routes from both ENUM and LRT results (default)
   - **none** = do not use embedded routes
   - **enum** = use embedded routes from ENUM results only
   - **lrt** = use embedded routes from LRT results only

   In order to append the new option to the sip-config's options list, you must prepend the new option with a plus sign. For example:

   Set the options parameter by typing **options**, a Space, the option name **use-embedded-route**, and then press Enter.

```
ORACLE(sip-config)# options +use-embedded-route=none
```

5. Save your work using the ACLI **done** command.

LRT Lookup Key Creation

This section describes the Oracle Communications Session Border Controller's LRT lookup key creation capability.

Arbitrary LRT Lookup Key

In addtion to the standard From, To, and and P-Asserted-Identity header fields the Oracle Communications Session Border Controller can now use the values from any arbitrary SIP header as an LRT or ENUM lookup key. This is preformed by prepending a dollar sign $ by the header name whose value's userinfo portion of the URI will be used as the lookup key. For example, key=$Refer-To would use the userinfo portion of the URI in the Refer-To header of the request as the lookup key.

An ampersand & followed by a header name will use the whole value of the header as the lookup key. For example, key=&X-Route-Key would use the whole value of the X-Route-Key as the lookup key. As a shortcut, an ampersand is not required for a "hidden" header. For example, "key=@LRT-Key" would use the value of the @LRT-Key header as the lookup key.

Hidden Headers for HMR and LRT lookup

When an LRT lookup key is more complex than just the URI's userinfo or a Tel-URI, HMR can be used to extract the data and build a special header.

By using a header name that begins with the at-sign "@" (e.g. @lrt-key), the header can be hidden and not included in outgoing SIP message, thus eliminating the need for an extra HMR rule to remove it.

Since '@' is not a valid character in a header name as defined by RFC 3261, there is no possibility of a collision between a header name defined in the future and a hidden header name beginning with @.

Compound Key LRT Lookup

LRT lookup keys can be combinations of more than one key value. For example, "key=$FROM,$TO" would construct a compound key with the userinfo of the From URI followed by a comma flowed by the userinfo of the To URI.

If the request message contained:

```
From: <sip:1234@example.com>
To: <sip:5678@example.com>
```

The compound key to match this From/To pair is "1234,5678".

In the table lookup, the compound key is a single key value and there is no special treatment of the comma in key matching. The comma is simply an ordinary additional character that is matched like any letter or digit (i.e. the comma must appear in the LRT entry's "type" element data). For example, if the table were configured for "best" match-mode, the lookup key "1234,5678" would match a table entry of "1234,567", but it would not match a table entry of "123,5678".

Retargeting LRT ENUM-based Requests

Request re-targeting is when a target or a request as indicated in the Request-URI, is replaced with a new URI.

The happens most commonly when the "home" proxy of the target user replaces the Request-URI with the registered contact of that user. For example, the original request is targeted at the

Address-of-Record of bob (e.g. sip: bob@example.net). The "home" proxy for the domain of the original target, example.net, accesses the location service/registration database to determine the registered contact(s) for the user (e.g. sip:bob@192.168.0.10). This contact was retrieved in a REGISTER request from the user's UA. The incoming request is then re-targeted to the registered contact. When retarget-requests is **enabled**, or the original Request-URI is the Oracle Communications Session Border Controller itself, the URI from the LRT lookup is used as the new Request-URI for the outgoing request.

When a request is routed rather than re-targeted, the Request-URI is not changed, but one or more Route headers may be inserted into the outgoing request. Sometimes a request which already contains Route headers will be routed without adding additional Route headers.

When the Oracle Communications Session Border Controller routes requests and the original Request-URI was not the Oracle Communications Session Border Controller itself, the URI from the LRT /ENUM lookup is added as the top Route: header including the "lr" parameter. The Request-URI then remains unchanged.

Whether the Oracle Communications Session Border Controller re-targets or routes a request depends on the following:

- The target (Request-URI) of the received request
- The presence of Route headers
- Local Policy Attributes,
- Registration Cache matching.

If the original target is the Oracle Communications Session Border Controller itself (i.e. the Request-URI contains the IP Address in the SIP interface the request was received on), the request is always retargeted. When the original target is not the Oracle Communications Session Border Controller and Local Policy is applied, the request will be re-targeted when the policy attribute action parameter is **replace-uri**. The request will also be re-targeted when the policy attribute specifies an ENUM or LRT lookup.

Retargetting requests can be configured in either the ENUM or LRT config depending on the request URI retrieval method choosen.

Re-targeting LRT ENUM-based Requests Configuration

This section shows you how to configure the Oracle Communications Session Border Controller to retarget/re-route request message when performing an LRT or an ENUM lookup.

To configure the Oracle Communications Session Border Controller to retarget or route request messages when performing an LRT lookup:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# local-routing-config
   ORACLE(local-routing-config)#
   ```

4. **retarget-requests**—Leave this parameter set to **enabled** for the Oracle Communications Session Border Controller to replace the Request-URI in the outgoing request. Change this parameter to **disabled** for the Oracle Communications Session Border Controller to route the request by looking to the Route header to determine where to send the message.
5. Save your work using the **done** command.

   To configure the Oracle Communications Session Border Controller to retarget or route request messages when performing an ENUM lookup:
6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```
7. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```
8. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# enum-config
   ORACLE(enum-config)#
   ```
9. **retarget-requests**—Leave this parameter set to **enabled** for the Oracle Communications Session Border Controller to replace the Request-URI in the outgoing request. Change this parameter to **disabled** for the Oracle Communications Session Border Controller to route the request by looking to the Route header to determine where to send the message.
10. Save your work using the **done** command.

Recursive ENUM Queries

If the Oracle Communications Session Border Controller receives an A-record in response to an ENUM query, it will reperform that ENUM query to the server received in the A-record.

If the Oracle Communications Session Border Controller receives an NS record in response to an ENUM query, it will resend the original ENUM query to the DNS server defined in the realm of the FQDN in the NS record. It will use the response to perform a subsequent ENUM query.

This behavior is configured by setting the **recursive query** parameter in the enum config to **enabled**.

Recursive ENUM Queries Configuration

To configure the Oracle Communications Session Border Controller to query a DNS server for a hostname returned in an ENUM lookup result:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```
2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config** and press Enter.

```
ORACLE(session-router)# enum-config
ORACLE(enum-config)#
```

4. **recursive-query**—Set this parameter to **enabled** for the Oracle Communications Session Border Controller to query a DNS server for a hostname returned in an ENUM result.
5. Save your work using the **done** command.

Multistage Local Policy Routing

Multistage local policy routing enables the Oracle Communications Session Border Controller to perform multiple stages of route lookups where the result from one stage is used as the lookup key for the next routing stage.

Routing Stages

A routing stage signifies a re-evaluation of local policy based on the results of a local policy lookup. In the simplest, single stage case, the Oracle Communications Session Border Controller performs a local policy lookup on a SIP message's Request URI. The result of that local policy lookup is a next hop FQDN, IP address, ENUM lookup, or LRT lookup; that result is where the Oracle Communications Session Border Controller forwards the message. In the multistage routing model, that resultant next hop is used as the lookup key for a second local policy lookup.

The results of each stage do not erase the results of the previous stage. Thus, previous results are also possible routes to use for recursion, but the next stage results are tried first.

> **Note:**
>
> Setting a next hop to a SAG in a multistage scenario constitutes an error.

Multi-stage Routing Source Realm

By default, the Oracle Communications Session Border Controller uses the realm within which a message was received as the source realm through all stages of a multistage local policy routing lookup. You can change this by setting the **multi-stage-src-realm-override** parameter in the session router config to enabled. Enabling this setting causes the Oracle Communications Session Border Controller to use the next-hop realm from the current local policy stage as the source realm for the next stage of the lookup. This source realm selection process also repeats for each stage of a multistage routing scenario.

Network Applications

The following are typical applications of multistage routing:

- An operator might need to query an ENUM server for a destination number. Based on the NAPTR result of the ENUM query, the Oracle Communications Session Border Controller performs a local policy lookup to decide how to route the request, perhaps based on a LRT table lookup.
- An operator might need to query one ENUM server for a number portability lookup, then based on the routing number perform a second ENUM query to a different server to learn

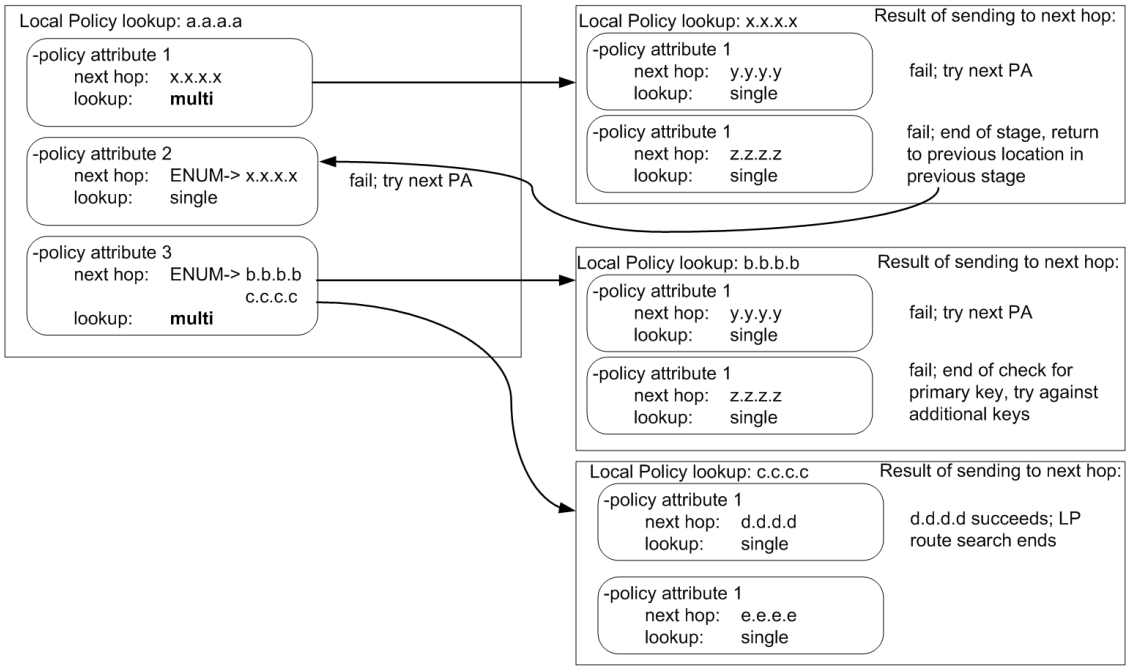
which carrier to use for the routing number. Then, then based on the identified carrier perform a LRT lookup for what next-hop(s) to use for that carrier.

- An operator might query an LRT table to confirm the allowed source number. Then, based on the result, query an ENUM server for destination routing.

Multistage Routing Conceptual Example

Multistage routing is enabled by setting a policy attribute's lookup parameter to multi. Instead of replacing the SIP message's request URI with the policy attribute's next hop address or response from an ENUM or LRT lookup, the system uses that next hop or ENUM or LRT lookup response to reconstruct the SIP message. The reconstructed SIP message is fed again through all configured local policy configuration elements (and policy attribute sub elements). Each time the Oracle Communications Session Border Controller re-evaluates a SIP message against local policies, it is considered an additional routing stage. When multiple records are returned from an ENUM or LRT lookup, the Oracle Communications Session Border Controller evaluates the first response against all applicable local policies. If unsuccessful, the Oracle Communications Session Border Controller evaluates all additional responses, in turn, against all applicable local policies.

For example:



Multistage Routing Example 2

The following three local policy configuration elements are configured in the Oracle Communications Session Border Controller:

Local Policy 1

- from-address=*
- to-address=159
- source-realm=private

- policy-attribute
  - next-hop=lrt:default-lrt
  - lookup=multi
- policy-attribute
  - next-hop=192.168.200.50
  - lookup=single

Local Policy 2

- from-address=*
- to-address=192.168.1.49
- source-realm=private
- policy-attribute
  - next-hop=lrt:carrier-lrt
  - lookup=multi
- policy-attribute
  - next-hop=lrt:emergency
  - lookup=single

Local Policy 3

- from-address=*
- to-address=215680000002
- source-realm=private
- policy-attribute
  - next-hop=192.168.200.98
  - lookup=single
- policy-attribute
  - next-hop=192.168.200.97
  - lookup=single
- policy-attribute
  - next-hop=192.168.200.44
  - lookup=multi

```
<route>
   <user type="E164">159</user>
      <next type="regex">!^.*$!sip:11568000000@192.168.200.47!</next>
      <next type="regex">!^.*$!sip:215680000002@192.168.200.99!</next>
      <next type="regex">!^.*$!sip:11578000000@192.168.200.44!</next>
</route>

INVITE sip:159@192.168.1.49:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.48:5060
From: sipp <sip:sipp@192.168.1.48:5060>;tag=1
To:sut<sip:159@192.168.1.49:5060>
Call-ID: 1-4576@192.168.1.48
```

```
CSeq: 1 INVITE
Contact: sip:sipp@192.168.1.48:5060
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 135
```

The local route table in default-lrt appears as follows:

```
<route>
    <user type="E164">159</user>
                <next type="regex">!^.*$!sip:11568000000@192.168.200.47!</
next>
        <next type="regex">!^.*$!sip:215680000002@192.168.200.99!</next>
        <next type="regex">!^.*$!sip:11578000000@192.168.200.44!</next>
</route>
```

1. The Oracle Communications Session Border Controller receives an INVITE on realm, private (SDP is omitted below):

   ```
   INVITE sip:159@192.168.1.49:5060 SIP/2.0
   Via: SIP/2.0/UDP 192.168.1.48:5060
   From: sipp <sip:sipp@192.168.1.48:5060>;tag=1
   To: sut <sip:159@192.168.1.49:5060>
   Call-ID: 1-4576@192.168.1.48
   CSeq: 1 INVITE
   Contact: sip:sipp@192.168.1.48:5060
   Max-Forwards: 70
   Subject: Performance Test
   Content-Type: application/sdp
   Content-Length:  135
   ```

2. The Oracle Communications Session Border Controller performs a local policy search based on the following parameters:

   ```
   from-address:    sipp <sip:sipp@192.168.1.48:5060>;tag=1
   to-address:           sip:159@192.168.1.49:5060
   Source Realm:    private
   ```

3. The local policy search returns the four following routes to try:

   ```
   lrt:default-lrt
   192.168.200.50
   lrt:emergency
   lrt:carrier-lrt
   ```

   The first next-hop route will be an LRT query. In addition, this policy attribute is configured with lookup=multi, meaning the results of the LRT query should be used for another local policy query, i.e., a second stage. More specifically, the request-uri that was received in response to the LRT query will be used as the to-uri in the next LP query.

The Oracle Communications Session Border Controller performs the LRT lookup in the default-lrt configuration element and is returned the following:

```
sip:11568000000@192.168.200.47
sip:215680000002@192.168.200.99
sip:11578000000@192.168.200.44
```

The Oracle Communications Session Border Controller attempts to use the results from the LRT query for the next stage Local Policy lookup(s). Beginning with the first route and continuing in sequential order, the Oracle Communications Session Border Controller will try to route the outgoing INVITE message by performing additional Local Policy lookups on the remaining LRT query results, until the INVITE is successfully forwarded.

The Oracle Communications Session Border Controller performs a local policy query on:

```
sip:11568000000@192.168.200.47
```

Which equates to a local policy lookup on:

```
        from-URI=sipp <sip:sipp@192.168.1.48:5060>;
        to-URI=sip:11568000000@192.168.200.47
Source Realm: private
```

The query fails because there is no Local Policy entry for 11568000000.

The Oracle Communications Session Border Controller performs a second query on request-uri

```
sip:215680000002@192.168.200.99
```

Which equates to a local policy lookup on:

```
        from-URI=sipp <sip:sipp@192.168.1.48:5060>;
        to-URI=sip:215680000002@192.168.200.99
Source Realm: private
```

The LP query is successful and returns the following next- hops:

```
192.168.200.98
    192.168.200.99
192.168.200.44
```

The three routes shown above represent the next stage of the multistage routing for this INVITE. The policy attributes' lookup parameter is set to single for these next-hops. Therefore, the Oracle Communications Session Border Controller will attempt to send the outgoing INVITE message to one or more of these next-hops; there are no more stages to check.

4. The Oracle Communications Session Border Controller sends an INVITE to 192.168.200.98:

```
INVITE sip:215680000002@192.168.200.98;lr SIP/2.0
Via: SIP/2.0/UDP 192.168.200.49:5060
```

```
From: sipp <sip:sipp@192.168.1.48:5060>
To: sut <sip:159@192.168.1.49:5060>
Call-ID: SDnhae701-76e8c8b6e168958e385365657faab5cb-v3000i1
CSeq: 1 INVITE
Contact: <sip:sipp@192.168.200.49:5060;transport=udp>
Max-Forwards: 69
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 140
```

5. If the INVITE is sent to 192.168.200.98 successfully, the local policy routing will conclude and the call will continue processing. Otherwise the Oracle Communications Session Border Controller will try the other next hops until a route succeeds or all next-hops have been exhausted

Customizing Lookup Keys

When the **next hop** parameter points to perform an ENUM or LRT lookup, it can be provisioned with a "key=" attribute in order to specify a parameter other than the username to perform the lookup on. The following table lists the header, key value, and corresponding syntax to configure the Oracle Communications Session Border Controller with.

Username from Header:	Key Value	Example
To-URI	$TO	key=$TO
From-URI	$FROM	key=$FROM
P-Asserted-Identity	$PAI	key=$PAI

For a subsequent stage in multistage local policy routing, the lookup key to use for the next stage can be explicitly specified by configuring the **next key** parameter. By default, multistage lookups use the modified Request-URI returned from the ENUM/LRT response as the to-address key for the next local policy lookup. When the **next key** parameter is configured, its value will be used for the to-address key in the subsequent local policy lookup regardless if an ENUM or LRT lookup is configured for that policy attribute. The key syntax is for this parameter is the same as with the Routing-based RN and CIC feature.

Multistage Routing Lookup Termination

It is important for the Oracle Communications Session Border Controller to have a mechanism to stop performing additional stages of route lookups and limit the number of attempts and results to be tried. Routing termination can be performed at in the non-multistage way or at the global session router level.

Global Local Policy Termination

The Oracle Communications Session Border Controller can be configured to limit local policy lookups based several aspects of the route lookup process:

- Limiting the number of stages per message lookup—The Oracle Communications Session Border Controller can limit to the number of additional local policy lookup stages it will perform received message to a maximum of 5. This is configured with the **additional lp lookups** parameter. Leaving this parameter at its default value of 0 essentially disables multistaged local policy lookups.
- Limiting the number of routes per Local Policy lookup—The Oracle Communications Session Border Controller can limit the number of route results to use as returned for each

Local-Policy lookup. This is configured with the **max lp lookups routes per lookup** parameter. Leaving this parameter at its default value of 0 places no limit on the number of returned routes the Oracle Communications Session Border Controller can try.

- Limiting the total number of routes for all local policy lookups per message request—The Oracle Communications Session Border Controller can limit the number of route returned in total across all lookups for a given request, including additional stages. This is configured with the **total lp routes** parameter. Leaving this parameter at its default value of 0 places no limit on the number of returned routes the Oracle Communications Session Border Controllercan try. This parameter overrides any configured options.

Additionally, the Oracle Communications Session Border Controller monitors for local policy lookup loops which could cause a significant deterioration in performance. If a loop is found, the Oracle Communications Session Border Controller stops trying the looping route list and proceeds to try any remaining routes..

Multistage Local Policy Routing Configuration

To set up your local policy attributes for routing using the TO header:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit a local policy.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

   ```
   ORACLE(local-policy)# policy-attributes
   ORACLE(local-policy-attributes)#
   ```

5. **next-hop**—This is the next signaling host and/or object to query. This parameter can be configured as an IP address, ENUM server, or LRT. You can also add a lookup key to an ENUM server or LRT lookup with the following syntax:

   ```
   next-hop     enum:ENUM-object;key=$TO
   ```

6. **terminate-recursion**—Set this parameter to **enabled** to terminate local policy route recursion when the current stage completes.
7. **lookup**—Leave this parameter at the default **single** for single stage local policy routing or set it to **multi** to enable multistage local policy routing.
8. **next-key**—Set this parameter to $TO, $FROM, or $PAI if you wish to override the recently-returned lookup key value for the next stage.
9. Save and activate your configuration.

Maintenance and Troubleshooting

The **show sipd policy** command includes four additional counters that refer to single and multistage local policy lookups. All counters are reported for the recent period, and lifetime total and lifetime period maximum. These counters are:

- Local Policy Inits—Number of times the Oracle Communications Session Border Controller makes an initial local policy lookup.
- Local Policy Results Max—Number of times the Oracle Communications Session Border Controller truncated the number of routes returned for a local policy lookup because the maximum number of routes per local policy lookup (**max lp lookups routes per lookup**) threshold was reached.
- Local Policy Exceeded—Number of times the Oracle Communications Session Border Controller truncated the number of routes returned for a local policy lookup because the maximum number of routes per message request (**total lp routes**) threshold was reached.
- Local Policy Loops—Number of times the Oracle Communications Session Border Controller detected a loop while performing a multistage local policy lookup.

Traps

An SNMP trap is generated to notify that the limit on the **additional lp lookups** threshold has been reached during the recent window period. This trap occurs a maximum of once during a window period.

```
apSysMgmtLPLookupExceededTrap NOTIFICATION-TYPE
    STATUS           current
       DESCRIPTION
          " The trap will be generated the first time the additional Local
Policy Lookups limit is reached is in the recent window period. The trap will
only occur once during a window period."
       ::= { apSystemManagementMonitors 65}
```

Routing-based RN and CIC

When the Oracle Communications Session Border Controller performs local policy routing, it selects local policy entries based on from addresses, to addresses, and source realms. All three are configurable in the local policy configuration. The to addresses can either be the username in a Request-URI (if it is an E.164/phone number format), or the request-URI's hostname or IP address. The Oracle Communications Session Border Controller sorts matching local policies based on policy attribute entries. A policy attribute defines a next hop, which can be a session agent or a session agent group. Alternatively, the next hop might define an ENUM server group or local route table to use to find the next hop.

If the routing-based RN and CIC feature is not enabled, the Oracle Communications Session Border Controller performs the subsequent ENUM query or local route table lookup using the Request-URI's username, if it is a telephone number (TN). The TN is the normalized user part of the Request-URI, ignoring any user parameters or non-digit characters.

If the routing-based RN and CIC feature is enabled, the Oracle Communications Session Border Controller instead performs the ENUM or local route table lookup based on a user parameter, which is useful for lookups based on routing number (RN) or carrier identification code (CIC):

- An RN is a number that identifies terminating switch nodes in Number Portability scenarios when the original TN has been moved to the switch defined by the RN.
- A CIC is the globally unique number of the terminating carrier to which a ported number has been moved.

In applications where the Oracle Communications Session Border Controller is given the RN or the CIC in the Request-URI, this feature is useful because the Oracle Communications Session Border Controller can perform an additional ENUM or local route table lookup to find the next hop to the RN or the CIC. Typically, ENUM servers have imported Number Portability data with which to respond to the Oracle Communications Session Border Controller query, and (for example) the Oracle Communications Session Border Controller can use local route tables for storing CIC values for direct carrier hand-off.

Even with this feature enabled, the Oracle Communications Session Border Controller still performs local policy match selection based on the TN. This feature only uses the RN or CIC user-parameter for the ENUM or local route table lookup after the local policy and policy attributes have been selected.

Routing-based RN Configuration

This section shows you how to specify that a set of local policy attributes should use an RN for lookup. You can also set this value to CIC, or to any value you require.

You can set the lookup key to an RN in the local policy attributes' **next-hop** parameter.

To set the lookup key to RN:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-policy** and press Enter.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **policy-attributes** and press Enter.

   ```
   ORACLE(local-policy)# policy-attributes
   ORACLE(local-policy-attributes)#
   ```

5. **next-hop**—In the **next-hop** parameter—after the kind of ENUM service used—type a colon (;). Then, without spaces, type in **key=rn** and press Enter.

   ```
   ORACLE(local-policy-attributes)# next-hop lrt:lookup;key=rn
   ```

6. Save and activate your configuration.

Codec Policies for SIP

The Oracle Communications Session Border Controller has the ability to add, strip, and reorder codecs for SIP sessions. This builds on the SBC's pre-existing abilities to route by codec and reorder one codec in an SDP offer by allowing you to configure the order of multiple codecs and to remove specific codecs within the media descriptions in SDP offers.

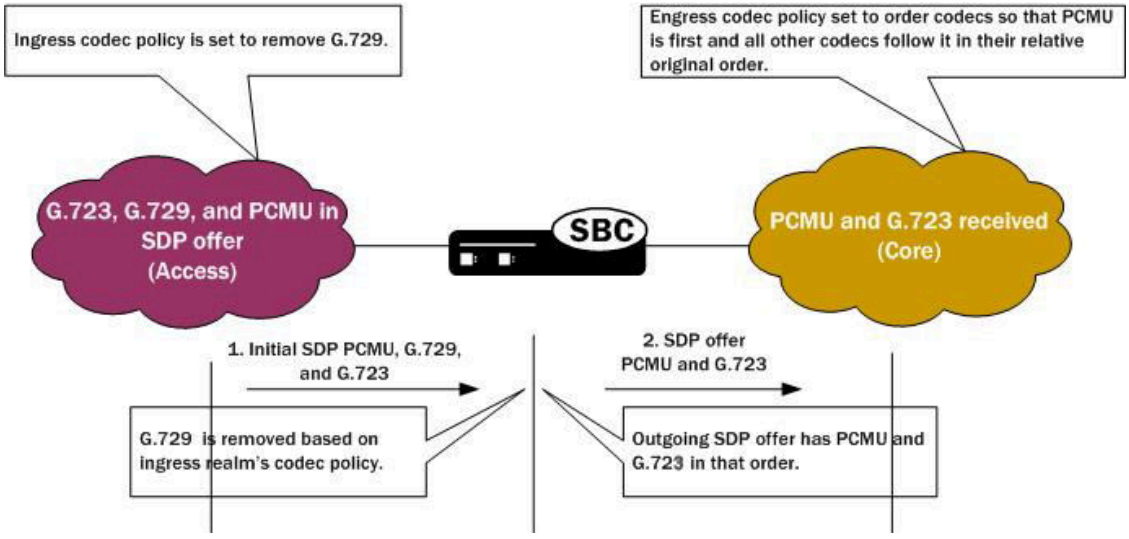
You can enable the SBC to perform these operations on SDP offers by configuring codec policies. Codec policies are sets of rules that specify the manipulations to be performed on SDP offers. They are applied on an ingress and egress basis using the realm and session agent configurations.

SBC supports three types of codec policies:

- Ingress policy—Codec policy that the SBC applies to the SDP offer for incoming traffic
- Egress policy—Codec policy that the SBC applies to the SDP offer for traffic leaving the SBC
- Conditional policy—Codec policy that the SBC applies to the SDP offer for traffic leaving the SBC. A conditional policy differs from an egress policy in providing the capability to perform standard codec manipulations (add and strip) dynamically, based on the codec list and associated parameters contained in the original SDP offer.

The SBC applies codec policies during the offer phase of media format negotiation. If codec manipulation is enabled, then the SBC performs the modification according to the specific policy and forwards on the traffic.

For example, when the SBC receives a SIP INVITE with SDP, it refers to the realm through which the INVITE arrived and performs any manipulations specified by an ingress codec policy that may have been assigned to the ingress realm. With the media description possibly changed according to the ingress codec policy, the SBC passes the SDP offer to the outgoing realm so that the an egress codec policy can be applied. Note that the SDP to be evaluated by the egress codec policy may match the original SDP, or it may have been changed during transit through the ingress realm. After applying the egress coded policy, the SBC forwards the INVITE.



Since the offer-answer exchange can occur at different stages of SIP messaging, the assigned ingress and egress roles follow the media direction rather than the signaling direction. It might be, for example, that the offer is in an OK that the SBC modifies.

You can apply codec policies to realms and to session agents; codec policies configured in session agents take precedence over those applied to realms. However, it is not required that there be both an ingress and an egress policy either for realms or for session agents. If either one is unspecified, then no modifications take place on that side. If neither ingress nor egress policies specified, SDP offers are forwarded as received.

Relationship to Media Profiles

For each codec that you specify in a codec policy, there must be a corresponding media profile configuration on the Oracle Communications Session Border Controller. You configure media profiles in the ACLI via the session-router path. In them, you can specify codec type, transport protocol, required bandwidth, and a number of constraints.

Manipulation Modes

You can configure a codec policy to perform several different kinds of manipulations:

- Allow—List of codecs that are allowed for a certain codec policy; if a codec does not appear on this list, then the Oracle Communications Session Border Controller removes it. You can wildcard this list with an asterisk (*) so that all codecs are allowed. Further, you can create exceptions to a wildcarded allow list.
  - You make an exception to the wildcarded list of codecs by entering the codec(s) that are not allowed with a **no** attribute. This tells the Oracle Communications Session Border Controller to allow all codecs except the one(s) you specify.

    ```
    ACMEPACKET(codec-policy)# allow-codecs (* PCMA:no)
    ```

  - You can also create exceptions to allow lists such that audio or video codecs are removed. However, when the allow list specifies the removal of all audio codecs and an INVITE arrives at the Oracle Communications Session Border Controller with only audio codecs, the Oracle Communications Session Border Controller behaves in accordance with RFC 3264. This means that the resulting SDP will contain one attribute line, with the media port for the media line set to 0. The terminating side will need to supply new SDP in its reply because the result of the manipulation is the same as an INVITE with no body.

    ```
    ACMEPACKET(codec-policy)# allow-codecs (* audio:no)
    ```

- Order—List of the codecs where you specify their preferred order in the outgoing media offer. The Oracle Communications Session Border Controller arranges matching codecs according to the rule you set, and any remaining ones are added to the list in the same relative order they took in the incoming media offer. If your list specifies a codec that is not present, then the ordering proceeds as specified but skips the missing codec.
  You can use an asterisk (*) as a wildcard in this list, too. The placement of the asterisk is key, as you can see in the following examples:
  - For an order rule set this way

    ```
    ACMEPACKET(codec-policy)# order (A B C *)
    ```

    codecs A, B, and C will be placed at the front of the codec list in the order specified; all other codecs in the offer will follow A, B, and C in the same relative order they had in the original SDP offer.

  - For an order rule set this way:

    ```
    ACMEPACKET(codec-policy)# order (* A B C)
    ```

    codecs A, B, and C will be placed at the end of the codec list in the order specified; all other codecs in the offer will come before A, B, and C in the same relative order they had in the original SDP offer.

  - For an order rule set this way

    ```
    ACMEPACKET(codec-policy)# order (A * B C)
    ```

    codec A will be placed at the beginning of the codec list, to be followed by all other codecs in the offer in the same relative order they had in the original SDP offer, and then B and C will end the list.

- Force—An attribute you can use in the allow list with one codec to specify that all other codecs should be stripped from the outgoing offer. You can specify multiple forced codecs in your rules.
  - If you set multiple codecs in the allow list and one of them is forced, then the outgoing offer will contain the forced codec.
  - If you set multiple codecs in the allow list and the one that is forced is not present in the offer, then the Oracle Communications Session Border Controller will select a non-forced codec for the outgoing offer.

    ```
    ACMEPACKET(codec-policy)# allow (PCMU G729:force)
    ```

  You cannot use the force attribute with a wildcarded allow list.

  > **Note:**
  >
  > The Force attribute can only be applied to ingress realms receiving the offers, and is not applied to egress realms.

- No—An attribute that allows you to strip specified codecs or codec types from a wildcarded allow list.

  ```
  ACMEPACKET(codec-policy)# allow (* PCMA:no)
  ```

In-Realm Codec Manipulation

In addition to being able to apply codec policies in realms, the realm configuration supports a setting for determining whether codec manipulation should be applied to sessions between endpoints in the same realm.

In-realm codec manipulation can be used for simple call flows that traverse two realms. If the originating and terminating realms are the same, the Oracle Communications Session Border Controller checks to see if you have enabled this capability. If you have enabled it, then the Oracle Communications Session Border Controller performs the specified manipulations. If this capability is not enabled, or if the realm's media management in realm (**mm-in-realm**) setting is disabled, then the Oracle Communications Session Border Controller does not perform codec manipulations.

For more complex calls scenarios that involve call agent or reinitiation of a call back to the same realm, the Oracle Communications Session Border Controller does not perform in-realm codec manipulation.

Codec Policy Configuration

This section gives instructions and examples for how to configure codec policies and then apply them to realms and session agents. It also shows you how to configure settings for in-realm codec manipulation.

Creating a Codec Policy

To create a codec policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

   ```
   ACMEPACKET(configure)# media-manager
   ACMEPACKET(media-manager)#
   ```

3. Type codec-policy and then press Enter.

   ```
   ACMEPACKET(media-manager)# codec-policy
   ACMEPACKET(codec-policy)#
   ```

4. **name**—Enter the unique name for the codec policy. This is the value you will use to refer to this codec policy when you apply it to realms or session agents. This parameter is required and is empty by default.
5. **allow-codecs**—Enter the list of media format types (codecs) to allow for this codec policy. In your entries, you can use the asterisk (*) as a wildcard, the force attribute, or the no attribute so that the allow list you enter directly reflects your configuration needs. Enclose entries of multiple values in parentheses (( )).

   The codecs that you enter here must have corresponding media profile configurations.
6. **add-codecs-on-egress**—Enter the codecs that the Oracle Communications Session Border Controller adds to an egress SDP offer if that codec is not already there. This parameter applies only to egress offers.

   The codecs that you enter here must have corresponding media profile configurations.

   add-codecs-on-egress can be used to construct ingress, egress, or conditional codec policies.
7. **order-codecs**—Enter the order in which you want codecs to appear in the outgoing SDP offer. Remember that you can use the asterisk (*) as a wildcard in different positions of the order to directly reflect your configuration needs. Enclose entries of multiple values in parentheses (( )).

   The codecs that you enter here must have corresponding media profile configurations.
8. Save and activate your configuration.

Your codec policy configuration will resemble the following example:

```
codec-policy
        name            private
        allow-codecs    g723:no pcmu video:no
        order-codecs    pcmu
```

Applying a Codec Policy to a Realm

Note that codec policies defined for session agents always take precedence over those defined for realms.

To apply a codec policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
5. Save and activate your configuration.

Applying a Codec Policy to a Session Agent

Note that codec policies that are defined for session agents always take precedence over those that are defined for realms.

To apply a codec policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ```

   If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
5. Save and activate your configuration.

In-Realm Codec Manipulations

To enable in-realm codec manipulations:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-manip-in-realm**—Enter the name of the codec policy that you want to apply to this realm. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

QoS Based Routing

In addition to configuring your system for routing based on certain session constraints, you can also set up routing based on QoS. QoS based routing uses the R-Factor on a per-realm basis to either cut back on the traffic allowed by a specific realm, or to shut that traffic off altogether.

To use this feature, you set up QoS constraints configurations and apply one per realm. The QoS constraints configuration allows you to set up two thresholds:

- Major—The major threshold sets the R-Factor limit beyond which the Oracle Communications Session Border Controller rejects a certain percentage (that you configure) of calls. That is to say, it rejects inbound calls at the rate you set with a 503 Service Unavailable status code, and rejects outbound calls if there are no alternative routes.
- Critical—The critical threshold, when exceeded, causes the Oracle Communications Session Border Controller to behave the same way it does when any of the session constraints (set in the session-constraints configuration) are exceeded. All inbound calls to the realm are rejected with a 503 Service Unavailable status code, and (if there is no alternate route) outbound calls are rejected, too. Until the R-Factor falls within acceptable means and the session constraint's time-to-resume value has elapsed, the realm remains in this state.

Management

This feature is supported by MIBs and traps. Historical data recording (HDR) also supports this feature by providing the following metrics in the session realm statistics collection group:

- Average QoS RFactor (0-93)
- Maximum QoS RFactor (0-93)
- Current QoS Major Exceeded
- Total QoS Major Exceeded
- Current QoS Critical Exceeded
- Total QoS Critical Exceeded

QoS Contraints Configuration

This section shows you how to configure a QoS constraints configuration and then how to apply it to a realm.

Configuring QoS Constraints

Your first step to enabling QoS based routing is to set up a QoS constraints configuration. This configuration is where you enter major and critical thresholds, as well as the load reduction for the realm should the R-Factor exceed the major threshold.

To set up a QoS constraints configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **qos-constraints** and press Enter.

   ```
   ORACLE(session-router)# qos-constraints
   ORACLE(qos-constraints)#
   ```

4. **name**—Enter the name of this QoS constraints configuration. This parameter uniquely identifies the configuration, and you use this value when applying the configuration to a realm. This parameter has no default and is required.
5. **state**—Set the state of this QoS constraints configuration. The default is **enabled**, but you can set this parameter to **disabled** if you want to stop applying these constraints.
6. **major-rfactor**—Enter a numeric value between **0** (default) and **9321** to set the threshold that determines when the Oracle Communications Session Border Controller applies the call reduction rate. If you leave this parameter set to **0**, then the Oracle Communications Session Border Controller will not apply a major threshold for any realm where you apply this QoS constraints configuration.

Note that this value must be greater than that you set for the **critical-rfactor**, except when the **major-rfactor** is 0.

7. **critical-rfactor**—Enter a numeric value between **0** (default) and **9321** to set the threshold that determines when the Oracle Communications Session Border Controller rejects all inbound calls for the realm, and rejects outbound calls when there is no alternate route. If you leave this parameter set to **0**, then the Oracle Communications Session Border Controller will not apply a critical threshold for any realm where you apply this QoS constraints configuration.

   Note that this value must be less than that you set for the **major-rfactor**, except when the **major-rfactor** is 0.

8. **call-load-reduction**—Enter a number from **0** (default) to **100** representing the percentage by which the Oracle Communications Session Border Controller will reduce calls to the realm if the **major-rfactor** is exceeded. If you leave this parameter set to 0, then the Oracle Communications Session Border Controller will not reduce call load for the realm—even when the major-rfactor is configured.

   This is the percentage of inbound and outbound calls the Oracle Communications Session Border Controller will reject. For example, if you set this parameter to 50 and the major threshold is exceeded, then the Oracle Communications Session Border Controller rejects every other call to the realm.

9. Save and activate your configuration.

Applying QoS Constraint to a Realm

You apply QoS constraints to realms using the **qos-constraint** parameter.

To apply a QoS constraint to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-config** and press Enter. If you adding this feature to a pre-existing realm, then you need to select and edit that realm.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. **qos-constraints**—Enter the name value from the QoS constraints configuration you want to apply to this realm.

   Save and activate your configuration.

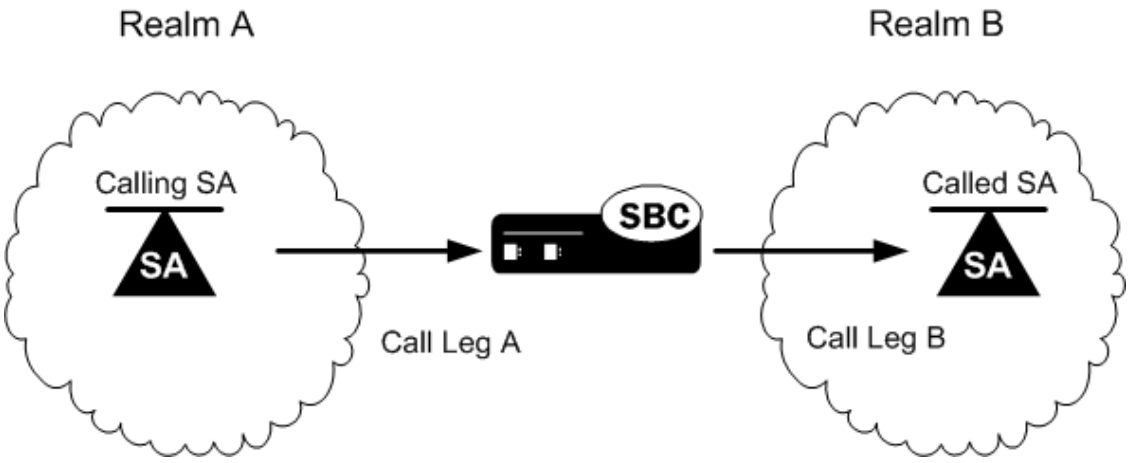
10

Session Translation

Session translations are changes to a layer-5 endpoint header. Session translations can modify SIP headers like the To, From, Diversion, History-Info, or P-Asserted-Id headers. They can also modify parameters like ISUP calling party, ISUP called party, ISUP generic number, ISUP redirect number, and ISUP original called number. They can modify the To and From headers of H323 messages.

With session translations, you can strip address prefixes added by external gateways. Or you can add a string tag to an address for the SBC's local policy routing and then remove the tag upon egress. The most common use of session translation is to add or remove a "1" or a "+" from a phone number sent from or addressed to a device.

Session translations are applied to inbound and outbound call legs independently. On the inbound call leg (Call Leg A), session translations are performed after HMRs and before routing occurs. On the outbound call leg (Call Leg B), session translations are performed after routing and before HMRs.



Session translations are attached to either session agents or to realms. Session translations attached to session agents take precedence over session translations attached to realms. If no session translation is applied to a session agent, then the SBC will use the session translation applied to a realm. If a session translation is applied to both a realm and session agent, the translation attached to the session agent will apply. If session agents and realms have no associated translations, then all headers will remain in their original state as they pass through the SBC.

SBC session translations are implemented in three steps.

1. First, the individual translation rules are defined in the **translation-rules** element.
2. Next, the translation rules are grouped in a specified order in the **session-trans-rule** subelement of the **session-translation** element.
3. Finally, session translations are attached to either session agents or realms in the **session-agent** element or **realm-config** element.

Test Translation Limitations

Although you can use the **test-translation** command to test the address translation feature, you can only use it for rules configured with specific **input-header-type** and **output-header-type** values. Applicable rules include those configured with the **called-header** or **calling-header** values for both the input and output header type.

Session Translation Headers

Translation rules change predefined parts or parameters within SIP headers.

Table 10-1 SIP Headers to header-type mapping

ACLI input/output header-type value	SIP Header
request-uri	The address or number of the Request URI. `INVITE sip:`**`service`**`@10.1.1.4:5060 SIP/2.0`
called-header	The complete user part of the To header. `sip:`**`2222;phone-context=example.com;ext=1234`**`@example.com:5060`
called-address-or-number	The address or number within the To header. `sip:`**`2222`**`;phone-context=example.com;ext=1234@example.com:5060`
called-extension	The extension parameter within the To header. `sip:2222;phone-context=example.com;ext=`**`1234`**`@example.com:5060`
called-phone-context	The phone-context parameter in the To header. `sip:2222;phone-context=`**`example.com`**`;ext=1234@example.com:5060`
calling-header	The complete user part of the From header. `sip:`**`2222;phone-context=example.com;ext=1234`**`@example.com:5060`

Table 10-1 (Cont.) SIP Headers to header-type mapping

ACLI input/output header-type value	SIP Header
calling-address-or-number	The address or number within the From header. `sip:`**`2222`**`;phone-context=example.com;ext=1234@example.com:5060`
calling-extension	The extension parameter within the From header. `sip:2222;phone-context=example.com;ext=`**`1234`**`@example.com:5060`
calling-name	The name in the From header. `From:` **`Bob`** `<sip:2222;phone-context=example.com;ext=1234@example.com:5060>;tag=1`
calling-phone-context	The phone-context parameter in the From header. `sip:2222;phone-context=`**`example.com`**`;ext=1234@example.com:5060`
redirect-header	The Diversion header. `Diversion:` **`<sip:2222@example.com>;reason=user-busy`**
redirect-address-or-number	The address or phone number of the Diversion header. `Diversion: <sip:`**`2222`**`@example.com>;reason=user-busy`
p-asserted-id-header	The P-Asserted-Identity header. `P-Asserted-Identity: tel:`**`5555555555`**
history-info-header	The History-Info header. `History-Info:` **`<sip:bob@example.com>;index=1`**
isup-called-party-number	Same field as called-address-or-number in an ISUP context.
isup-calling-party-number	Same field as calling-address-or-number in an ISUP context.
isup-generic-number	When isup-generic-number is set and the number qualifier indicator is set to "additional calling party number" (00000110), the SBC replaces the user part in the From header with the generic number. The SBC also prefixes the From header with a plus sign (+) if the nature of address is set to international.

Table 10-1 (Cont.) SIP Headers to header-type mapping

ACLI input/output header-type value	SIP Header
isup-redirect-number	The redirect number is the number that most recently redirected. If there are multiple redirects, this changes with each redirect.
isup-original-called-number	When a call is redirected, the original called number is the first number that was called. When there are multiple redirects, this number is different from the redirect number.

Session Translation Configuration

Configuring a session translation requires several steps.

1. Configure individual translation rules in the **translation-rules** element.
2. Group these rules in the **session-translation** element.
3. Apply a group of rules to a session agent or realm using the **session-agent** or **realm-config** elements.

Configure Translation Rules

You create translation rules in the **translation-rules** element.

1. Access the **translation-rules** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# translation-rules
ORACLE(translation-rules)#
```

2. **id**—Set the name of this translation rule.

   You'll use this name later when you group translation rules within the **session-translation** element.
3. **description**—(Optional) Provide a description of what this translation rule does.
4. **input-header-type**—Select the header type that the translation rule will apply to.

   The translation rule will check for the existence of this header, user part, or parameter. If a particular SIP message doesn't contain it, the rule won't modify that message.
5. **input-header-value**—Enter the regex pattern to use when searching the input header.

   If the regex pattern does not match, no modification happens.

   If the regex pattern does match, the whole output header will be set to the value of **output-header-value**.

   When entering regex patterns on the ACLI, wrap the pattern in double quotes if it includes spaces.
6. **output-header-type**—Enter the header whose value will be modified when the rule is executed.
7. **output-header-value**—Enter the new value of the output header.

If your **input-header-value** includes regex capture groups, use the **$<group number>** syntax to identify the captured content.

When using capture groups that are followed by numbers, use the **$0<group number>** syntax for single-digit capture groups. For example, the SBC will read `$1781` as capture group 17 followed by the literal digits 81. The SBC will read `$01781` as capture group 1 followed by the literal digits 781. Capture groups that are followed by a letter, such as `$3a`, are not affected by this rule.

8. Type **done** to save your work.

```
translation-rules
        id                                      prependPlusOne
        description
        input-header-type                       calling-address-or-number
        input-header-value                      ^(.*)$
        output-header-type                      calling-address-or-number
        output-header-value                     +1$1


translation-rules
        id                                      changeFromDomain
        description
        input-header-type                       calling-header
        input-header-value                      (.*)example.com(.*)
        output-header-type                      calling-header
        output-header-value                     $01example.com$02
```

Configure Session Translation

Group your translation rules in the **session-translation** element. Within a session-translation group, you can enable or disable specific rules, declare whether a rule is required, and define the order in which your rules execute.

1. Access the **session-translation** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-translation
   ORACLE(session-translation)#
   ```

2. **id**—Enter a name for this session translation.

   You'll use this name later when applying it to a session agent or realm.

3. Navigate to the **session-trans-rule** element.

   ```
   ORACLE(session-translation)# session-trans-rule
   ```

4. **rule-id**—Enter the name of the translation-rule element you want to add to this session-translation group.

   The value here matches the **id** attribute of the translation rule you previously configured.

5. **mandatory**—Set whether this rule is required for the session translation group.

   If one mandatory rule fails to be applied successfully, all the rules within that session translation group are reverted. The default is disabled.

6. **move**—Move a translation rule from one position to another.

   The syntax for the **move** command.

   ```
   move <from-position> <to-position>
   ```

   For example, to move the group's second rule to the first position so that it gets executed first, run this command:

   ```
   move 2 1
   ```

7. Type **done** to save your translation rule.
8. Add more translation rules if necessary.
9. Type exit to return to the **session-translation** element.
10. Type **done** to save your session translation.

Apply a Session Translation

Session translations can be applied to session agents and realms.

The **session-agent** element and the **realm-config** element both contain the two subelements **in-session-translation** and **out-session-translation**. These two fields are populated with session translation element IDs.

If none of these fields are populated, no translations take place and the original address will remain unchanged as it traverses the SBC. Further, session translations applied to a session agent takes precedence over one applied to a realm.

Apply a Session Translation to a Session Agent

To configure number translation for a session agent:

1. Access the **session-agent** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)
   ```

2. Select the **session-agent** object to edit.

   ```
   ORACLE(session-agent)# select
   <hostname>:
   1: 192.168.100.101:1813

   selection: 1
   ORACLE(session-agent)#
   ```

3. If configuring an inbound session translation, navigate to the **in-session-translations** subelement. If configuring an outbound session translation, navigate to the **out-session-translations** subelement.
4. **in-session-translation-id**—If configuring an inbound session translation, enter the **id** of the session-translation to apply to this session agent.

5. **out-session-translation-id**—If configuring an outbound session translation, enter the **id** of the session-translation to apply to this session agent.
6. **state**—Set this to enabled.
7. **move**—Move a session-translation from one position to another.

   The syntax for the **move** command.

   ```
   move <from-position> <to-position>
   ```

   For example, to move the group's second rule to the first position so that it gets executed first, run this command:

   ```
   move 2 1
   ```

8. Type **done** to save your configuration.

Apply a Session Translation to a Realm

To configure number translation for a realm:

1. Access the **realm-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

2. Select the **realm-config** object to edit.

   ```
   ORACLE(realm-config)# select
   identifier:
   1: realm01 left-left:0 0.0.0.0

   selection: 1
   ORACLE(realm-config)#
   ```

3. If configuring an inbound session translation, navigate to the **in-session-translations** subelement. If configuring an outbound session translation, navigate to the **out-session-translations** subelement.
4. **in-session-translation-id**—If configuring an inbound session translation, enter the **id** of the session-translation to apply to this realm.
5. **out-session-translation-id**—If configuring an outbound session translation, enter the **id** of the session-translation to apply to this realm.
6. **state**—Set this to enabled.
7. **move**—Move a session-translation from one position to another.

   The syntax for the **move** command.

   ```
   move <from-position> <to-position>
   ```

For example, to move the group's second rule to the first position so that it gets executed first, run this command:

```
move 2 1
```

8. Type **done** to save your configuration.

Message Types

The SBC only applies session-translation processing to a limited number of out-of-dialog requests:

- INVITE
- CANCEL
- REFER
- OPTIONS
- SUBSCRIBE

The SBC does not manipulate any header within in-dialog requests using session-translation or translation-rules.

11

Admission Control and QoS

This chapter describes how to configure the Oracle Communications Session Border Controller for call admission control and Quality of Service (QoS) monitoring. Call admission control lets you manage call traffic based on several different policies. It is aimed at managing call admission rates in the network, enabling you to maintain suitable QoS levels. A new call is admitted only if it meets the requirements

QoS reporting provides you with real-time evaluation of network and route performance. It lets you contrast internal domain and external domain performance and facilitates SLA verification and traffic engineering.

About Call Admission Control

The Oracle Communications Session Border Controller provides call admission control capabilities based on the following policies:

- Bandwidth (single and multi-level policies)
- Session capacity
- Session rate (sustained and burst)

> **Note:**
>
> In order to provide admission control for networks to which the Oracle Communications Session Border Controller is not directly connected, you need to define multiple realms per network interface.

Bandwidth-Based Admission Control

The Oracle Communications Session Border Controller is a policy enforcement point for bandwidth-based call admission control. Sessions are admitted or rejected based on bandwidth policies, configured on the Oracle Communications Session Border Controller for each realm.

To manage bandwidth consumption of a network's overall capacity, you can configure aggregate bandwidth policies for each realm.

As the Oracle Communications Session Border Controller processes call requests to and from a particular realm, the bandwidth consumed for the call is decremented from the bandwidth pool for that realm. The Oracle Communications Session Border Controller determines the required bandwidth from the SDP/H.245 information for SIP and from the OLC sent in the SETUP message for H.323. Any request that would cause the bandwidth constraint to be exceeded is rejected with a SIP 503 Service Unavailable or an H.323 Release Complete.

For example, if an incoming SIP message requests PCMU for a payload/encoding name, a zero (0) payload type, and an 8000 cycle clock rate, the Oracle Communications Session Border Controller must determine how much bandwidth is needed.

To accomplish this task, the system checks the media profile values and reserves the bandwidth required for flows. If the required bandwidth for the new flow exceeds the available bandwidth at the time of the request, the Oracle Communications Session Border Controller rejects the session.

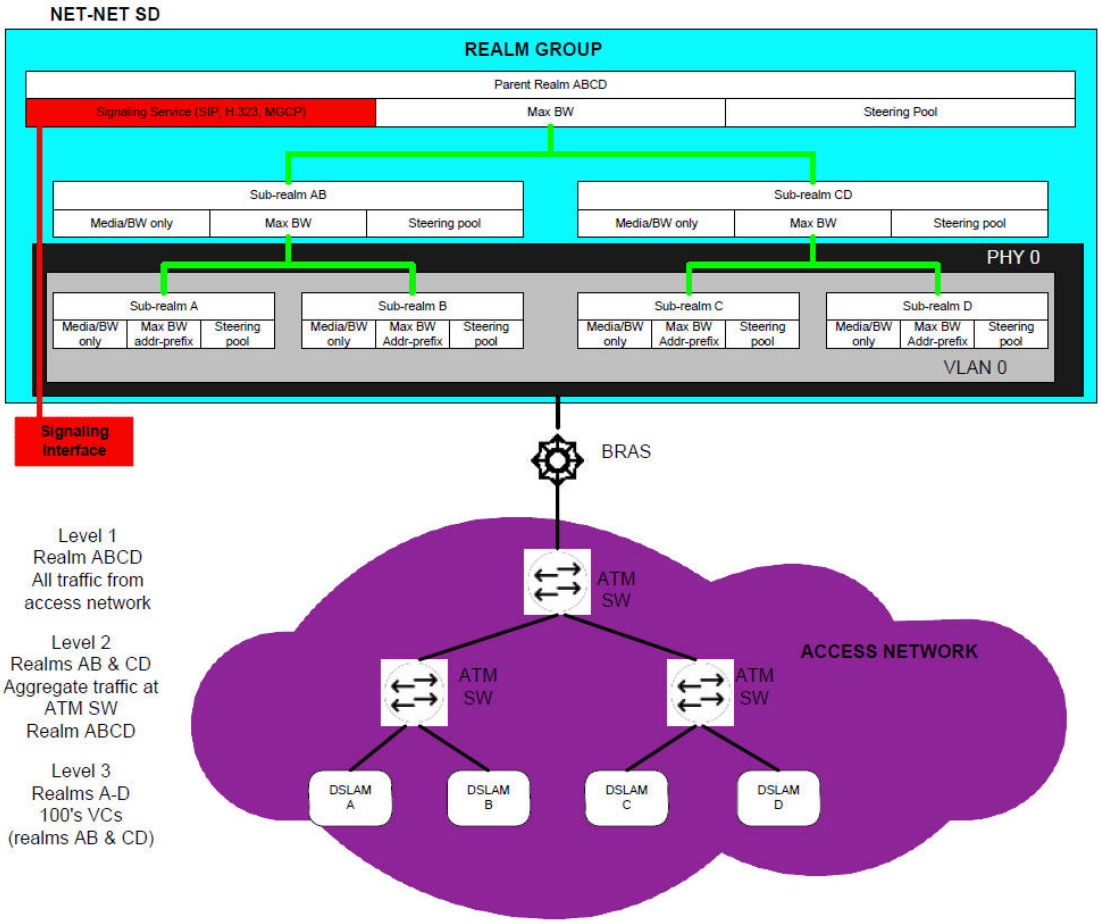
With these mechanisms, the Oracle Communications Session Border Controller provides bandwidth-based admission control.

Multi-Level Bandwidth Policy Nesting

Multi-level nesting of bandwidth policy enforcement addresses the following issues:

- Bandwidth over-subscription: access or transit transport networks are aggregated and/or oversubscribed. For example, digital subscriber lines (DSL), Frame Relay (FR), and Asynchronous Transfer Mode (ATM). Admission control policies must reflect access network topology.
- Bandwidth partitioning for multiple services: access or transit bandwidth is partitioned among multiple service profiles (SIP, for example) in the same customer network.
- Multi-site VPN environments: admission control must be applied at the site level as well as the VPN level.

The following example illustrates different scenarios; in each there are two or more levels of admission control required. Nested admission control is best depicted by the DSL broadband example.



In DSL access networks, ATM network bandwidth is typically oversubscribed at rates up to 400/1. At Level 3 (above), hundreds of users virtual circuits (VCs) are aggregated to a smaller set of virtual paths (VPs) at each DSLAM. At Level 2, many virtual paths are aggregated at the first ATM switch. Finally, at Level 1, all traffic from all subscribers in the access network is aggregated at the BRAS. Each level of aggregation is oversubscribed, creating the need to perform admission control at each level.

From a SBC perspective, multiple tiers of realms are supported, each with its unique bandwidth policy. Only the lowest order realm (Level 3) requires an address prefix (that assigned to the DSLAM) that must be used by the SBC to determine in which realm a user resides. When a call request to or from a particular user is received, the SBC checks each realm in the path to determine whether sufficient bandwidth is available to place the call.

Session Capacity- and Rate-based Admission Control

A session agent defines a signaling endpoint. It is a next hop signaling entity that can be configured to apply traffic shaping attributes. You can define concurrent session capacity and rate attributes for each session agent.

You can configure a set of attributes and constraints for each session agent to support session access control. In this configuration, the Oracle Communications Session Border Controller only accepts requests from configured session agents. And you can set up session admission control so that the Oracle Communications Session Border Controller limits the number of concurrent inbound and outbound sessions for any known service element.

The Oracle Communications Session Border Controller denies a call request to any destination that has exceeded its configured policies for session capacity and session rate. The Oracle Communications Session Border Controller might reject the call request back to the originator. If multiple destinations are available, the Oracle Communications Session Border Controller will check current capacity and rate for each destination and attempt to route the call only to destinations whose policy limits have not been reached.

You assign a media profile to a session agent and indicate whether the transport protocol is SIP or H.323. If the protocol is H.323, you need to indicate whether the session agent is a gateway or a gatekeeper.

Constraints for Proxy Mode

The Oracle Communications Session Border Controller applies session router and session agent constraints when it is in proxy (transaction or stateless) mode if you enable the ACLI **constraints** parameter for a session agent. However, the Oracle Communications Session Border Controller does not track SIP sessions when in transaction mode, so the following session-specific constraints are not applied:

- max-sessions
- max-inbound-sessions
- max-outbound-sessions
- min-seizures
- min-asr

Constraints the Oracle Communications Session Border Controller applies are:

- max-burst-rate
- max-inbound-burst-rate

- max-outbound-burst-rate
- max-sustain-rate
- max-inbound-sustain-rate
- max-outbound-sustain-rate

In order to set the desired time windows for computing burst rates and sustain rates, you also need to configure these parameters in the session agent configuration: **burst-rate-window** and **sustain-rate-window**. You can also set the **time-to-resume** and in-service-period parameters to control how long to wait before bringing a session agent back into service after its constraints are no longer exceeded.

CAC Policing and Marking for non-Audio non-Video Media

The Oracle Communications Session Border Controller supports non-AVT (audio-visual transport) media profile and media policy configurations.

In previous releases, the Oracle Communications Session Border Controller only policed media based on average rate limits configured in media profiles, but these are only applied to AVT. And if there are not required bandwidth or average rate limit values set for the media profile, CAC and policing functions are not applied to media—even if the SDP specifies appropriate bandwidth values. Likewise, ToS markings are not applied for non-AVT media, but only for SIP, H.323, and AVT media types.

With this feature addition, you can now enable your Oracle Communications Session Border Controller to handle non-AVT media types like image and text, and use application and data type for policing purposes. Bandwidth CAC support has also been added for non-AVT media types, as has support for the application specific (AS) bandwidth modifier (b=AS:<value>) in the SDP with specification of a defined amount of headroom for that value.

Bandwidth CAC Fallback Based on ICMP Failure

For networks where backup links (operating in active-standby mode) from CE-routers to the MPLS backbone are provisioned with less bandwidth than the primary links, the Oracle Communications Session Border Controller can:

- Detect remote link failures
- Trigger bandwidth updates at the realm level when using backup links
- Detect remote link failback to primary

To do so, the Oracle Communications Session Border Controller monitors the primary link status using ICMP echo requests (or pings). It issues the pings at regular intervals, forming a heartbeat mechanism. The CE-router can respond to these pings on the primary link, which is represented by the WAN IP address. When this link fails over, the backup link assumes the same WAN IP address but is not responsive to the pings. This way, the Oracle Communications Session Border Controller determines failover when the ICMP ping fails.

When there is an ICMP ping failure, the Oracle Communications Session Border Controller adjusts the realm's available bandwidth pool from its maximum bandwidth setting to its fallback setting. If the fallback amount is less than the maximum amount, it is possible for the Oracle Communications Session Border Controller to start rejecting calls. It does so until enough calls are released to free adequate bandwidth to stay under the fallback limit and still accept calls.

Bandwidth CAC Fallback Based on ICMP Failure Configuration

You can set up ICMP heartbeats and fallback bandwidth pools in the realm configuration. Leaving the **icmp-detect-multiplier**, **icmp-advertisement-interval**, or **icmp-target-ip** parameters blank or set to zero turns the feature off.

To enable bandwidth CAC fallback based on ICMP failure:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **icmp-detect-multiplier**—Enter the multiplier you want to use when determining how long to send ICMP pings before considering a target unreachable. This number multiplied by the time you set for the **icmp-advertisement-interval** determines the length of time. For example, if you set this parameter to 10 and the advertisement interval to 20, the Oracle Communications Session Border Controller will send ICMP pings for 200 seconds before declaring the target unreachable.
5. **icmp-advertisement-interval**—Enter the time in seconds between ICMP pings the Oracle Communications Session Border Controller sends to the target. The default is 0.
6. **icmp-target-ip**—Enter the IP address to which the Oracle Communications Session Border Controller should send the ICMP pings so that it can detect when they fail and it needs to switch to the fallback bandwidth for the realm. There is no default.
7. **fallback-bandwidth**—Enter the amount of bandwidth you want available once the Oracle Communications Session Border Controller has determined that the target is unreachable.

   If the fallback amount is less than the **max-bandwidth** value, the Oracle Communications Session Border Controller might start to reject calls. It does so until enough calls are released to free adequate bandwidth to stay under the fallback limit and still accept calls.
8. Save and activate your configuration.

Bandwidth CAC for Aggregate Emergency Sessions

You can configure the maximum amount of bandwidth on your Oracle Communications Session Border Controller you want used specifically for priority (emergency) calls in the realm configuration's **max-priority-bandwidth** parameter. You set this limit on a per-realm basis, and the limit is enforced for nested realms. Setting a bandwidth limit specifically for priority calls allows the Oracle Communications Session Border Controller to reject calls exceeding the threshold, and also to accept calls that exceed the bandwidth limit for non-priority calls (set in the **max-bandwidth** parameter).

The bandwidth limit for emergency calls operates in conjunction with the bandwidth limits you can set for all other types of calls. When an emergency call comes in, the Oracle Communications Session Border Controller checks the non-priority bandwidth limit. If bandwidth is sufficient, the call goes through and the Oracle Communications Session Border Controller decrements the bandwidth used from the pool of the amount available.

However, if a priority call exceeds the **max-bandwidth** setting, the Oracle Communications Session Border Controller checks the **max-priority-bandwidth** parameter. If is it within the limit for priority calls, the system allows the call and decrements the amount of used bandwidth from what is available.

When there is not enough bandwidth in either the priority or non-priority pool, the Oracle Communications Session Border Controller rejects the call with the corresponding error code and reason phrase.

Any bandwidth subtracted from either pool during a session is returned to that pool as soon as the session ends.

Bandwidth CAC for Aggregate Emergency Sessions Configuration

You configure bandwidth CAC for priority calls on a per-realm basis.

> **Note:**
>
> This parameter honors the hierarchy of nested realms if you have them configured.

To enable bandwidth CAC for aggregate emergency sessions:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **max-priority-bandwidth**Enter the amount of bandwidth you want to want to use for priority (emergency) calls. The system first checks the **max-bandwidth** parameter, and allows the call if the value you set for priority calls is sufficient. If there is not enough priority and non-priority bandwidth allotted for an incoming call, the Oracle Communications Session Border Controller rejects it.

   This parameter defaults to 0. You can enter any value between 0 and 999999999.

5. Save and activate your configuration.

Admission Control for Session Agents

This section explains how to configure session agents for admission control.

Session Agents Admission Control Configuration

To use admission control based on session rate, you need to configure session agent session rate constraints.

To configure session rates:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Enable session agent constraints and then configure the parameters related to session capacity or session rate to set admission control.

   constraints—Enable this parameter. From here you can either configure admission control based on session capacity, session rates, or both. The default value is enabled. The valid values are:

   - enabled | disabled

5. **max-sessions**—Set the maximum number of sessions (inbound and outbound) allowed by the session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—4294967295
6. **max-inbound-sessions**—Enter the maximum number of inbound sessions allowed from this session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
7. **max-outbound-sessions**—Enter the maximum number of concurrent outbound sessions (outbound from the Oracle Communications Session Border Controller) that are allowed from this session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—4294967295

> **Note:**
>
> The number you enter here cannot be larger than the number you entered for max-sessions.

8. **max-burst-rate**—Enter a number to set how many SIP session invitations or H.323 SETUPs this session agent can send or receive (per second) within the configured burst rate window value. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—4294967295

   For example, with a max-burst-rate of 20 and a burst-rate-window of 10, the Oracle Communications Session Border Controller permits 200 sessions within the first 10 seconds and then reject all new sessions until it exits constraint mode.
9. **max-inbound-burst-rate**—Enter the maximum burst rate (number of session invitations per second) for inbound sessions from this session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
10. **max-outbound-burst-rate**—Enter the maximum burst rate (number of session invitations per second) for outbound sessions to this session agent.The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
11. **max-sustain-rate**—Enter a number to set the maximum rate of session invitations (per second) this session agent can send or receive within the current window. The default value is zero (**0**). The valid range is:
    - Minimum—zero (0)
    - Maximum—4294967295

    The number you enter here must be larger than the number you enter for **max-burst-rate**.

    For the sustained rate, the Oracle Communications Session Border Controller maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.

    For example, if you enter a value of 50 here and a value of 36 (seconds) for the sustain rate window constraint, no more than 1800 session invitations can arrive at or leave from the session agent in any given 36 second time frame (window). Within that 36 second window, sessions over the 1800 limit are rejected.
12. **max-inbound-sustain-rate**—Enter the maximum sustain rate (of session invitations allowed within the current window) of inbound sessions from this session agent. This value should be larger than the **max-inbound-burst-rate** value. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
13. **max-outbound-sustain-rate**—Enter the maximum sustain rate (of session invitations allowed within the current window) of outbound sessions to this session agent. This value

should be larger than the **max-outbound-burst-rate** value. The default value is zero (**0**). The valid range is:

- Minimum—0
- Maximum—999999999

14. **burst-rate-window**—Enter a number to set the burst window period (in seconds) that is used to measure the burst rate. The term window refers to the period of time over which the burst rate is computed. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—4294967295
15. **sustain-rate-window**—Enter a number to set the sustained window period (in seconds) that is used to measure the sustained rate. The default value is zero (**0**), which disables the functionality. The valid range is:
    - Minimum—10
    - Maximum—4294967295

    The value you set here must be higher than or equal to the value you set for the burst rate window.

> **Note:**
>
> If you are going to use this parameter, you must set it to a minimum value of 10.

The following example shows session agent constraints that are enabled and the session capacity parameters have been configured. Other session agent parameters have been omitted for brevity.

```
session-agent
constraints                    enabled
max-sessions                   355
max-inbound-sessions          355
max-outbound-sessions          355
```

The following example shows session agent constraints are enabled and the session rate parameters have been configured. Other session agent parameters have been omitted for brevity.

```
session-agent
max-burst-rate                 0
max-inbound-burst-rate         10
max-outbound-burst-rate        1
max-sustain-rate               3000
max-inbound-sustain-rate       0
max-outbound-sustain-rate      0
burst-rate-window              0
sustain-rate-window            0
```

Realm Bandwidth Configuration

To configure admission control based on bandwidth, you set the max and min bandwidth parameters in the realm configuration.

To configure realm bandwidth:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. Configure the maximum bandwidth.

   max-bandwidth—Enter a number that sets the maximum bandwidth for dynamic flows to/from the realm in kilobits (Kbps) per second. The default value is zero (0). The valid range is:

   - Minimum—0
   - Maximum—4294967295

   The following example shows the maximum bandwidth for the realm has been configured. All other realm parameters have been omitted for brevity.

```
realm-config
max-bandwidth                  64000
```

SIP Admission Control Configuration

You can configure the registered endpoint to accept and process requests from SIP realms. If a request does not meet the criteria of the option you choose here, it is rejected with a 403 (Forbidden) response.

To configure admission control:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. Type **sip-ports** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(sip-interface)# sip-port
ORACLE(sip-port)#
```

5. Set the criteria for admission control.

   allow-anonymous—Enter the anonymous connection mode you want applied when SIP requests are processed. The default value is all.

   The following are valid values:

   - **all**—No ACL is applied and all anonymous connections are allowed.
   - **agents-only**—Only requests from configured session agents are processed. The Oracle Communications Session Border Controller responds to all other requests with a forbidden response.
   - **realm-prefix**—Only requests from session agents and addresses matching the realm's address prefix are processed. All other requests are rejected with a 403 (Forbidden) response.
   - **registered**—Only requests from session agents and registered endpoints are processed. REGISTER allowed from any endpoint.
   - **registered-prefix**—Only requests from session agent and registered endpoint addresses that match the realm's realm prefix are processed.

     The following example shows the **allow-anonymous** parameter that has been configured to allow only requests from session agents and registered endpoints. All other session agent parameters following the **allow-anonymous** parameters are omitted for brevity.

```
sip-port
                address
                port                            5060
                transport-protocol              UDP
                allow-anonymous                 registered
```

Aggregate Session Constraints for SIP

You can set a full suite of session constraints and then apply them to a SIP interface. The session constraints configuration contains many of the same parameters as the session agent, so you can configure a group of constraints and then apply them to a SIP interface/

The SIP interface configuration's **constraint-name** parameter invokes the session constraint configuration you want to apply. Using the constraints you have set up, the Oracle Communications Session Border Controller checks and limits traffic according to those settings for the SIP interface. Of course, if you do not set up the session constraints or you do not apply them in the SIP interface, then that SIP interface will be unconstrained. If you apply a single session-constraint element to multiple SIP interfaces, each SIP interface will maintain its own copy of the session-constraint.

SIP interfaces now have two states: "In Service" and "Constraints Exceeded." When any one of the constraints is exceeded, the status of the SIP interface changes to Constraints Exceeded and remains in that state until the **time-to-resume** period ends. The session constraint timers that apply to the SIP interface are the time-to-resume, burst window, and sustain window.

Aggregate Session Constraints Configuration

This section shows you how to configure aggregate session constraints and then apply them to a SIP interface.

The session constraints configuration contains many of the same parameters as the session agent does; it also incorporates the changes to the session agent parameters that are described in this section.

To configure the session constraints:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-constraints** and press Enter.

   ```
   ORACLE(session-router)# session-constraints
   ```

4. **name**—Enter the name for this session constraints configuration; this is a unique identifier that you will use in the SIP interface when you want the session constraints applied there. This is a required parameter that has no default.
5. **state**—Enable this parameter to use these session constraints. The default value is **enabled**. The valid values are:
   - enabled | disabled
6. **max-sessions**—Enter the maximum sessions allowed for this constraint. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
7. **max-outbound-sessions**—Enter the maximum outbound sessions allowed for this constraint. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
8. **max-inbound-sessions**—Enter the maximum inbound sessions allowed for this constraint. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
9. **max-burst-rate**—Enter the maximum burst rate (invites per second) allowed for this constraint. This value should be the sum of the **max-inbound-burst-rate** and the **max-outbound-burst-rate**. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999

10. **max-sustain-rate**—Enter the maximum rate of session invitations per second allowed within the current window for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

    For the sustained rate, the Oracle Communications Session Border Controller maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.
11. **max-inbound-burst-rate**——Enter the maximum inbound burst rate (number of session invitations per second) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
12. **max-inbound-sustain-rate**—Enter the maximum inbound sustain rate (of session invitations allowed within the current window) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

    For the sustained rate, the Oracle Communications Session Border Controller maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.
13. **max-outbound-burst-rate**——Enter the maximum outbound burst rate (number of session invitations per second) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
14. **max-outbound-sustain-rate**—Enter the maximum outbound sustain rate (of session invitations allowed within the current window) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

    For the sustained rate, the Oracle Communications Session Border Controller maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.
15. **time-to-resume**—Enter the number of seconds after which the SA (Session Agent) is put back in service (after the SA is taken OOS (Out Of Service) because it exceeded some constraint). The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
16. **ttr-no-response**—Enter the time delay in seconds to wait before the SA (Session Agent) is put back in service (after the SA is taken OOS (Out Of Service) because it did not respond to the Oracle Communications Session Border Controller). The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

17. **in-service-period**—Enter the time in seconds that elapses before an element (like a session agent) can return to active service after being placed in the standby state. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
18. **burst-rate-window**—Enter the time in seconds that you want to use to measure the burst rate; the window is the time over which the burst rate is calculated, and is used for the over all burst rate as well as the inbound and outbound burst rates. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
19. **sustain-rate-window**—Enter the time in seconds used to measure the sustained rate; the window is the time over which the sustained rate is calculated, and is used for the over all sustained rate as well as the inbound and outbound sustained rates. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

Applying Session Constraints in a SIP Interfaces

In the SIP interface, there is a new parameter that allows you to use a set of session constraints for that interface; the parameter is called constraint-name.

To apply session constraints to a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. **constraint-name**—Enter the name of the session constraints configuration that you want to apply to this SIP interface. There is no default for this parameter.
5. Save and activate your configuration.

Configuring CAC Policing and Marking for non-Audio non-Video Media

In the media profile and the media policy configurations, the following values have been added for the **media-type** parameter:

- **application | data | image | text**

For the media policy, these new values apply to ToS marking.

Support for the AS Bandwidth Modifier

Two new parameters have been added to the media profile configuration:

- **sdp-bandwidth**—Enable or disable the use of the AS modifier in the SDP if the **req-bandwidth** and **sdp-rate-limit-headroom** parameters are not set to valid values in a corresponding media profile. The default value is **disabled**. The valid values are:
  - enabled | disabled
- **sdp-rate-limit-headroom**—Specify the percentage of headroom to be added while using the AS bandwidth parameter while calculating the **average-rate-limit** (rate limit for the RTP flow). The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—100

The following conditions apply to the use and application of these two new parameters:

- If the amount of required bandwidth is not specified in the media profile (**req-bandwidth**) for the media type in the m= line of the SDP, then the value specified in the AS modifier is used. The Oracle Communications Session Border Controller only uses the AS value if you set the new **sdp-bandwidth** to enabled.
- If the average rate limit value for RTP flows is not specified in the media profile (**average-rate-limit**) for the media type in the m= line of the SDP, then the value specified in the AS modifier is used. The system only uses the AS value if you set the new **sdp-bandwidth** to enabled. When calculating the average rate rate limit that it will use based on the AS modifier, the Oracle Communications Session Border Controller applies the percentage set in the **sdp-rate-limit-headroom** parameter.
- The Oracle Communications Session Border Controller uses the value specified in the AS modifier (if **sdp-bandwidth** is enabled, and **req-bandwidth** is set to 0) along with the **user-cac-bandwidth** value set in the realm configuration; this works the same way that the **req-bandwidth** parameter does.
- The system uses the value specified in the AS modifier (if **sdp-bandwidth** is enabled, and **req-bandwidth** is set to 0) along with the **max-bandwidth** value set in the realm configuration; this works the same way that the **req-bandwidth** parameter does.

Media Profile Configuration

To set any of the new media types in the media profile configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. **media-type**—Enter the media type that you want to use for this media profile. The valid values are:
   - **audio** | **video | application | data | image | text**
5. Save and activate your configuration.

   To set any of the new media types in the media policy configuration:
6. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

7. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

8. Type **media-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# media-policy
ORACLE(media-policy)#
```

9. **media-type**—Enter the media type that you want to use for this media profile. The valid values are:
   - **audio** | **video | application | data | image | text**
10. Save and activate your configuration.

AS Modifier and Headroom Configuration

To enable AS modifier use and establish the percentage of headroom to use:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# media-profile
ORACLE(media-profile)#
```

4. **sdp-bandwidth**—Enable this parameter to use the AS bandwidth modifier in the SDP. The default is **disabled**. Valid values are:
   - enabled | disabled
5. **sdp-rate-limit-headroom**—Specify the percentage of headroom to be added while using the AS bandwidth parameter while calculating the **average-rate-limit** (rate limit for the RTP flow). The default is **0**. The valid range is:
   - Minimum—0

- Maximum—100

6. Save and activate your configuration.

Offerless Bandwidth CAC for SIP

For SIP sessions offerless INVITEs (i.e., INVITEs that have no SDP offer), the Oracle Communications Session Border Controller can reserved bandwidth and support the session if you set up applicable media profile associations in the global SIP configuration. Otherwise, the Oracle Communications Session Border Controller terminates these sessions.

You configure support for offerless bandwidth CAC by setting up your global SIP configuration with the options parameters set to **offerless-media-bw-profiles**. The option takes multiple media profile names as values to apply when treating offerless INVITEs. When such an INVITE arrives and your configuration supports this option, the Oracle Communications Session Border Controller checks and reserves bandwidth for the session. If there is insufficient bandwidth to reserve, the Oracle Communications Session Border Controller terminates the session. Otherwise, the actual SDP negotiation takes place unaffected while the Oracle Communications Session Border Controller forwards the offerless INVITE. Once the negotiation completes, the Oracle Communications Session Border Controller updates bandwidth reservation.

If the called party's actual bandwidth needs exceed available bandwidth, the Oracle Communications Session Border Controller must terminate the session, even if the session is ringing or answered. To minimize this occurrence as much as possible, you should consider all case scenarios when you select media profiles to use with the **offerless-media-bw-profiles** option.

Offerless Bandwidth CAC for SIP Configuration

To configure offerless bandwidth CAC for SIP:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **options——Your entry will look like this:**

```
ORACLE(sip-config)# options offerless-bw-media-profiles=PCMU,G729
```

You can you the plus sign (+) and the minus sign (-) to add and remove values from the options list.

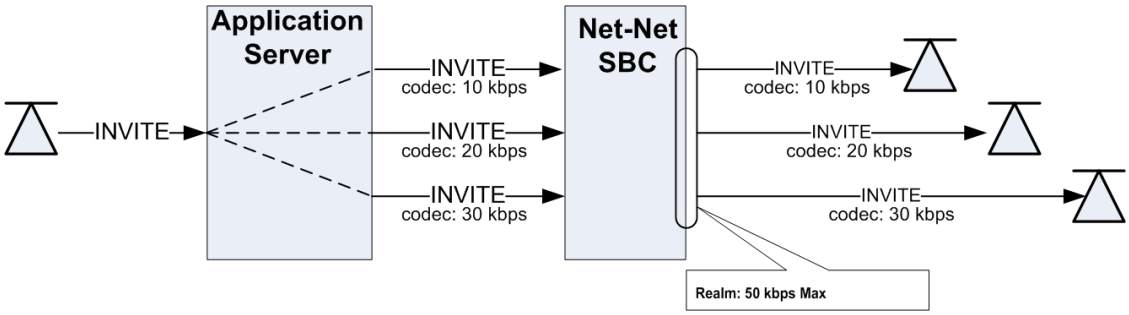
5. Type **done** and continue

Shared CAC for SIP Forked Calls

A forked call is one which has multiple INVITEs for the same call. For example, if an Application Server in the provider core network forks a call attempt, the application server sends several INVITEs for the same call toward the SBC. Each INVITE is destined for a unique device that belongs to the same user. Ideally, that user will only answer one device. The SBC treats each INVITE as a unique call request.

By default, each of the multiple INVITE forks are checked against CAC bandwidth limits, and thus they each consume bandwidth resources when they are received, even though only one of the forks will succeed in establishing a permanent session. Therefore, for many operators the CAC behavior of the SD is too restrictive and results in rejected call attempts which should have been allowed.

The following diagram shows a forked call scenario. The total bandwidth counted against the realm is 60 kbps. If the realm has a bandwidth ceiling of 50 kbps, one of the INVITEs will be rejected.



You can, however, enable the system to enforce CAC limits only once for SIP forked calls as long as the calls are identified as such, meaning that they will use the same bandwidth resources. The SBC counts the forked call's most bandwidth-hungry codec at the time it arrives at the SBC. In the above diagram, with shared bandwidth for forked calls enabled, the SBC counts 30 kbps against the realm's total bandwidth after that INVITE arrives, even after the first two INVITES have passed into the final realm.

Bandwidth Sharing Scenarios

The following table summarizes how bandwidth would be shared given certain ingress and egress realms with this feature enabled. Realms A and C are call ingress realms.; realms B and D are egress realms. For the bandwidth to be shared, Call A and Call B must have the same forked Call-ID in the P-Multiring-Correlator header and be entering or exiting the Oracle Communications Session Border Controller on the same realm.

		CALL A			
		Ingress Realm A	Egress Realm B	Ingress Realm C	Egress Realm D
CALL B	Ingress Realm A	bandwidth shared	N/A	bandwidth not shared	N/A
	Egress Realm B	N/A	bandwidth shared	N/A	bandwidth not shared
	Ingress Realm C	bandwidth not shared	N/A	bandwidth shared	N/A
	Egress Realm D	N/A	bandwidth not shared	N/A	bandwidth shared

Bandwidth Sharing Configuration

To enable bandwidth sharing of forked calls, set the **forked-cac-bw** parameter in the SIP profile configuration to **shared**. Although there are other parameters available in the SIP profile configuration, you only have to set the **name** and the **forked-cac-bw** values to use this feature.

After you set up the SIP profile, you apply it to a realm, SIP interface, or session agent.

Configuring a SIP Profile

The SIP profile is an element in the ACLI's **session-router** path, and you can configure multiple SIP profiles.

To configure a SIP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-profile** and press Enter.

   ```
   ORACLE(session-router)# sip-profile
   ORACLE(sip-profile)#
   ```

4. **name**—Enter a name for this SIP profile configuration. This parameter is blank by default, and it is required. You will need the SIP profile's **name** when you want to apply this profile to a realm, SIP interface, or SIP session agent.
5. **forked-cac-bw**—Set this parameter to **shared** if you want forked sessions to share bandwidth resources, or set it to **per-session** if you want bandwidth to be counted for each session individually. There is no default for this parameter, and leaving it blank means:
   - For an ingress session agent without a SIP profile or with a SIP profile where the forked CAC mode is blank, the Oracle Communications Session Border Controller will reference the associated realm.
   - For an ingress realm without a SIP profile or with a SIP profile where the forked CAC mode is blank, the Oracle Communications Session Border Controller will reference the associated SIP interface.
   - For an ingress SIP interface without a SIP profile or with a SIP profile where the forked CAC mode is blank, the Oracle Communications Session Border Controller will not perform bandwidth sharing for forked calls.
6. Save your work.

Applying a SIP Profile

Once you have configured one or more SIP profiles, you can apply them to realms, SIP interfaces, and SIP session agents. As an example, this section shows you how to apply a SIP profile to a SIP interface. But the parameter name is the same in these configurations:

- realm-config
- sip-interface
- session-agent
  To apply a SIP profile to a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **sip-profile**—Enter the name of SIP profile configuration that includes the forked-cac-bandwidth parameter configured.
5. Save your work.

RADIUS Accounting Support

VSA 171, Acme-Session-Forked-Call-Id, is part of the Oracle RADIUS dictionary. The VSA is a string value, and appears as the header-value without the header parameters from the P-Multiring-Correlator header for a session identified as part of a forked call.

Monitoring

Using the ACLI **show sipd forked** command, you can display the total number of forked sessions the Oracle Communications Session Border Controller received and the total number it rejected. The Oracle Communications Session Border Controller counts forked sessions when it receives a dialog-creating INVITE and is enabled to shared bandwidth. Further, it counts as forked all session with the P-Multiring-Correlator header.

```
ORACLE# show sipd forked
11:19:20-116
Forked Sessions               ---- Lifetime ----
                       Recent      Total  PerMax
Forked Sessions             0          0       0
Forked Sessions Rej         0          0       0
```

Conditional Bandwidth CAC for Media Release

The Oracle Communications Session Border Controller supports conditional call admission control (CAC) using the SIP profile configuration. With this feature enabled, you can allow the conditional admission of SIP calls that could potentially have their media released instead of risking the possible rejection of those calls due to internal bandwidth limits.

About Conditional Bandwidth CAC for Media Release

The Oracle Communications Session Border Controller performs bandwidth CAC for SIP per realm, for each Address of Record (AoR) or IP address. The system checks bandwidth limits based on the codecs listed in SDP. If a new SIP INVITE contains codecs in an SDP message that exceed bandwidth available for a given resource, the system rejects that INVITE. This check occurs both on the ingress and egress sides of a call, and both sides must have enough available resources to support the call for it to be admitted.

In the case of calls where media is released, the Oracle Communications Session Border Controller does not count bandwidth consumed by the call. However, this exemption is not given until the media is actually released—and media release conditions are unknown at the time SIP INVITE is admitted. This is because an INVITE received on one side of the Oracle Communications Session Border Controller is only media-released when that INVITE is routed back through the Oracle Communications Session Border Controller as a hairpin or other multi-system media release. So there has to be enough bandwidth for the initial INVITE; otherwise, and even if the INVITE is a candidate for media release, it will be rejected.

When there is a significant volume of such calls—ones that are candidates for media release, but cannot be admitted because of CAC limits—it becomes important to admit them so long as they truly end in media release. This feature thus allows admission of SIP calls that might otherwise be rejected because of bandwidth limitations when the far-end of the call causes media to be released.

Details and Conditions

This feature applies in a two system scenario. In order to track a call as a candidate for provisional media release, the access-side Oracle Communications Session Border Controller adds a Require: header with an option tag to the INVITE or UPDATE message on egress. The option tag is configurable in the sip config option. The default is com.acmepacket.cac .

The following sections describe when the SIP INVITE or SIP UPDATE are:

- initially received by the Oracle Communications Session Border Controller
- received by the second Oracle Communications Session Border Controller

INVITEs UPDATEs Initially Received By Oracle Communications Session Border Controller

When the Oracle Communications Session Border Controller first receives an INVITE or UPDATE message, it considers if it should be admitted provisionally or rejected outright due to CAC bandwidth constraints. If the INVITE or UPDATE is admitted provisionally, a Require: header is inserted on egress from the system.

The Oracle Communications Session Border Controller inserts the Require header on egress under these conditions:

- It receives an INVITE / UPDATE with no or a non-matching Require header.
- The **egress conditional cac admit** parameter in the SIP profile on the egress realm, SIP interface, session agent is set to enabled in the egress realm
- The request would otherwise be rejected because of current bandwidth CAC limits in the ingress OR egress realms
- The call is a candidate for media-release in the ingress realm

A call is considered a candidate for media-release when the ingress realm has any of these parameters set to disabled:

- **mm-in-realm**
- **mm-in-network**
- **mm-same-ip**
- **mm-in-system**

INVITEs UPDATEs Received by Second SBC

The second Oracle Communications Session Border Controller receives the INVITE or UPDATE with the newly inserted Require: header. Standard SIP convention indicates that if the UAS receiving the request does not know how to handle the Require header, the request should be rejected.

When the following three conditions are met, the INVITE is permitted into the system for processing:

- The **ingress conditional cac admit** in the SIP profile on the ingress realm, SIP interface, session agent parameter is set to enabled
- The **con-cac-tag** sip config option is configured to the same value as the received Require header's option tag
- The call is a candidate for media-release

The call is considered a candidate for media-release on the second system (indicated by the **ingress conditional cac admit** parameter is set to enabled) when either the ingress or egress realms have any of these parameters set to disabled:

- **mm-in-realm**
- **mm-in-network**
- **mm-same-ip**
- **mm-in-system**

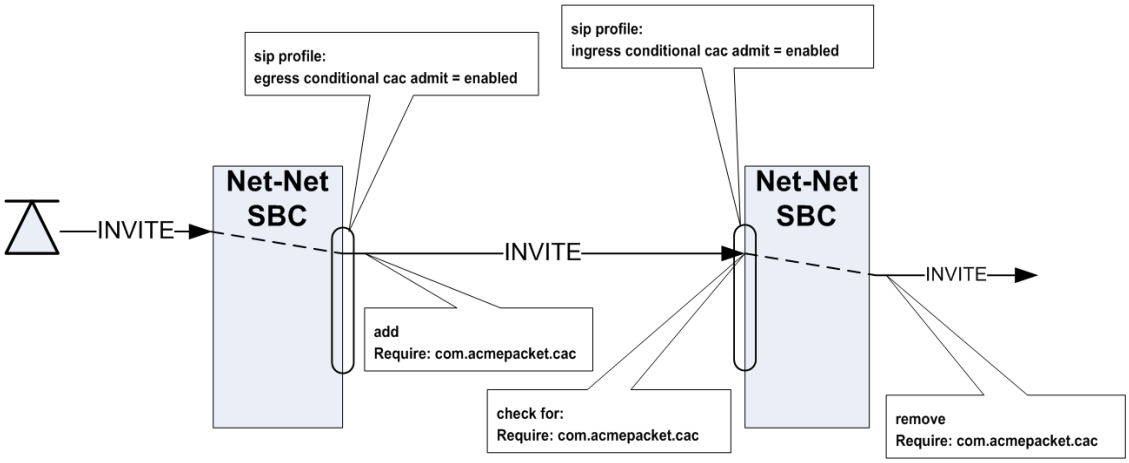
and the following parameter is set to enabled:

- msm-release

If the call, as received by the second system is not considered a candidate for release, the INVITE or UPDATE is failed with a 503 Insufficient Bandwidth message.

After the INVITE has been processed by the Oracle Communications Session Border Controller, the Require: header is removed upon egress from the system.

The following diagram shows the two-system scenario:



Conditional Admission with Per-user CAC

In the event that the per-user CAC feature is also being used, and per-user CAC bandwidth is exceeded, the Oracle Communications Session Border Controller also uses this option tag mechanism. However, if the per-user CAC implementation does count bandwidth regardless of media-release, then the Oracle Communications Session Border Controller will reject calls exceeding the per-user CAC limits when it receives them.

On the second system, when the per-user CAC feature is being used, the Oracle Communications Session Border Controller will perform the same option tag mechanism based on if the **ingress conditional cac admit** parameter is enabled.

Conditional Bandwidth CAC Configuration

You enable this feature by first configuring a SIP profile, and then applying the profile to any of these:

- realm
- SIP interface
- SIP session agent

SIP Profile Configuration

The SIP profile is an element in the ACLI's **session-router** path, and you can configure multiple SIP profiles. Though this configuration contains additional parameters, you do not have to use them for the conditional bandwidth CAC for media release.

To configure a SIP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-profile** and press Enter.

```
ORACLE(session-router)# sip-profile
ORACLE(sip-profile)#
```

4. **name**—Enter a name for this SIP profile configuration. This parameter is blank by default, and it is required. You will need the SIP profile's **name** when you want to apply this profile to a realm, SIP interface, or SIP session agent.
5. **ingress-conditional-cac-admit**—Set this parameter to enabled to process an INVITE with a Require tag as received on an ingress interface. You can set this parameter to disabled if you do not want to use this feature on the ingress side. There is no default for this parameter.
6. **egress-conditional-cac-admit**—Set this parameter to enabled if you want to use conditional bandwidth CAC for media release for calls that are first received by this system. This results in option tags being inserted on the INVITE's egress if the conditional CAC conditions are met. You can set this parameter to disabled if you do not want to use this feature. There is no default for this parameter.
7. Save your work.

Applying a SIP Profile

Once you have configured one or more SIP profiles, you can apply them to realms, SIP interfaces, and SIP session agents. As an example, this section shows you how to apply a SIP profile to a SIP interface. But the parameter name is the same in these configurations:

- realm-config
- sip-interface
- session-agent

To apply a SIP profile to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **sip-profile**—Enter the name of SIP profile configuration you want to use for conditional bandwidth CAC for media release for this SIP interface. This value is blank by default, but it must be the value of the **name** parameter from a valid SIP profile.
5. Save your work.

Configuring Require Header Option Tag

You may change the Require: header's option tag from the default com.acmepacket.cac to one of your own choosing. Remember that both systems' option tags must match exactly.

To configure the Require: header's option tag:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ```

4. Use the ACLI **select** command so that you can work with the SIP configuration.

   ```
   ORACLE(sip-config)# select
   ```

5. **options**—Set the options parameter by typing +**options**, a Space, the option name **con-cac-tag=** your-new-tag, and then press Enter.

   ```
   ORACLE(sip-config)# options +con-cac-tag=com.test.cac
   ```

6. Save your work.

CAC Utilization Statistics via SNMP

The Oracle Communications Session Border Controller allows you to retrieve information on current session utilization and burst rate as a percentage of their configured maximums on per session-agent and/or realm basis. The Oracle Communications Session Border Controller uses the configured max-session and max-burst-rate settings in conjunction with a percentage formula to calculate this value. The system also uses an ACLI configuration setting to establish the threshold at which trap and trap clear messages are sent from the SNMP agent to the configured manager(s).

The user must load the MIB version associated with this software version on all pertinent SNMP managers to query these CAC utilization (occupancy) values and interpret the traps. In addition, the user must configure the threshold at which the system generates the CAC utilization trap. Note that the corresponding clear trap uses the same threshold setting, sending the clear trap when utilization falls below 90% of the threshold.

SNMP Get for CAC Utilization

Using a MIB browser, the user can query the current percentage utilization values for both max-session and max-burst-rate for any session-agent or realm. The calculations for these utilization levels are:

- Session utilization level = (current session count * 100) / max-sessions

- Burst rate utilization level = (current burst rate * 100) / max-burst-rate

The MIB objects associated with these statistics are parallel for session agent and realm and include a table to contain the objects, an object associating the objects containing the values with the applicable table, and objects containing the values themselves. These objects are listed below.

The MIB objects containing CAC utilization data for Session Agents are listed below.

The object establishing the statistics table for session agent CAC utilization follows:

```
--apSip Session Agent Connection Admission Control Stats Table
apSipSaCacStatsTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF ApSipSaCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "SIP Session Agent Connection Admission Control Stats Table."
        ::= { apSipMIBTabularObjects 5 }
```

The object establishing the session agent CAC utilization statistics objects follows:

```
apSipSaCacStatsEntry OBJECT-TYPE
        SYNTAX          ApSipSaCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "Connection Admission Control Statistics."
        AUGMENTS { apSipSessionAgentStatsEntry }
     ::= { apSipSaCacStatsTable 1 }
```

The session agent CAC utilization statistics values include:

```
ApSipSaCacStatsEntry ::= SEQUENCE {
    apSipSaCacSessionUtilLevel                Gauge32,
    apSipSaCacBurstRateUtilLevel              Gauge32
}
```

The above objects, specifying the CAC utilization value for sessions and burst rate utilization for session agents include:

```
apSipSaCacSessionUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
    STATUS        current
    DESCRIPTION
        "Current session utilization level."
    ::= { apSipSaCacStatsEntry 1 }


apSipSaCacBurstRateUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
```

```
    STATUS        current
    DESCRIPTION
        "Current burst rate utilization level."
    ::= { apSipSaCacStatsEntry 2 }
```

The MIB objects containing CAC utilization data for Realms are listed below.

The object establishing the statistics table for realm CAC utilization follows:

```
--apSig Realm Connection Admission Control Stats Table
apSigRealmCacStatsTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF ApSigRealmCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "Realm Connection Admission Control Stats Table."
        ::= { apSipMIBTabularObjects 6 }
```

The object establishing the realm CAC utilization statistics objects follows:

```
apSigRealmCacStatsEntry OBJECT-TYPE
        SYNTAX          ApSigRealmCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "Connection Admission Control Statistics."
        AUGMENTS { apSigRealmStatsEntry }
     ::= { apSigRealmCacStatsTable 1 }
```

The session agent CAC utilization statistics values include:

```
ApSigRealmCacStatsEntry ::= SEQUENCE {
    apSigRealmCacSessionUtilLevel              Gauge32,
    apSigRealmCacBurstRateUtilLevel            Gauge32
}
```

The above objects, specifying the CAC utilization value for sessions and burst rate utilization for realms include:

```
apSigRealmCacSessionUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
    STATUS        current
    DESCRIPTION
        "Current session utilization level."
    ::= { apSigRealmCacStatsEntry 1 }

apSigRealmCacBurstRateUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
    STATUS        current
    DESCRIPTION
```

```
        "Current burst rate utilization level."
    ::= { apSigRealmCacStatsEntry 2 }
```

CAC Utilization Traps

The Oracle Communications Session Border Controller can issue a trap when either the value of max-session or CAC burst rate exceeds a configured value. The system only sends one trap when the threshold is exceeded. When the value falls back under 90% of this threshold, the Oracle Communications Session Border Controller sends a clear trap.

You configure the value that triggers these traps as a percentage of the max-session and max-burst-rate settings configured for the applicable session agent and/or realm. The system uses the same setting to specify when to send both the sessions and burst rate traps. The name of this parameter is the **cac-trap-threshold**.

For realms, you configure a **session-constraint** element with the **cac-trap-threshold** setting and apply that session constraint to the realm. For a session agent however, you configure the **cac-trap-threshold** directly within the session agent's configuration.

The syntax for the command is the same within session constraint and session agent configurations.

cac-trap-threshold[0-99]

You must express the value as a number less than 100. There is no default setting; the system does not generate a trap if you have not configured this setting.

The apSipCACUtilAlertTrap identifies the threshold exceeded on a per-element and per-value (session count or burst rate) for each trap, including:

- apSipSaCacSessionUtilLevel
- apSipSaCacBurstRateUtilLevel
- apSipRealmCacSessionUtilLevel
- apSipRealmCacBurstRateUtilLevel

CAC utilization threshold trap on a session agent configuration

The CAC utilization threshold causes the system to generate a trap when session count or CAC max-burst-rate exceeds the configured percentage value of these values maximums. This setting is available within a session agent's configuration.
To configure the CAC trap threshold on a session agent, follow the procedure below.

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. Select the **session-agent** object to edit.

```
ORACLE(session-agent)# select
<hostname>:
1: 192.168.100.101:1813
```

```
selection: 1
ORACLE(session-agent)#
```

3. **cac-trap-threshold**—Set the threshold when reached, expressed as a percentage of **max-sessions**, when the CAC trap is sent.
4. Type **done** to save your configuration.

Configuring the CAC Utilization Thresholds - **realm**

To configure the CAC trap threshold on a realm or sip interface, create a session constraint object and apply it to your realm, as shown below.

1. Use the following sequence to navigate to session constraint elements.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-constraints
ORACLE(session-constraints)#
```

2. Select or create the desired session constraint element.

```
ORACLE(session-constraints)#name trap-at-90-percent
```

3. Configure the desired value for **cac-trap-threshold** expressed as a percentage value, such as 90%, as follows.

```
ORACLE(session-constraints)#cac-trap-threshold 90
```

4. Navigate to the realm-config to which you want to apply the session constraint.

```
ORACLE(realm-config)#session-constraint trap-at-90-percent
```

5. Execute the **done** and **exit** commands.
6. Save and activate your configuration.

About QoS Reporting

This section describes the Oracle Communications Session Border Controller QoS reporting. QoS reporting provides you with real-time evaluation of network and route performance. It lets you contrast internal domain and external domain performance and facilitates SLA verification and traffic engineering. Oracle Communications Session Border Controller QoS reporting is a measurement tool that collects statistics on Voice over IP (VoIP) call flows for SIP and H.323. To provide information, the Oracle Communications Session Border Controller writes additional parameters to the Remote Authentication Dial-in User Service (RADIUS) call record. To provide information, the Oracle Communications Session Border Controller writes additional parameters to the Remote Authentication Dial-in User Service (RADIUS) call record and Historical Data Recording (HDR) records.

You can use QoS statistics for SLA customer reporting, fault isolation, SLA verification, and traffic analysis. The Oracle Communications Session Border Controller employs specialized hardware to inspect Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol (RTCP) flows while maintaining wire-speed packet forwarding. QoS metrics are collected and reported on a per-session and per call-leg basis. These metrics are reported through real-time RADIUS records along with call accounting data.

Conflicting rFactor Data and MOS Scores

When comparing r-factor statistics from the **show sipd realm** command with other rFactor information, the output may be deceptively low. For example, you may evaluate QoS reporting within CDRs on a per-call basis and find that the rfactor data within show **show sipd realm** lower. The rfactor row in show sipd realm presents rfactor information for all flows in that realm, which is larger than the number of calls. This detail applies to all output tools that present **show sipd realm**, including CLI, HDR and SNMP.

Consider as an example a call that goes on hold, which would generate signaling that includes SDP and therefore Add flows to that realm. CDR data does not include these extra flows to calculate rFactor data. The discrepancy between this data is because the system tracks rFactor data for **show sipd realm** using all flows added to a realm and CDR rFactor data as a weighted average per call.

Overview

When a conversation is established between two endpoints, two flows are present in each direction:

- RTP flow carries traffic between endpoints with a predictable packet arrival rate. The packets headers have sequence numbers that are used to determine whether packets are missing or lost.
- RTCP flow carries information about the RTP flow and keeps a different record. The RTCP packets contain timestamps based on Network Time Protocol (NTP).

QoS Statistics

Reported QoS data includes the following per-flow statistics:

- RTP and RTCP lost packets—Count of lost packets for both RTP and RTCP based on comparing the sequence numbers since the beginning of the call or the last context memory poll.
- RTP and RTCP average jitter—Incremental number of packets for both RTP and RTCP that have been used to generate the total and max jitter since the beginning of the call or the last context memory poll. The incremental accumulated jitter (in milliseconds) over all the packets received.
- RTP and RTCP maximum jitter—Maximum single jitter value (in milliseconds) for both RTP and RTCP from all the packets since the beginning of the call or the last context memory poll.
- RTCP average latency—Number of RTCP frames over which latency statistics have been accumulated and the incremental total of latency values reported since the beginning of the call or the last context memory poll.
- RTCP maximum latency—Highest latency value measured since the beginning of the call or the last context memory poll.
- RTP packet count
- RTP bytes sent and received
- RTCP lost packets—RTP lost packets reported in RTCP packets.
- ATP lost packets—Lost packets determined by monitoring RTP sequence numbers.
- R-Factor and MOS data—R-Factor and MOS data for the calling and called segments at the end of a session

Incremental QoS Updates

The Interim Quality of Service (QoS) Update setting provides a more granular view of voice quality for troubleshooting by providing updates in 10 second increments. Without the Interim QoS Update setting selected, the Oracle Communications Session Border Controller (SBC) probe provides an average Mean Opinion Score (MOS) only at the end of the call. A troubleshooter cannot see what occurred in other parts of the call. For example, suppose your employee or agent complains of poor voice quality that occurred in the middle of the call, but the average MOS score at the end of the call is 4.40. The troubleshooter might determine that the quality is acceptable, without knowing that the score in the middle of the call is 2.50. The Interim QoS Update setting provides MOS scores every ten seconds, and with more granular data to help troubleshooting efforts.

Standalone Operations Monitor (OM) probes, such as those that run OM software on Linux COTS servers, provide MOS scores in ten second time chunks. With the Interim QoS Update parameter enabled, the data presented in OM looks similar whether coming from an SBC probe, OM probe, or both. To set voice quality sampling in ten second increments, go to **system-config**, **comm-monitor** and enable **interim-qos-update**.

The SBC provides the following data, per ten second interval.

- start + end time of the stream
- IP 5-tuple information to correlate to SIP sessions
- correlation information if available
- SSRC of the RTP stream (to be checked)
- Codec type
- Codec change information (if codecs changed)

The SBC provides the following data, per ten second chunk.

- jitter
- min/avg/max
- packet loss
- # of packets received
- # of packets lost

The SBC delivers voice quality details, as follows:

- Per RTP stream.
- In ten second increments, where the increment starts on a full minute based on the NTP clock (not the start time of the stream).
- Intervals not covering the full ten seconds do not return a MOS value.

> **Note:**
>
> The comm-monitor VQ reports do not support disabling latching for a stream because the SBC does not have access to the stream source IP address. Latching may be globally disabled via the **media-manager** object or, dynamically disabled even when globally enabled in **media-manager**, for example, when a media for a session has been successfully negotiated but the source of the media flow changes.

RADIUS Support

All the QoS statistics go into the RADIUS CDR. If a RADIUS client is configured on the Oracle Communications Session Border Controller, any time a call occurs a record is generated and sent. Only Stop RADIUS records contain the QoS statistic information.

Only RADIUS Stop records contain QoS information. For non-QoS calls, the attributes appear in the record, but their values are always be zero (0). When you review the list of QoS VSAs, please note that “calling” in the attribute name means the information is sent by the calling party and called in the attribute name means the information is sent by the called party.

The following example shows a CDR that includes QoS data:

```
Wed Jun 13 18:26:42 2007
        Acct-Status-Type = Stop
        NAS-IP-Address = 127.0.0.100
        NAS-Port = 5060
        Acct-Session-Id = "SDgtu4401-c587a3aba59dcae68ec76cb5e2c6fe6f-v3000i1"
        Acme-Session-Ingress-CallId =
"8EDDDC21D3EC4A218FF41982146844310xac1ec85d"
        Acme-Session-Egress-CallId = "SDgtu4401-
c587a3aba59dcae68ec76cb5e2c6fe6f-v3000i1"
        Acme-Session-Protocol-Type = "SIP"
        Calling-Station-Id = ""9998776565"
<sip:9998776565@10.10.170.2:5060>;tag=2ed75b8317f"
        Called-Station-Id = "<sip:7143221099@10.10.170.2:5060>"
        Acct-Terminate-Cause = User-Request
        Acct-Session-Time = 7
        h323-setup-time = "18:24:36.966 UTC JUN 13 2007"
        h323-connect-time = "18:24:37.483 UTC JUN 13 2007"
        h323-disconnect-time = "18:24:44.818 UTC JUN 13 2007"
        h323-disconnect-cause = "1"
        Acme-Session-Egress-Realm = "peer"
        Acme-Session-Ingress-Realm = "core"
        Acme-FlowID_FS1_F = "localhost:65544"
        Acme-FlowType_FS1_F = "PCMA"
        Acme-Flow-In-Realm_FS1_F = "core"
        Acme-Flow-In-Src-Addr_FS1_F = 10.10.170.15
        Acme-Flow-In-Src-Port_FS1_F = 49156
        Acme-Flow-In-Dst-Addr_FS1_F = 10.10.170.2
        Acme-Flow-In-Dst-Port_FS1_F = 31008
        Acme-Flow-Out-Realm_FS1_F = "peer"
        Acme-Flow-Out-Src-Addr_FS1_F = 10.10.130.2
        Acme-Flow-Out-Src-Port_FS1_F = 21008
        Acme-Flow-Out-Dst-Addr_FS1_F = 10.10.130.15
        Acme-Flow-Out-Dst-Port_FS1_F = 5062
        Acme-Calling-RTCP-Packets-Lost_FS1 = 0
        Acme-Calling-RTCP-Avg-Jitter_FS1 = 15
        Acme-Calling-RTCP-Avg-Latency_FS1 = 0
        Acme-Calling-RTCP-MaxJitter_FS1 = 15
        Acme-Calling-RTCP-MaxLatency_FS1 = 0
        Acme-Calling-RTP-Packets-Lost_FS1 = 0
        Acme-Calling-RTP-Avg-Jitter_FS1 = 3
        Acme-Calling-RTP-MaxJitter_FS1 = 44
        Acme-Calling-Octets_FS1 = 957
```

```
Acme-Calling-Packets_FS1 = 11
Acme-FlowID_FS1_R = "localhost:65545"
Acme-FlowType_FS1_R = "PCMA"
Acme-Flow-In-Realm_FS1_R = "peer"
Acme-Flow-In-Src-Addr_FS1_R = 10.10.130.15
Acme-Flow-In-Src-Port_FS1_R = 5062
Acme-Flow-In-Dst-Addr_FS1_R = 10.10.130.2
Acme-Flow-In-Dst-Port_FS1_R = 21008
Acme-Flow-Out-Realm_FS1_R = "core"
Acme-Flow-Out-Src-Addr_FS1_R = 10.10.170.2
Acme-Flow-Out-Src-Port_FS1_R = 31008
Acme-Flow-Out-Dst-Addr_FS1_R = 10.10.170.15
Acme-Flow-Out-Dst-Port_FS1_R = 49156
Acme-Called-RTCP-Packets-Lost_FS1 = 0
Acme-Called-RTCP-Avg-Jitter_FS1 = 13
Acme-Called-RTCP-Avg-Latency_FS1 = 0
Acme-Called-RTCP-MaxJitter_FS1 = 21
Acme-Called-RTCP-MaxLatency_FS1 = 0
Acme-Called-RTP-Packets-Lost_FS1 = 0
Acme-Called-RTP-Avg-Jitter_FS1 = 0
Acme-Called-RTP-MaxJitter_FS1 = 3
Acme-Called-Octets_FS1 = 77892
Acme-Called-Packets_FS1 = 361
Acme-Firmware-Version = "C5.0.0"
Acme-Local-Time-Zone = "Time Zone Not Set"
Acme-Post-Dial-Delay = 110
Acme-Primary-Routing-Number = "sip:7143221099@10.10.170.2:5060"
Acme-Ingress-Local-Addr = "10.10.170.2:5060"
Acme-Ingress-Remote-Addr = "10.10.170.15:5060"
Acme-Egress-Local-Addr = "10.10.130.2:5060"
Acme-Egress-Remote-Addr = "10.10.130.15:5060"
Acme-Session-Disposition = 3
Acme-Disconnect-Initiator = 2
Acme-Disconnect-Cause = 16
Acme-SIP-Status = 200
Acme-Egress-Final-Routing-Number = "sip:7143221099@10.10.130.15:5060"
Acme-CDR-Sequence-Number = 14
Client-IP-Address = 172.30.20.150
Acct-Unique-Session-Id = "0832b03cd3a290b3"
Timestamp = 1181773602
```

Configuring QoS

This section explains how to configure QoS. To generate QoS metrics, you need to enable QoS for the realm of the originating caller. The ingress realm determines whether QoS is turned on for a specific flow.

> **Note:**
>
> If you run with QoS turned on one side only and disabled on the other you lose the ability to measure latency through the use of RTCP timestamps.

QoS Configuration

To enable QoS:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **qos-enable**—Enable this parameter. The default value is **disabled**.

> **Note:**
>
> You do not have to reboot the SBC after enabling this parameter.

Network Management Controls

The Oracle Communications Session Border Controller supports network management controls for multimedia traffic specifically for static call gapping and 911 exemption handling. These controls limit the volume or rate of traffic for a specific set of dialed numbers or dialed number prefixes (destination codes).

In TDM networks, automatic call/code gapping was developed as part of the advanced intelligent network (AIN) to enable network element load shedding based on destination number (DN) in case of overload. However, since there are as yet no standards for call/code gapping for next generation multimedia networks, the Oracle Communications Session Border Controller provides statically-provisioned network management controls.

To enable network management controls on your Oracle Communications Session Border Controller, you set up the ACLI **net-management-control** configuration and then enable the application of those rules on a per-realm basis. Each network management control rule has a unique name, in addition to information about the destination (IP address, FQDN, or destination number or prefix), how to perform network management (control type), whether to reject or divert the call, the next hop for routing, and information about status/cause codes. Details about the content of control rules and how to set them appear in the instructions and examples section.

When a SIP INVITE or an H.323 Setup for a call arrives at the Oracle Communications Session Border Controller on an ingress realm where network management controls have been enabled, the Oracle Communications Session Border Controller takes the following steps:

- It searches the network management rules you have created (which are stored in tables on the Oracle Communications Session Border Controller) for a control rule that best matches the newly-received call.
- If it does not find a matching control rule, the Oracle Communications Session Border Controller allows the call to proceed normally.
- If it finds a matching control rule, then the Oracle Communications Session Border Controller treats the call according to the specifics of the rule and the treatment method that it designates.

Matching a Call to a Control Rule

The Oracle Communications Session Border Controller uses the call classification key (specified by the **destination-identifier** parameter) to match calls so that it can apply control rules. The call classification key specifies information about the destination, which can be an IP address, an FQDN, a destination (called) number, or destination prefix. You configure the classification key as part of the control rule.

Matching is performed from left to right, starting at the left-most character. A wildcard matches any digit.

The Oracle Communications Session Border Controller compares the following information from the SIP INVITE or H.323 Setup for matching:

- SIP INVITE—User part of the Request URI, or the host part of the Request URI
- H.323 Setup—Q.931 Called Party Number IE

With Release 6.0, the Oracle Communications Session Border Controller now normalizes the user-part of the Request-URI prior to performing any matching for NMC based on the dialed number. A departure from this feature's prior implementation, this normalization strips out any of the visual-separator characters.

Note that normalization occurs only for NMC look-up purposes, and it does not alter the actual Request-URI. For previous releases, NMC rule matching based on the dialed number fails when the dialed number has visual separators or additional parameters such as: rn, npdi, cic, postd, etc. If multiple rules match an incoming call, then the Oracle Communications Session Border Controller gives first priority to destination number or the destination prefix. Next, it tries to match based on the IP address, and finally it looks to the domain (lowest priority).

Specifically, the Oracle Communications Session Border Controller supports the following:

- The user-part can contain escaped sequences that the Oracle Communications Session Border Controller normalizes to their unescaped representation. For examples %23(358)555.1234567 would be normalized to #3585551234567.
- The Oracle Communications Session Border Controller parses the user-part of the Request-URI up to the first semicolon (;). For example, the user-part in tel: +358-555-1234567;postd=pp22 will be +358-555-12134567.

Call Handling Determination

There are three types of control rules from which you can choose; each is a different way for the Oracle Communications Session Border Controller to handle calls matching the classification key:

- Call gap rate—Controls the maximum sustained rate of calls that match the classification key.

Using this type, the Oracle Communications Session Border Controller calculates the time since the last matching call. If that time is equal to or greater than the minimum time gap set in the control rule (i.e., it does not exceed the rate), then the call proceeds normally. If the call is less than the minimum time gap (i.e., it causes the call rate to be exceeded), then the Oracle Communications Session Border Controller either rejects or diverts the call.

To keep the call rate below the control value, the Oracle Communications Session Border Controller ensures a minimum call gap time between the matching calls. For example, if the control value is 10 calls per second, the minimum call gap time would be 0.1 second. And if a matching call were to arrive within a tenth of a second since the last matching call, then the Oracle Communications Session Border Controller applies the treatment method.

- Call gap percentage—Controls the percentage of calls matching the classification key you set for the control rule.
  When using this control rule type, the Oracle Communications Session Border Controller applies the treatment method to the percentage of matching calls (that you set in the value parameter) out the total number of matching calls it receives. For example, if you set the value parameter for the control rule to 50 and use this control type, the Oracle Communications Session Border Controller applies the treatment method to every other call it receives (or 50% of the calls it receives) that matches the classification key.

  Note that the Oracle Communications Session Border Controller cannot maintain exact percentages for the control value at all times, especially at system start-up when the number of incoming calls is small.
- Priority—Exempts calls to a destination (like 911) from local network management controls such as:
  - Session agent constraints
  - Bandwidth constraints (such as per-realm bandwidth)
  - External policy servers (requests are made to the policy server; calls are admitted and processed regardless of the decision or reachability of the policy server)
  - Per-user call admission control
  - CPU constraints
    The Oracle Communications Session Border Controller will not bypass licensing constraints, however.

Treatment Methods

You can choose from two different treatment methods:

- Call rejection—The Oracle Communications Session Border Controller rejects the call.
  - For SIP, the Oracle Communications Session Border Controller sends a response messages with the status code of the control rule. This response message also includes a Reason header with the Q.850 cause code that you configure as part of the control rule; it contains only the Q.850 cause code, and there is no reason-text included. For example:

    ```
    Reason: Q.850; cause=63
    ```

  - For H.323, the Oracle Communications Session Border Controller sends a releaseComplete message with the Q.850 cause code (that you configure as part of the control rule) of the control rule as the Q.931 Cause IE.

- Call diversion—The Oracle Communications Session Border Controller routes the call to the location you specify in the control rule's next hop parameter.
  Except for this routing, the call proceeds as normal. Local treatments such as number translation apply to the call, as do local controls such as licensing. Note the following:
  - If the next hop is an FQDN, the Oracle Communications Session Border Controller performs DNS queries to resolve the next hop to an IP address so that it can route the call properly.
    DNS queries only apply to pure SIP or IWF calls that originate in H.323 and are interworked to SIP.
  - If the next hop is a session agent group, the Oracle Communications Session Border Controller selects a session agent from the group according to the selection strategy you set for the group. Then the Oracle Communications Session Border Controller uses the IP address of the selected session agent.

Priority Call Exemption from Policy Server Approval

The Oracle Communications Session Border Controller now identifies priority calls and provides expedited treatment for them, even is these calls use associated realms for which there is an associated policy server handling bandwidth allocation. Instead of waiting for a response for the policy server, the Oracle Communications Session Border Controller immediately processes the call. When and if the policy server responds, the Oracle Communications Session Border Controller handles the response, but in all likelihood the priority calls has already been processed.

Enhanced Call Gapping

NMC provides flexibility by allowing a desired call-per-second (CPS) threshold to be achieved or surpassed by a predictable amount. Referred to as call gapping, this allows the Oracle Communications Session Border Controller to average the call rate and widen the period of a surge that would invoke NMC rules.

Without call gapping enabled, the NMC carries out a call gapping policy that monitors the arrival times between INVITEs, and then compares the arrival times to with the threshold. To enable this, you set the **type** parameter to gap-rate, and then configure the **value** parameter with the maximum sustained rate of calls. The threshold is equal to 1/gap-rate value. However, this implementation means that if two calls arrive simultaneously at the Oracle Communications Session Border Controller, one of them might be rejected or diverted if it exceeds the threshold and the control rule is applied. This is the case even when the sustained call rate does not exceed the control rule.

To resolve this, call gapping uses two parameters that form part of an calculation the Oracle Communications Session Border Controller performs for applying NMC rules. Using the current time, the time of the last call gapped, the call counter value (tracked internally by the Oracle Communications Session Border Controller), the CPS value for the gap-rate control rule, and the values of the new parameters, the Oracle Communications Session Border Controller performs calculations that determine whether or not to apply the control rule.

About the Call Gapping Algorithm

The Oracle Communications Session Border Controller employs this leaky bucket algorithm to enforce calls per second. It smooths the call rate over a defined window of time to protect against surges. The values used for the calculation are:

- A—Calls per second; configure by setting the **type** parameter to gap-rate, and the **value** parameter to the CPS you want enforced

- m—Maximum counter value; must be greater than 0
- W—Window size; must be greater than
- deltaT—Time between allowed calls matching an NM control rule

The calculation is performed as follows, with the noted results:

- 1 + m - m*A*deltaT/W =< M—Means the call is allowed
- 1 + m - m*A*deltaT/W >M—Means that NMC rules are applied

Note the following:

- Setting the counter value and the window size to the same values guarantees that the processed CPS load will not exceed the desired CPS target.
- As the counter value becomes greater than the window size value, rejection rate will drop and the desired CPS threshold is not guaranteed.
- Increasing the window size results in a lower rejection rate when the attempted CPS is the same as the desired CPS; as the attempted CPS rate increases, rejection rates increase at a steeper rate.
- If either the count rate or the window size is set to 0, then the Oracle Communications Session Border Controller reverts to call gapping behavior it uses when the relevant parameters are not configured.

Network Management Control Configuration

In order use the network management controls feature, you need to set control rules and then enable their application on a per-realm basis. This section shows you how to set up those configuration.

Configuring an Individual Control Rule

To configure individual network management control rule:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **net-management-control** and press Enter.

   ```
   ORACLE(session-router)# net-management-control
   ```

4. **name**—Enter the name of this network management control rule; this value uniquely identifies the control rule. There is no default for this parameter.
5. **state**—Enable or disable this network management control rule. The default value is **enabled**. The valid values are:
   - enabled | disabled
6. **destination-identifier**—Enter the call classification key. This parameter specifies information about the destination, which can be an IP address, an FQDN, a destination

(called) number, or destination prefix. You can wildcard characters in the classification key using the carat symbol (^).

You can enter special characters in the **destination-identifier** parameter. You can enter characters such as the plus-sign (+), the asterisk (*), the pound sign (#), capital letter A (A), capital letter B (B), capital letter C (C), capital letter D (D), lowercase letter p (p), lowercase letter w (w).

This parameter can accommodate a list of entries so that, if necessary, you can specify multiple classification keys. You can edit the list of classification keys using the ACLI **add-destination-identifier** and **remove-destination-identifier** commands from within the network management controls configuration.

7. **type**—Enter the control type that you want to use. The valid values are:
   - **GAP-RATE**—Controls the maximum sustained rate of calls that match the classification key.
   - **GAP-PERCENT**—Controls the percentage of calls matching the classification key you set for the control rule.
   - **PRIORITY**—Exempts calls to a destination (like 911) from local network management controls. Use this value if you want to enable Priority Call Exemption from Policy Server Approval.
8. **value**—When you set the control type to either GAP-RATE or GAP-PERCENT, enter the maximum sustained rate of calls or the percentage of calls to which you want the control rule applied. The default value is zero (**0**). The valid values are:
   - **GAP-RATE**—Maximum is 2147483647
     - Using the minimum value (0) means that the Oracle Communications Session Border Controller treats all calls
     - Using the maximum value means that the Oracle Communications Session Border Controller treats no calls
   - **GAP-PERCENT**—Maximum is 100
     - Using the minimum value (0) means that the Oracle Communications Session Border Controller treats no calls
     - Using the maximum value (100%) means that the Oracle Communications Session Border Controller treats all calls
9. **treatment**—Enter the treatment method that you want to use. The default value is **none**. The valid values are:
   - **reject**—The Oracle Communications Session Border Controller rejects the call.
   - **divert**—The Oracle Communications Session Border Controller routes the call to the location you specify in the control rule's next hop parameter.
10. **next-hop**—Enter the next hop for the Oracle Communications Session Border Controller to use when the treatment method is **DIVERT.** The valid values are:
    - hostname(:port)
    - IP address(:port)
    - Name of a valid, configured session agent
    - Name of a valid, configured session agent group—When you set this parameter to a session agent group, you must specify that it is a session agent group by prepending the name of the group with either SAG: or sag:. For example, the entry for a session agent group with Group2 as its name would be SAG:Group2 or sag:Group2.

11. **realm-next-hop**—Enter the realm identifier to designate the realm of the next hop when the treatment type is **DIVERT**.
12. **protocol-next-hop**—Enter the signaling protocol for the next hop when the treatment type is **DIVERT**.
13. **status-code**—Enter the SIP response code that you want the Oracle Communications Session Border Controller to use when the treatment method is **REJECT**. The default value is **503** (Service Unavailable). The valid range is:
    - Minimum—1
    - Maximum—699
14. **cause-code**—Enter the Q.850 cause code that you want the Oracle Communications Session Border Controller to use when the treatment method is **REJECT**. The default value is **63** (Service or option not available). The valid range is:
    - Minimum—1
    - Maximum—999999999

      For a SIP call, the Oracle Communications Session Border Controller replaces the cause code in the Reason header of the SIP response.

      For a H.323 call, the Oracle Communications Session Border Controller converts the cause code to a Q.931 cause code in the Q.931 Cause IE in the releaseComplete message.

Enabling Enhanced Call Gapping

Enhanced NMC call gapping uses new configuration parameters to the network management controls configuration:

- **gap-rate-max-count**—Maximum count that triggers the application of network management control rule if it is exceeded. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999
- **gap-rate-window-size**—Length of time in seconds used for the gapping rate calculation. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999

For this feature to behave as intended, you also need to set the **type** parameter to **gap-rate**, and set the **value** parameter to the maximum sustained rate of calls that you want to support.

To configure NMC call gapping enhancements:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **net-management-control** and press Enter.

```
ORACLE(session-router)# net-management-control
```

To add support to a pre-existing network management control configuration, use the ACLI **select** command to choose the configuration you want to edit.

- **gap-rate-max-count**—Maximum count that triggers the application of network management control rule if it is exceeded. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999
    Along with the current time, the last time of a gapped call, the call counter value, the CPS value, and the gap-rate-window-size value, the Oracle Communications Session Border Controller uses **gap-rate-max-count** as a measurement to determine if a control rule will be applied.
- **gap-rate-window-size**—Length of time in seconds used for the gapping rate calculation. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999
    Along with the current time, the last time of a gapped call, the call counter value, and the CPS value, the Oracle Communications Session Border Controller uses the **gap-rate-window-size** value to calculate whether the maximum count is within allowable limits.

1. Save and activate your configuration.

Applying a Network Management Control Rule to a Realm

Once you have configured network management control rules, you can enable their use on a per-realm basis.

To apply a network management control rule to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   If you are enabling network management controls for a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **net-management-control**—Set this parameter to **enabled** to apply network control rules in this realm. The default value is **disabled**. The valid values are:

- enabled | disabled

5. Save and activate your configuration.

3147 - Emergency Call Fallback

You can enable the rejection of priority calls per a configured SIP interface. When this option is enabled, the Oracle Communications Session Border Controller does not allow priority calls through that SIP interface, and the call is rejected with a 380 response.

If the send-380-response parameter in the SIP interface is configured, this string is provided as a reason in the 380 message. If this parameter is not populated, the default message priority calls not allowed is sent.

Emergency Call Fallback Configuration

To enable emergency call fallback, you must select an option for the SIP-interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
```

2. Type **session-router** and press Enter to access the system-level configuration elements.

```
ACMEPACKET(configure)# session-router
```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ACMEPACKET(session-router)# sip-interface
ACMEPACKET(sip-interface)#
```

4. **options**—Type **disallow-priority-calls** to send a 380 response to priority calls.

```
ACMEPACKET(session-router)# options disallow-priority-calls
```

5. Save and activate your configuration.

Accounting Configuration for QoS

This section explains how to configure the account configuration and account servers so you can use the Oracle Communications Session Border Controller in conjunction with external RADIUS (accounting) servers to generate CDRs and provide billing services requires.

QoS Accounting Configuration

To configure the account configuration and account servers:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **account-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# account-config
ORACLE(account-config)#
```

4. To configure account server parameters (a subset of the account configuration parameters, type **account-servers** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(account-config)# account-servers
ORACLE(account-server)#
```

The following example shows both the account config and account server parameters.

```
account-config
        hostname                        acctserver1
        port                            1813
        strategy                        Hunt
        state                           enabled
        max-msg-delay                   60
        max-wait-failover               100
        trans-at-close                  disabled
        generate-start                  OK
        generate-interim
                                        OK
                                        Reinvite-Response
account-server
                hostname                        192.168.2.2
                port                            1813
                state                           enabled
                min-round-trip                  100
                max-inactivity                  100
                restart-delay                   100
                bundle-vsa                      enabled
                secret                          testing
                NAS-ID                          acme-accounting
last-modified-date              2005-01-15 02:23:42
```

Account Configuration

You set the account configuration parameters to indicate where you want accounting messages sent, when accounting messages you want them sent, and the strategy you want used to select account servers.

To configure the account configuration:

1. **hostname**—Enter a name for the host associated with the Oracle Communications Session Border Controller in hostname (FQDN) format. The default value is the name of the local host.

The value you enter here must match the configured phy-interface's operation type **control** or **maintenance**, to determine on which network to send RADIUS messages.

2. **port**—Enter the number of the UDP port associated with the Oracle Communications Session Border Controller from which RADIUS messages are sent.The default value is **1813**. The valid range is:
   - Minimum—1025
   - Maximum—65535
3. **strategy**—Indicate the strategy you want used to select the accounting servers to which the Oracle Communications Session Border Controller will send its accounting messages. The default value is **hunt**. The following table lists the available strategies:
   - **hunt**—Selects accounting servers in the order in which they are listed.

     If the first accounting server is online, working, and has not exceeded any of the defined constraints, all traffic is sent to it. Otherwise the second accounting server is selected. If the first and second accounting servers are offline or exceed any defined constraints, the third accounting server is selected. And so on through the entire list of configured servers
   - **failover**—Uses the first server in the list of predefined accounting servers until a failure is received from that server. Once a failure is received, it moves to the second accounting server in the list until a failure is received. And so on through the entire list of configured servers.
   - **roundrobin**—Selects each accounting server in order, distributing the selection of each accounting server evenly over time.
   - **fastestrtt**—Selects the accounting server that has the fastest round trip time (RTT) observed during transactions with the servers (sending a record and receiving an ACK).
   - **fewestpending**—Selects the accounting server that has the fewest number of unacknowledged accounting messages (that are in transit to the Oracle Communications Session Border Controller).
4. **state**—Enable this parameter if you want the account configuration active on the system. Disable it if you do not want the account configuration active on the system. The default value is **enabled**. The valid values are:
   - enabled | disabled
5. **max-msg-delay**—Indicate the length of time in seconds that you want the Oracle Communications Session Border Controller to continue trying to send each accounting message. During this delay, the Oracle Communications Session Border Controller can hold a generic queue of 4096 messages. The default value is **60**.
   - Minimum—zero (0)
   - Maximum—4294967295
6. **max-wait-failover**—Indicate the maximum number of accounting messages the Oracle Communications Session Border Controller can store its message waiting queue for a specific accounting server, before it is considered a failover situation.

   Once this value is exceeded, the Oracle Communications Session Border Controller attempts to send it accounting messages, including its pending messages, to the next accounting server in its configured list. The default value is **100**. The valid range is:
   - Minimum—1
   - Maximum—4096

7. **trans-at-close**—Disable this parameter if you do not want to defer the transmission of message information to the close of a session. Enable it if you want to defer message transmission. The default value is **disabled**. The valid values are:

   - **disabled**—The Oracle Communications Session Border Controller transmits accounting information at the start of a session (Start), during the session (Interim), and at the close of a session (Stop). The transmitted accounting information for a single session might span a period of hours and be spread out among different storage files.
   - **enabled**—Limits the number of files on the Oracle Communications Session Border Controller used to store the accounting message information for one session. It is easiest to store the accounting information from a single session in a single storage file.

8. **generate-start**—Select the type of SIP event that triggers the Oracle Communications Session Border Controller to transmit a RADIUS Start message. The default value **is** ok. The valid values are:

   - **start**—RADIUS Start message should not be generated
   - **invite—**RADIUS Start message should be generated once the Oracle Communications Session Border Controller receives a SIP session INVITE.
   - **ok**—RADIUS Start message is generated once the Oracle Communications Session Border Controller receives an OK message in response to an INVITE.

9. **generate-interim**—Retain the default value **reinvite-response** to cause the Oracle Communications Session Border Controller to transmit a RADIUS Interim message. (A RADIUS Interim message indicates to the accounting server that the SIP session parameters have changed.)

   To disable interim message generation, enter a pair of quotes as the value for this parameter. Otherwise, select one or more than one of the following values:

   - **ok**—RADIUS Interim message is generated when the Oracle Communications Session Border Controller receives an OK message in response to an INVITE.
   - **reinvite**—RADIUS Interim message is generated when the Oracle Communications Session Border Controller receives a SIP session reINVITE message.
   - **reinvite-response**—RADIUS Interim message is generated when the Oracle Communications Session Border Controller receives a SIP session reINVITE and responds to it (for example, session connection or failure).
   - **reinvite-cancel**—RADIUS Interim message is generated when the Oracle Communications Session Border Controller receives a SIP session reINVITE, and the Reinvite is cancelled before the Oracle Communications Session Border Controller responds to it.

10. **account-server**—Create the account server list to store accounting server information for the account configuration. Each account server can hold 100 accounting messages.

    Account server entries are specific to the account configuration. They cannot be viewed or accessed for editing outside of the account configuration.

> **Note:**
>
> RADIUS will not work if you do not enter one or more servers in a list.

Account Server

You must establish the list of servers to which the Oracle Communications Session Border Controller can send accounting messages.

1. **hostname**—Name of the host associated with the account server as an IP address.
2. **port**—Enter the number of the UDP port associated with the account server to which RADIUS messages are sent. The default value is **1813**. The valid range is:
   - Minimum—1025
   - Maximum—65535
3. **state**—Enable or disable the account servers on the system. The default value is **enabled**. The valid values are:
   - enabled | disabled
4. **min-round-trip**—Indicate the minimum round trip time of an accounting message in milliseconds. The default value is **250**. The valid range is:
   - Minimum—10
   - Maximum—5000

   A round trip consists of the following:

   The system sends an accounting message to the account server.

   The account server processes this message and responds back to the Oracle Communications Session Border Controller.

   If the **fastest RTT** is the **strategy** for the account configuration, the value you enter here can be used to determine an order of preference (if all the configured account servers are responding in less than their minimum RTT).
5. **max-inactivity**—Indicate the length of time in seconds that you want the Oracle Communications Session Border Controller with pending accounting messages to wait when it has not received a valid response from the target account server. The default value is **60**. The valid range is:
   - Minimum—1
   - Maximum—300

   Once this timer value is exceeded, the Oracle Communications Session Border Controller marks the unresponsive **account server** as disabled in its failover scheme. When a server connection is marked as inactive, the Oracle Communications Session Border Controller attempts to restart the connection and transfers pending messages to another queue for transmission. RADIUS messages might be moved between different **account server**s as servers become inactive or disabled.
6. **restart-delay**—Indicate the length of time in seconds you want the Oracle Communications Session Border Controller to wait before resending messages to a disabled account server. The default value is **30**. The valid range is:
   - Minimum—1
   - Maximum—300
7. **bundle-vsa**—Retain the default **enabled** if you want the account server to bundle the VSAs within RADIUS accounting messages. Enter **disabled** if you do not want the VSAs to be bundled. (Bundling means including multiple VSAs within the vendor value portion of the message.) The valid values are:

- enabled | disabled

  In a bundled accounting message, the RADIUS message type is vendor-specific, the length is determined for each individual message, and the vendor portion begins with a 4-byte identifier, and includes multiple vendor type, vendor length, and vendor value attributes.

8. **secret**—Enter the secret passed from the account server to the client in text format. Transactions between the client and the RADIUS server are authenticated by the shared secret; which is determined by the source IPv4 address of the received packet.
9. **NAS-ID**—Enter the NAS ID in text format (FQDN allowed). The account server uses this value to identify the Oracle Communications Session Border Controller for the transmittal of accounting messages.

   The remote server to which the **account configuration** sends messages uses at least one of two potential pieces of information for purposes of identification. The Oracle Communications Session Border Controller accounting messages always includes in the first of these:

   - Network Access Server (NAS) IP address (the IP address of the Oracle Communications Session Border Controller's SIP proxy)
   - **NAS ID (t**he second piece of information) provided by this value. If you enter a value here, the NAS ID is sent to the remote server.

Allowlists for Managing Incoming SIP Headers and Parameters

By default, the Oracle Communications Session Border Controller (SBC) ignores unknown SIP headers and URI parameters and passes them through. If you want the SBC to accept only messages with headers and URI parameters complying with those supported by your internal equipment, you can use allowlists to control unknown headers and parameters in request and response traffic. An allowlist is an approved list of entities for which your equipment provides particular privileges, access, and recognition. The SBC uses configured allowlist profiles to control and accept specific inbound SIP headers and URI parameters. When you configure this service, the SBC rejects requests not matching the configured profile, or removes the headers or URI parameters not specified in the configured profile.

With allowlists, you can specify which SIP signaling messages you want to allow into your network and which messages to reject or delete. In the flow of SIP traffic to and from the SBC, the SBC matches any received request or response against the allowlist and rejects or deletes elements that do not match based on the actions specified in the allowlist configuration.

For responses, the SBC does not reject the message if a header or parameter is not found in the allowlist even when the action is set to reject. Instead, the SBC deletes the offending parameter or header. In addition, if the message is a request of the method type ACK, PRACK, CANCEL or BYE, the SBC deletes all unmatched elements and does not reject the request even when the action is configured to reject.

The allowlist verification performs for any method, but you can narrow the list to operate only on specific methods by defining them in the **methods** parameter of the configuration.

Allowlist verification occurs when the SBC receives a request or response, but only after the SBC processes the inbound header manipulation rule (HMR), network management controls (NMC), Resource-Priority header (RPH), and monthly minutes checking.

The SBC responds to requests with non-matching headers or parameters configured with an action of reject with a "403 Forbidden" response, by default. You can use a local-response event, **allowed-elements-profile-rejection**, to override the default reject status code and reason phrase.

The configured allowlist operates transparently on headers that contain multiple URIs or multiple header values within a single header (header values separated by a comma).

Parameter parsing operates only on parameters that it can identify. For parameters that cannot be parsed, for example an invalid URI (<sip:user@example.com&hp=val>), the SBC ignores this URI header parameter value of "hp" because it is not contained within a valid URI. Although it might look like a URI header parameter, URI headers must come after URI parameters. Parameter matching does not occur if the headers and parameters in the URI are not well-formed. The SBC does not remove the parameter just because it cannot identify it.

What is an Allowlist?

An allowlist is an approved list of entities for which equipment provides particular privileges, access, and recognition. The Oracle Communications Session Border Controller can use configured allowlist profiles to control and accept specific inbound SIP headers and URI parameters that are being passed-through the Oracle Communications Session Border Controller. When you configure an allowlist, the Oracle Communications Session Border Controller rejects requests not matching the configured profile, or removes the unspecified headers or URI parameters not in the configured profile.

Configure Allowlists for SIP Header and URI Parameter Management

You can configure allowlist profiles that tell the Oracle Communications Session Border Controller (SBC) to accept only inbound SIP headers and URI parameters that are configured in this allowlist. Using the **allowed-elements-profile** parameter, you can configure the settings for this parameter using the ACLI interface at **session-router**, **enforcement-profile**. Because the **enforcement-profile** object also pertains to session agents, realms, and SIP interfaces, you can also apply the enforcement profiles you configure to these entities. (Use the ACLI interface at **session-router**, **session-agent**, **session-router**, **sip-interface**, and **media-manager**, **realm-config**.)

The following configuration example assumes that your baseline configuration passes SIP traffic, with the SBC in the role of an Access SBC. Use this procedure to configure a allowlist for the session router and optionally apply the specific allowlists to the session agent and SIP interface, as well as the media manager realm configuration.

1. Access the **allowed-elements-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# allowed-elements-profile
ORACLE(allowed-elements-profile)#
```

2. **name**—Type a unique name for the allowlist you are creating. You can reference this name when configuring the enforcement-profiles for session-agent, SIP interface, and realm-config.

```
ORACLE(allowed-elements-profile)# name allowlist1
```

3. **description**—Type a description that explains the purpose of creating this allowlist. Valid values: Any alpha-numeric characters.

```
ORACLE(allowed-elements-profile)# description Basic Allowlist
```

4. Navigate to the **rule-sets** configuration element to specify the rules to match against specific incoming SIP headers and URI parameters.

```
ORACLE(allowed-elements-profile)# rule-sets
ORACLE(rule-sets)#
```

5. **unmatched-action**—Select the action for the SBC to perform when a header does not exist in an incoming message. Default: **reject**. Valid values:
   - **reject**—Rejects all incoming messages that do not contain a header.
   - **delete**—Deletes all incoming message that do not contain a header.

> **Note:**
>
> This parameter applies to non-matching header names, only. (It does not apply to non-matching URI parameters.)

```
ORACLE(rule-sets)# unmatched action delete
```

6. **msg-type**—Specify the type of messages to which the SBC applies this allowlist configuration. Default: **any**. Valid values:
   - **any**—Applies to all incoming messages.
   - **request**—Applies to incoming REQUEST messages, only.
   - **response**—Applies to incoming RESPONSE messages, only.

```
ORACLE(rule-sets)# msg-type any
```

7. **methods**—Type the packet methods, separated by a comma, for which this allowlist is enforced. Packet methods include, INVITE, OPTIONS, ACK, BYE, and so on. If this field is left blank, the allowlist applies to all packet methods. You can type up to a maximum of 255 characters.

```
ORACLE(rule-sets)# methods INVITE,ACK,BYE
```

8. **logging**—Select whether or not an incoming message is written to the **matched.log** file, when the message contains an element not specified in the allowlist. Default: disabled. Valid values: enabled | disabled.
9. Navigate to **allowed-header-rule**—Configure multiple parameters as part of the allowlist to make up the header rule that the SBC allows from incoming messages. You can configure an unlimited number of header-rules, and they do not need to be in any specific order. The system prompt changes to let you know that you can begin configuring individual parameters for this object.

```
ORACLE(rule-set)# header-rule
ORACLE(allowed-header-rule)#
```

10. **header-name**—Type the name of the header in the allowlist that you want theSBC to allow from incoming messages. The text is not case sensitive and supports abbreviated forms of header names. For example, “Via”, “via”, or “v” all match against the same header. A header name of “request-uri” refers to the request URI of requests, while a header name of * applies to any header-type not matched by any other header-rule. Default: *. The default

value allows header-rules for commonly known headers that remove unknown parameters, but leave unknown headers alone.

```
ORACLE(allowed-header-rule)# header-name Contact
```

11. **unmatched-action**—Select the action for the SBC to perform when an incoming header's parameters do not match the relevant allowed parameters specified for this header-name. Default: reject. Valid values:
    - **reject**—Rejects all incoming messages that have header parameters that do not match the parameters specified in this header-name.
    - **delete**—Deletes all incoming messages that have header parameters that do not match the parameters specified in this header-name.

    > **Note:**
    >
    > This parameter applies to non-matching header names only (not to non-matching URI parameters).

```
ORACLE(allowed-header-rule)# unmatched-action delete
```

12. **allow-header-param**—Type the header parameter that the SBC allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). Default: *, which allows all header parameters to pass through. If you leave this field empty, no header parameters are allowed.

```
ORACLE(allowed-header-rule)# allow-header-param *
```

13. **allow-uri-param**—Type the URI parameter that the SBC allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). Default: *, which allows all URI parameters to pass through. If you leave this field empty, no URI parameters are allowed.

```
ORACLE(allowed-header-rule)# allow-uri-param *
```

14. **allow-uri-user-param**—Type the URI user parameter that the SBC allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). Default: *, which allows all URI user parameters to pass through. If you leave this field empty, no URI user parameters are allowed.

```
ORACLE(allowed-header-rule)# allow-uri-user-param *
```

15. **allow-uri-header-name**—Type the URI header name that the SBC allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). Default: *, which allows all URI header name parameters to pass through. If you leave this field empty, no URI header name parameters are allowed.

```
ORACLE(allowed-header-rule)# allow-uri-header-name *
```

16. Save your work using the **done** command.

Configuration Exception

In certain circumstances, the Oracle Communications Session Border Controller (SBC) ignores specific parameters in incoming Request-URI messages and automatically adds header-rules.

In a Request-URI, all parameters are URI parameters and URI headers are not allowed. If you define values for the “allow-header-param”, “allow-uri-header-name”, and “allow-uri-param”, the SBC ignores these parameters in the Request-URI. Instead, the SBC automatically adds header-rules for incoming “Via”, “From”, To”, “Call-ID”, and “CSeq” messages. These are explicit header rules that you cannot delete. Each header-rule in a Request-URI includes parameters populated with the value of *. If required, you can change the header-rule parameter values with the values identified in the following table.

Header Rule	Applicable Parameter	Required Value(s)
Via	allow-header-param	• branch • received • rport
From	allow-header-param	• tag
To	allow-header-param	• tag
Call-ID	allow-header-param	No restrictions
CSeq	allow-header-param	No restrictions

Verify Allowlist Configuration

After you configure and save an allowlist on the Oracle Communications Session Border Controller (SBC), you can use the **verify-config** command at the top level prompt to verify the saved configuration: For example:

```
ORACLE# verify-config
```

The **verify-config** command checks for errors in the SBC configuration. Allowlist configuration errors specifically related to the enforcement-profile object also display in the output of this command when applicable. The allowlist configuration errors display if any references to the allowed-element-profiles are improperly configured. If errors exist, the system displays the following message:

```
--------------------------------------------------------------------
ERROR: enforcement-profile [ep] contains a reference to an allowed-
enforcement-profile [abc] that does not exist
--------------------------------------------------------------------
```

How Allowlists Work

Allowlists allow you to customize which SIP signaling messages to allow into your network and which messages to reject or delete. In the flow of SIP traffic to/from the Oracle Communications Session Border Controller, the Oracle Communications Session Border Controller matches any received request or response, in or out of a dialog against the configured allowed list, and rejects or deletes the non-matching element based on the actions specified in the allowlist configuration.

For responses, the Oracle Communications Session Border Controller does not reject the message if a header or parameter is not found in the allowed list, even if the action is set to reject. Instead it deletes the offending parameter or header. In addition, if the message is a request of the method type ACK, PRACK, CANCEL or BYE, it deletes all unmatched elements, but does not reject the request, even if the action was configured to reject.

The allowlist verification performs for any method; however you can narrow this list to operate only on specific methods by defining them in the **methods** parameter of the configuration.

Allowlist verification occurs when a request or response is received by the Oracle Communications Session Border Controller, but only after the Oracle Communications Session Border Controller has processed the inbound header manipulation rule (HMR), network management controls (NMC), Resource-Priority header (RPH), and monthly-minutes checking.

The Oracle Communications Session Border Controller responds to requests which have non-matching headers or parameters configured with an action of reject, with a "403 Forbidden" response by default. You can use a local-response event, **allowed-elements-profile-rejection**, to override the default reject status code and reason phrase.

The configured allowlist operates transparently on headers that have multiple URIs or multiple header values within a single header (header values separated by a comma).

Parameter parsing operates only on parameters that it can identify. For parameters that can not be parsed, for example an invalid URI (e.g. <sip:user@example.com&hp=val>), the Oracle Communications Session Border Controller ignores this URI header parameter value of "hp" since it is not contained within a valid URI. Even though it would appear to be a URI header parameter, URI headers must come after URI parameters. Parameter matching does not occur if the headers and parameters in the URI are not well-formed. The Oracle Communications Session Border Controller does not remove the parameter since it cannot identify it.

Allowlist Learning

You can build a SIP header and URI parameter allowlist configuration by way of the learning capabilities of the Oracle Communications Session Border Controller (SBC). When you enable learning mode on the SBC, it acquires knowledge of the allowable headers and parameters currently coming into your network. The SBC collects the information about the headers received and the parameters that exist within each header. The system gathers the information until you disable the learning mode.

After you disable the learning mode, the SBC prompts you to enter a name for the allowed-elements-profile. If the profile name you entered does not exist, the SBC writes the captured information to the new allowed-elements-profile configuration. The administrator can then make changes to the configuration as applicable, save the configuration, and apply it to a logical remote entity.

The allowed-elements-profile does not contain any wild card rules because the SBC cannot generate wild card headers and parameters during the learning mode. The Methods object is populated from the list of methods seen by the SBC while learning.

> **Note:**
>
> Oracle recommends running the learning mode during off-peak and light traffic periods. Learning mode can operate in conjunction with the execution of an allowed-elements-profile. The learning occurs just before any configured allowed-elements-profile configuration.

Allowlist Learning Configuration

The ACLI interface provides two commands that allow a Superuser to start and stop allowlist learning on the Oracle Communications Session Border Controller (SBC):

Command	Description
start <argument> <options>	Starts allowlist learning on the SBC. You must specify the argument learn-allowed-elements with this command to start the learning operation. Optionally, you can use method, msg-type, and params after the argument.
stop <argument> <identifier>	Stops the allowlist learning on the SBC and writes the learned configuration to the editing configuration on the SBC where it is saved and activated. You must specify the argument learn-allowed-elements with this command to stop the learning operation. You must specify a unique identifier that identifies the allowed-elements-profile name. If you specify an identifier name that already exists as a profile, the ACLI returns an error message and prompts you to enter a different name.

You can use these commands at the top level ACLI prompt as required on the SBC.

You use these commands with the argument, learn-allowed-elements to start and stop allowlist learning. By default, the learning mode creates a single rule-set under which all of the headers and their respective parameters are stored.

For example:

```
ORACLE# start learn-allowed-elements
Learning mode for allowed-elements-profile started.
```

In the preceding example, **start** is the top level ACLI command and **learn-allowed-elements** is the operation being performed.

Optionally, you can specify [method], [msg-type], and [params] in any order, for the Oracle Communications Session Border Controller to learn specific rule-set elements from incoming messages and save them to the allowlist configuration.

For example:

```
ORACLE# start learn-allowed-elements method msg-type params
```

The **method** option creates a new rule-set per unique method. The **msg-type** option creates a new rule-set per unique message-type seen. The **params** option performs URI and header parsing to examine parameters within the message. By default, parameter parsing is disabled.

Rejected Messages Monitoring

Allowlists control whether or not the Oracle Communications Session Border Controller (SBC) accepts unknown headers and URI parameters in incoming request and response traffic. When the SBC rejects messages according to the allowlist, the system logs the rejected messages a

file called “matched.log,” if you set logging to enabled. You can open and view the log when you want to view the rejected messages.

In addition to sending rejected messages to the “matched.log” file, the system sends rejected messages through a burst counter that keeps track of the number of messages rejected. You can enter the **show sipd** command to display the number of rejected messages. The counter is titled Rejected Message.

12

Static Flows

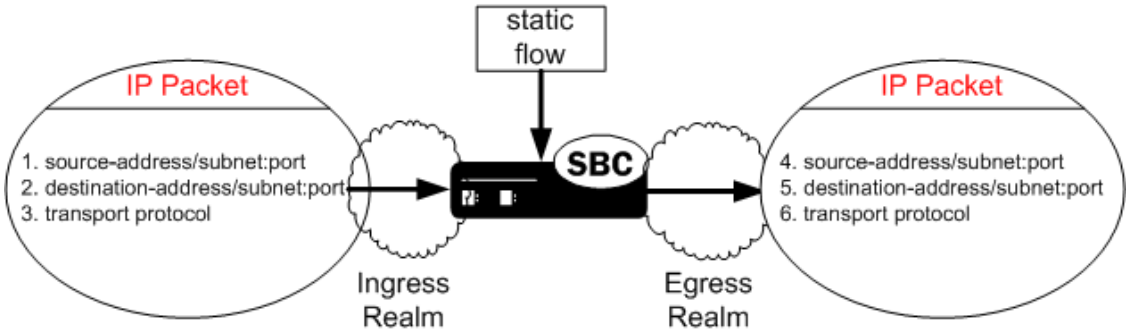
This chapter describes the Oracle Communications Session Border Controller's static flows feature. Static flows allow network traffic that matches specific criteria to pass through the Oracle Communications Session Border Controller unrestricted. Static flows are unidirectional. This feature lets you steer traffic toward a particular destination based on its original characteristics. Static flows can range from being widely accessible to very restrictive, depending on the values you establish. Static flows are used for transporting a variety of signaling messages through the Oracle Communications Session Border Controller to achieve vendor interoperability. The Oracle Communications Session Border Controller supports the following types of Static flows:

- IPv6 to IPv6 flows
- IPv4 to IPv6 flows
- IPv4 to IPv4 flows

> **Note:**
>
> Traffic that traverses the Oracle Communications Session Border Controller in two directions, such as ICMP requests and responses, requires static flows configured for both directions.

About Static Flows

The static flow element explicitly writes entries into the IP routing table. These entries are persistent and are not deleted as calls are set up and broken down. Refer to the following diagram to understand how a static flow works.



A static flow entry watches for traffic with specific criteria on a specified ingress realm; that traffic consists of the following criteria:

- The packet enters the Oracle Communications Session Border Controller on the specified ingress realm.

- The packet contains matching source address, subnet, and port criteria, field 1.
- The packet contains matching destination address, subnet, and port criteria, field 2.
- The packet contains a matching transport protocol, field 3.

If the above conditions are met, then the Oracle Communications Session Border Controller does the following:

- The IPv4 traffic is forwarded out of the Oracle Communications Session Border Controller on the specified egress realm.
- The configured source address, subnet, and port criteria are written to the exiting packet, field 4.
- The configured destination address, subnet, and port criteria are written to the exiting packet, field 5.
- The original transport protocol and its contents remain unchanged as the packet exits into the egress realm.

IPv6 / IPv4 Translations

The ingress or egress traffic type, whether IPv4 or IPv6, must match the configuration of the realm where attached, as **in-realm-id** or **out-realm-id**. A realm and IP version configuration mismatch results in an error message and log entry at error level.

IPv6 to IPv4 flows exit the Oracle Communications Session Border Controller with the prefix `::fff:0:0:0/96`. They may be written as `::ffff:0:a.b.c.d`, where a.b.c.d refers to an IPv6-enabled node.

While IPv4 addresses can be translated into IPv6 addresses, IPv6 address can not be translated to IPv4.

About Network Address Translation ALG

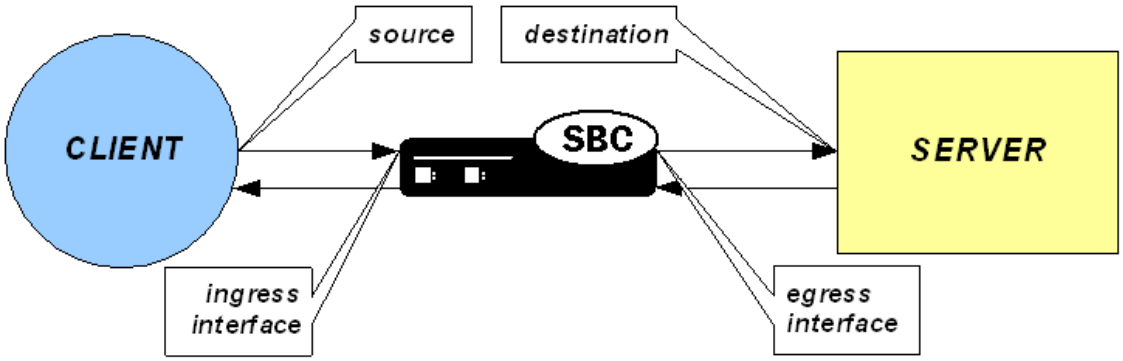
The Oracle Communications Session Border Controller supports Network Address and Port Translation (NAPT) and Trivial File Transfer Protocol (TFTP) functionality over media interfaces, collectively known as Network Address Translation (NAT) ALG. The NAT ALG feature is implemented as an extension of the static flow feature.

In some applications, the Oracle Communications Session Border Controller acts as an intermediary device, positioned between endpoints located in an access network and application servers located in a backbone network. The Oracle Communications Session Border Controller 's NAT ALG feature enables these endpoints to use non-VoIP protocols, such as TFTP and HTTP, to access servers in a provider's backbone network to obtain configuration information.

NAT ALG parameters support RTC and can be dynamically reconfigured. The active NAT ALG configuration can be replicated on the standby SD in an HA configuration.

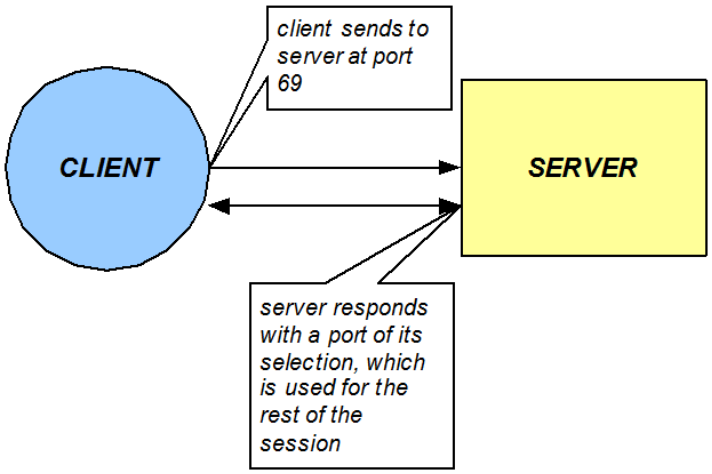
NAPT

The NAPT ALG functionality is the same as that found in commercially available enterprise and residential NAT devices. The SBC watches for packets entering a media interface that match source and destination IP address criteria. Matching packets are then redirected out of the egress interface, through a specified port range, toward a destination address.

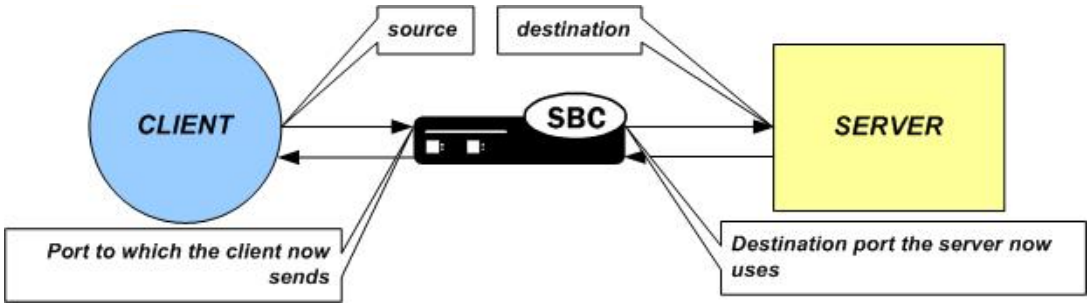


TFTP

The TFTP ALG is implemented as an extension of the NAT ALG. It works slightly differently than traditional NAPT. In a TFTP session, the first packet is sent from a source endpoint to port 69 on the TFTP server. The TFTP server responds from another port. This port, from which the TFTP response originates, is used for the remainder of the TFTP session.



To act as a TFTP ALG, the Oracle Communications Session Border Controller will latch on the first return packet from the server to learn the server's port. The ingress-side destination port of the Oracle Communications Session Border Controller is changed to reflect the new communications port for the TFTP session. This process takes place without any user intervention.



Configuring Static Flows

This section explains how to configure static flows. It also provides sample configurations for your reference. You can configure static flows with or without NAT ALG. If you configure static flows with NAT ALG, you can choose NAPT or TFTP as the ALG type.

Basic Static Flow Configuration Overview

This section outlines the basic static flow configuration, without NAT ALG. You configure static flows by specifying ingress traffic criteria followed by egress re-sourcing criteria.

When configuring static flows, the following conventions are used:

- An address of 0.0.0.0 matches all addresses. This token is used as the wildcard for both IPv4 and IPv6 static flows
- Enclose the address portion of an IPv6 address in brackets: `[7777::11]/64:5000`
- Not specifying a port implies all ports.
- Not specifying a subnet mask implies a /32, matching for all 32 bits of the IPv4 address , or a /128 matching for all 128 bits of the IPv6 address.

1. Set the static flows' incoming traffic-matching criteria. First set the ingress realm where you expect to receive traffic that will be routed via a static flow. Second, set the traffic's source IP address, source subnet, and source port or port range criteria. Third, set the traffic's destination IP address, destination subnet, and destination port criteria. This is usually an external address on theOracle Communications Session Border Controller.
2. Set the criteria that describes how traffic should be translated on the egress side of the Oracle Communications Session Border Controller. First set the egress realm where you want to send the traffic to be routed by this static flow. Second, set the traffic's source IP address, source subnet, and source port or port range criteria. This is usually an external address on the Oracle Communications Session Border Controller. Third, set the traffic's destination IP address, destination subnet, and destination port criteria.
3. Set the protocol this static flow entry acts upon. This type of packet, as the payload of the IP packet, remains untouched as traffic leaves the Oracle Communications Session Border Controller . Specifying a layer 4 protocol here acts as another criteria to filter against for this static flow.

   The combination of entries in the ingress realm, ingress source address, ingress destination address, and protocol fields must be unique. For bidirectional traffic, you need to define a separate static flow in the opposite direction.

Static Flow Configuration

This section describes how to configure the **static-flow** element using the ACLI.

The ingress IP address criteria is set first. These parameters are applicable to traffic entering the ingress side of the Oracle Communications Session Border Controller .

- **in-realm-id**—The access realm, where endpoints are located.
- **in-source**—The source network in the access realm where the endpoints exist. This parameter is entered as an IP address and netmask in slash notation to indicate a range of possible IP addresses.

- **in-destination**—The IP address and port pair where the endpoints send their traffic. This is usually the IP address and port on a Oracle Communications Session Border Controller interface that faces the access realm.

The egress IP address criteria is entered next. These parameters determine how traffic is re-sourced as it leaves the Oracle Communications Session Border Controller and enters the backbone network.

- **out-realm-id**—The backbone realm, where servers are located.
- **out-source**—The IP address on the interface of the Oracle Communications Session Border Controller where traffic exits the Oracle Communications Session Border Controller into the backbone realm. Do not enter a port for this parameter.
- **out-destination**—The IP address and port pair destination of the traffic. This is usually a server in the backbone realm.
- **protocol**—The protocol associated with the static flow. The protocol you choose must match the protocol in the IPv4 header. Valid entries are TCP, UDP, ICMP, ALL.

The type of NAT ALG, if any.

- **alg-type**—The type of NAT ALG. Set this to NAPT, TFTP, or none.

The port range for port re-sourcing as traffic affected by the NAT ALG exits the egress side of the Oracle Communications Session Border Controller is set next. (Not applicable if **alg-type** is set to none.)

- **start-port**—The starting port the NAT ALG uses as it re-sources traffic on the egress side of the Oracle Communications Session Border Controller .
- **end-port**—The ending port the NAT ALG uses as it re-sources traffic on the egress side of the Oracle Communications Session Border Controller .

The flow timers are set next. (Not applicable if **alg-type** is set to none.)

- **flow-time-limit**—Total session time limit in seconds. The default is 0; no limit.

> **Note:**
>
> Note that the static flow-time-limit must have a value larger than initial-guard-timer and subsq-guard-timer for static flows.

- **initial-guard-timer**—Initial flow guard timer for an ALG dynamic flow in seconds. The default is 0; no limit.
- **susbsq-guard-timer**—Subsequent flow guard timer for an ALG dynamic flow in seconds. The default is 0; no limit.

Finally, you can set the optional bandwidth policing parameter for static flows (with or without NAT ALG applied).

- **average-rate-limit**—Sustained rate limit in bytes per second for the static flow and any dynamic ALG flows. The default is 0; no limit.
  To configure static flow:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the **media-manager** path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **static-flow** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# static-flow
   ```

   From this point, you can configure media policing parameters.

4. **in-realm-id**—Enter the ingress realm or interface source of packets to match for static flow translation. This in-realm-id field value must correspond to a valid identifier field entry in a **realm-config**. This is a required field. Entries in this field must follow the Name Format.
5. **in-source**—Enter the incoming source IP address and port of packets to match for static flow translation. IP address of 0.0.0.0 matches any source address. Port 0 matches packets received on any port. The port value has no impact on system operation if either ICMP or ALL is the selected protocol. This parameter takes the format:

   in-source <ip-address>[:<port>]

   The default value is **0.0.0.0**. The valid port range is:

   - Minimum—0
   - Maximum—65535
6. **in-destination**—Enter the incoming destination IP address and port of packets to match for static-flow translation. An IP address of 0.0.0.0 matches any source address. Port 0 matches packets received on any port. The port value has no impact on system operation if either ICMP or ALL is the selected protocol. The in-source parameter takes the format:

   in-destination <ip-address>[:<port>]

   The default value is **0.0.0.0**. The valid port range is:

   - Minimum—0
   - Maximum—65535
7. **out-realm-id**—Enter the defined realm where traffic leaving this NAT ALG exits theOracle Communications Session Border Controller .
8. **out-source**—Enter the egress IPv4 address. This is the IPv4 address of the network interface where traffic subject to the NAT ALG you are defining leaves the Oracle Communications Session Border Controller . Do not enter a port number for this parameter. The default value is **0.0.0.0**.
9. **out-destination**—Enter the IPv4 address and port number of the server or other destination to which traffic is directed. The default value is **0.0.0.0**. The valid port range is:

   - Minimum—0
   - Maximum—65535
10. **protocol**—Enter the protocol this NAPT ALG acts upon. The default value is **UDP**. The valid values are:

    - TCP | UDP | ICMP | ALL
11. **alg-type**—Enter the type of NAT ALG to use. The default value is **none**. The valid values are:

    - **none**—No dynamic ALG functionality

- **NAPT**—Configure as NAPT ALG
- **TFTP**—Configure as TFTP ALG

12. **start-port**—Enter the beginning port number of the port range that the Oracle Communications Session Border Controller allocates on the egress side for flows that this NAPT ALG redirects. The default value is **0**. The valid range is:
    - Minimum—0, 1025
    - Maximum—65535
13. **end-port**—Enter the ending port number of the port range that the Oracle Communications Session Border Controller allocates on the egress side for flows that this NAPT ALG redirects. The default value is **0**. The valid range is:
    - Minimum—0, 1025
    - Maximum—65535
14. **flow-time-limit**—Enter the total time limit for a flow in seconds. A value of **0** means there is no limit. The valid range is:
    - Minimum—0
    - Maximum—999999999
15. **initial-guard-timer**—Enter the initial guard timer value in seconds. A value of **0** means there is no limit. The valid range is:
    - Minimum—0
    - Maximum—999999999
16. **subsq-guard-timer**—Enter the subsequent guard timer value in seconds. A value of **0** means there is no limit. The valid range is:
    - Minimum—0
    - Maximum—999999999
17. **average-rate-limit**—Enter a maximum sustained rate limit in bytes per second. The default value is **0**; no limit. The valid range is:
    - Minimum—0
    - Maximum—125000000

    The following example shows a **static-flow** configuration element configured for a NAPT ALG.

```
in-realm-id                access
in-source                  172.16.0.0/16
in-destination             172.16.1.16:23
out-realm-id               backbone
out-source                 192.168.24.16
out-destination            192.168.24.95:23
protocol                   TCP
alg-type                   NAPT
start-port                 11000
end-port                   11999
flow-time-limit            0
initial-guard-timer        60
subsq-guard-timer          60
average-rate-limit         0
```

Example Configuration: Bidirectional Static Flows

The configuration lines below present the configuration of two example static flows to be used for ICMP to a specific host through the Oracle Communications Session Border Controller.

The following lines present the example configuration for the access to core side.

```
static-flow
in-realm-id access
description
in-source 0.0.0.0
in-destination 10.1.215.63
out-realm-id core
out-source 10.2.214.63
out-destination 10.2.214.51
protocol ICMP
alg-type none
start-port 0
end-port 0
flow-time-limit 0
initial-guard-timer 60
subsq-guard-timer 60
average-rate-limit 0
```

The following lines present the example configuration for the core to access side.

```
static-flow
in-realm-id core
description
in-source 10.2.214.51
in-destination 10.2.214.63
out-realm-id access
out-source 10.1.215.63
out-destination 0.0.0.0
protocol ICMP
alg-type none
start-port 0
end-port 0
flow-time-limit 0
initial-guard-timer 60
subsq-guard-timer 60
average-rate-limit 0
```

13

High Availability Nodes

SBCs can be deployed in pairs to deliver high availability (HA). Two SBCs operating in this way are called an HA node. Over the HA node, media and call state are shared, keeping sessions/calls from being dropped in the event of a failure.

Two SBCs work together in an HA node, one in active mode and one in standby mode.

- The active SBC checks itself for internal process and IP connectivity issues. If it detects that it is experiencing certain faults, it will hand over its role as the active system to the standby SBC in the node.
- The standby SBC is the backup system, fully synchronized with active SBCs session status. The standby SBC monitors the status of the active system so that, if needed, it can assume the active role without the active system having to instruct it to do so. If the standby system takes over the active role, it notifies network management using an SNMP trap.

In addition to providing instructions for how to configure HA nodes and their features, this chapter explains how to configure special parameters to support HA for all protocols.

The SBC uses SSH keys to manage the switchover. When you enable HA, you'll see new known-host keys and new authorized keys added to the configuration of both the active and standby configuration.

Overview

To produce seamless switchovers from one SBC to the other, the HA node uses shared virtual MAC and virtual IPv4 or IPv6 addresses for the media interfaces in a way that is similar to VRRP (virtual router redundancy protocol). When there is a switchover, the standby SBC sends out gratuitous Address Resolution Protocol (ARP) or Network Discovery Protocol (NDP) messages using the virtual MAC address, establishing that MAC on another physical port within the Ethernet switch. To the upstream router, the MAC and IP are still alive, meaning that existing sessions continue uninterrupted.

> **Note:**
>
> NDP is an equivalent to ARP for IPv6.

Within the HA node, the SBCs advertise their current state and health to one another in checkpointing messages; each system is apprised of the other’s status. Using Oracle's HA protocol, the SBC communicates with UDP messages sent out and received on the rear interfaces.

The standby SBC shares virtual MAC and IP addresses for the media interfaces (similar to VRRP) with the active SBC. Sharing addresses eliminates the possibility that the MAC and IP address set on one SBC in an HA node will be a single point of failure. The standby SBC sends ARP or NDP requests using a utility IP address and its hard-coded MAC addresses to obtain Layer 2 bindings.

> **Note:**
>
> The system does not allow you to ping from a secondary SBC media interface, presenting a warning if you try. This prevents you from creating conflicts in the resolution of your interfaces in neighboring switches.

The standby SBC assumes the active role when:

- It has not received a checkpoint message from the active SBC for a certain period of time.
- It determines that the active SBC's health score has decreased to an unacceptable level.
- The active Oracle Communications Session Border Controller relinquishes the active role.

The SBC uses SSH keys to manage the switchover and handle HDR replication. When you enable HA, you'll see new known-host keys and new authorized keys added to the configuration of both the active and standby configuration.

You can use BFD for media interface gateway and virtual address availability within the context of HA. Refer to the section on Bidirectional Forwarding Detection for information and instructions on using BFD for HA.

Establishing Active and Standby Roles

Oracle Communications Session Border Controller s establish active and standby roles in the following ways.

- If an SBC boots up and is alone in the network, it is automatically the active system. If you then pair a second SBC with the first to form an HA node, then the second system to boot up will establish itself as the standby automatically.
- If both SBCs in the HA node boot up at the same time, they negotiate with each other for the active role. If both systems have perfect health, then the SBC with the lowest HA rear interface IPv4 address will become the active SBC. The SBC with the higher HA rear interface IPv4 address will become the standby SBC.
- If the rear physical link between the two SBCs fails during boot up or operation, both will attempt to become the active SBC. In this case, processing will not work properly.

Health Score

HA Nodes use health scores to determine their active and standby status. Health scores are based on a 100-point system. When an SBC is functioning properly, its health score is 100.

Generally, the SBC with the higher health score is active, and the SBC with the lower health score is standby. However, the fact that you can configure health score thresholds builds some flexibility into using health scores to determine active and standby roles. This could mean, for example, that the active SBC might have a health score lower than that of the standby SBC, but a switchover will not take place because the active SBC's health score is still above the threshold you configured.

Alarms are key in determining health score. Some alarms have specific health score value that are subtracted from the SBC's health score when they occur. When alarms are cleared, the value is added back to the SBC's health score.

You can look at an SBC's health score using the ACLI **show health** command.

Switchovers

A switchover occurs when the active Oracle Communications Session Border Controller stops being the active system, and the standby system takes over that function. There are two kinds switchovers: automatic and manual.

Automatic Switchovers

Automatic switchovers are triggered without immediate intervention on your part. SBCs switch over automatically in the following circumstances:

- When the active SBC's health score of drops below the threshold you configure.
- When a time-out occurs, meaning that the active SBC has not has not sent checkpointing messages to the standby SBC within the allotted time.
  The active SBC might not send checkpointing messages for various reasons such as link failure, communication loss, or advertisement loss. Even if the active SBC has a perfect health score, it will give up the active role if it does not send a checkpoint message or otherwise advertise its status within the time-out window. Then the standby SBC takes over as the active system.

When an automatic switchover happens, the SBC that has just become active sends an ARP message to the switch. This message informs the switch to send future messages to its MAC address. The SBC that has just become standby ignores any messages sent to it.

Manual Switchovers

You can trigger a manual switchover in the HA node by using the ACLI **notify berpd force** command. This command forces the two SBCs in the HA node to trade roles. The active system becomes standby, and the standby becomes active.

In order to perform a successful manual switchover, the following conditions must be met.

- The SBC from which you trigger the switchover must be in one of the following states: active, standby, or becoming standby.
- A manual switchover to the active state is only allowed on a SBC in the standby or becoming standby state if it has achieved full media, signaling, and configuration synchronization.
- A manual switchover to the active state is only allowed on a SBC in the standby or becoming standby state if it has a health score above the value you configure for the threshold.

When you force a switch-over manually, the new active system displays a message - Standby to BecomingActive peer relinquishing control we're the healthiest.

Refer to the following example of a switchover log for an HA SBC that displays this message.

```
ORACLE2# Dec 17 16:38:08.321: Standby to BecomingActive peer relinquishing
control we're the healthiest
ORACLE2#
```

State Transitions

SBCs can experience a series of states as they become active or become standby.

> **Note:**
>
> Packet processing only occurs on an active SBC.

State	Description
Initial	When the SBC is booting.
Becoming Active	When the SBC has negotiated to become the active system, but is waiting the time that you set to become fully active. Packets cannot be processed in this state.
Active	When the SBC is handling all media, signaling, and configuration processing.
Relinquishing Active	When the SBC is giving up its Active status, but before it has become standby. This state is very brief.
Becoming Standby	When the SBC is becoming the standby system but is waiting to become fully synchronized. It remains in this state for the period of time you set in the becoming-standby-time parameter, or until it is fully synchronized.
Standby	When the SBC is fully synchronized with its active system in the HA node.
OutOfService	When the SBC cannot become synchronized in the period of time you set in the becoming-standby-time parameter.

State Transition Sequences

When the active Oracle Communications Session Border Controller assumes its role as the as the active system, but then changes roles with the standby Oracle Communications Session Border Controller to become standby, it goes through the following sequence of state transitions:

- Active
- RelinquishingActive
- BecomingStandby
- Standby
  When the standby Oracle Communications Session Border Controller assumes its role as the standby system, but then changes roles with the active Oracle Communications Session Border Controller to become active, it goes through the following sequence of state transitions:
- Standby
- BecomingActive
- Active

HA Features

HA nodes support configuration checkpointing, which you are required to set up so that the configurations across the HA node are synchronized. In addition, you can set up the following optional HA node features:

- Multiple rear interface support
- Gateway link failure detection and polling

Multiple Rear Interfaces

Configuring your HA node to support multiple rear interfaces eliminates the possibility that either of the rear interfaces you configure for HA support will become a single point of failure. Using this feature, you can configure individual SBCs with multiple destinations on the two rear interfaces, creating an added layer of failover support.

When you configure your HA node for multiple rear interface support, you can use last two rear interfaces (wancom1 and wancom2) for HA—the first (wancom0) being used for SBC management. You can connect your SBCs using any combination of wancom1 and wancom2 on both systems. Over these rear interfaces, the SBCs in the HA node share the following information:

- Health
- Media flow
- Signaling
- Configuration

For example, if one of the rear interface cables is disconnected or if the interface connection fails for some other reason, all health, media flow, signaling, and configuration information can be checkpointed over the other interface.

Health information is checkpointed across all configured interfaces. However, media flow, signaling, and configuration information is checkpointed across one interface at a time, as determined by the SBC's system HA processes.

Configuration Checkpointing

During configuration checkpointing, all configuration activity and changes on one SBC are automatically mirrored on the other. Checkpointed transactions include adding, deleting, or modifying a configuration on the active SBC. This means that you only need to perform configuration tasks on the active SBC because the standby system will go through the checkpointing process and synchronize its configuration to reflect activity and changes.

Because of the way configuration checkpointing works, the ACLI **save-config** and **activate-config** commands can only be used on the active SBC.

- When you use the ACLI **save-config** command on the active SBC, the standby SBC learns of the action and updates its own configuration. Then the standby SBC saves the configuration automatically.
- When you use the ACLI **activate-config** command on the active SBC, the standby SBC learns of the action and activates its own, updated configuration.

The ACLI **acquire-config** command is used to copy configuration information from one SBC to another.

Gateway Link Failure Detection and Polling

In an HA node, the Oracle Communications Session Border Controllers can poll for and detect media interface links to the gateways as they monitor ARP connectivity. The front gateway is assigned in the network interface configuration, and is where packets are forwarded out of the originator’s LAN.

The Oracle Communications Session Border Controller monitors connectivity using ARP messages that it exchanges with the gateway. The Oracle Communications Session Border

Controller sends regular ARP messages to the gateway in order to show that it is still in service; this is referred to as a heartbeat message. If the Oracle Communications Session Border Controller deems the gateway unreachable for any of the reasons discussed in this section, a network-level alarm is generated and an amount you configure for this fault is subtracted from the system's health score.

The Oracle Communications Session Border Controller generates a gateway unreachable network-level alarm if the Oracle Communications Session Border Controller has not received a message from the media interface gateway within the time you configure for a heartbeat timeout. In this case, The Oracle Communications Session Border Controller will send out ARP requests and wait for a reply. If no reply is received after resending the set number of ARP requests, the alarm remains until you clear it. The health score also stays at its reduced amount until you clear the alarm.

When valid ARP requests are once again received, the alarm is cleared and system health scores are increased the appropriate amount.

You can configure media interface detection and polling either on a global basis in the SD HA nodes/redundancy configuration or on individual basis for each network interface in the network interface configuration.

> **Note:**
>
> To improve the detection of link failures, the switchport connected to the NIU should have Spanning Tree disabled. Enabling Spanning Tree stops the switchport from forwarding frames for several seconds after a reset. This prevents the NIU from reaching the gateway and generates a "gateway unreachable" network-level alarm.

Georedundant High Availability (HA)

You can locate the two nodes that make up an HA pair in different locations from one another. This is known as georedundancy, which increases fault tolerance. A georedundant pair must adhere to rigid network operating conditions to ensure that all state and call data is shared between the systems, and that failovers happen quickly without losing calls.

The following network constraints are required for georedundant operation:

- A pair of dedicated fiber routes between sites is required. Each route must have non-blocking bandwidth sufficient to connect wancom1 and wancom2 ports (i.e., 1Gbps per port)
- Inter-site round-trip time (RTT) must be less than 10 ms. 5 ms or less is ideal. Georedundant operation must be built upon a properly engineered layer-2 WAN (eg. MPLS or Metro Ethernet) that connects active and standby HA pair members.
- Simultaneous packet loss across the inter-site link pair must be 0%. Loss of consecutive heartbeats could potentially result in split-brain behaviors.
- Security (privacy and data-integrity) must be provided by the network itself.

As with local HA nodes, management traffic (e.g. SSH, SFTP, SNMP, etc.) must be confined to the wancom0 management interface. HA node peers must have their wancom0 IP addresses on the same subnet. All Oracle Communications Session Border Controller configuration, including host routes and the system-config's **default-gateway**, is shared between the HA pair so it is not possible to have two different management interface default-gateways. This implies the requirement of an L2-switched connection between the 2 wancom0 management interfaces.

Troubleshooting Georedundant Deployments

The Oracle Communications Session Border Controller provides rich statistics and status information on HA operation, documented in the *ACLI Reference* and *Maintenance and Troubleshooting Guides*. Some of this information is especially suited for troubleshooting the latency and packet-loss requirements for georedundant deployments, including:

- Details within the **show redundancy** command output, including:
  - Request-response round-trip time measurements (**show redundancy <task-name>**)
  - Request-response loss measurements (**show redundancy <task-name>**)
  - journal statistics (**show redundancy <task-name> journals**)
  - journal latency (**show redundancy <task-name> journals**)
  - protocol-specific redundancy actions (**show redundancy <task-name> actions**)
  - protocol-specific redundancy objects (**show redundancy <task-name> objects**)
- Details within the **show queues** command output, including:
  - sipd command queue statistics (**show queue <task-name> commands**)
- Protocol-specific log messaging

Before Configuring a High Availability (HA) Pair

> **Note:**
>
> When you configure an HA pair, you must use the same password for both Oracle Communications Session Border Controllers.
> Before configuring the parameters that support HA, complete the following steps.

1. Establish the physical connections between the Oracle Communications Session Border Controllers. Avoid breaking the physical link (over the rear interfaces) between the Oracle Communications Session Border Controllers in an HA node. If the physical link between the Oracle Communications Session Border Controllers breaks, they will both attempt to become the active system and HA will not function as designed.
2. Confirm that both Oracle Communications Session Border Controllers are set to the same time. Use the ACLI **show clock** command to view the system time. If the Oracle Communications Session Border Controllers show different times, use the **system-timeset** command to change the time.

   Oracle recommends that you use NTP to synchronize your Oracle Communications Session Border Controllers, so that they have a common stratum time source.
3. HA nodes use specific ports for HA interfaces. See the documentation for the hardware that you use.
4. For ACLI configuration, you need to know the target names of the Oracle Communications Session Border Controllers making up the HA node. The target name of the system is reflected in the ACLI's system prompt. For example, in the ORACLE# system prompt, ORACLE is the target name.

   You can also see and set the target name in the Oracle Communications Session Border Controller boot parameters.

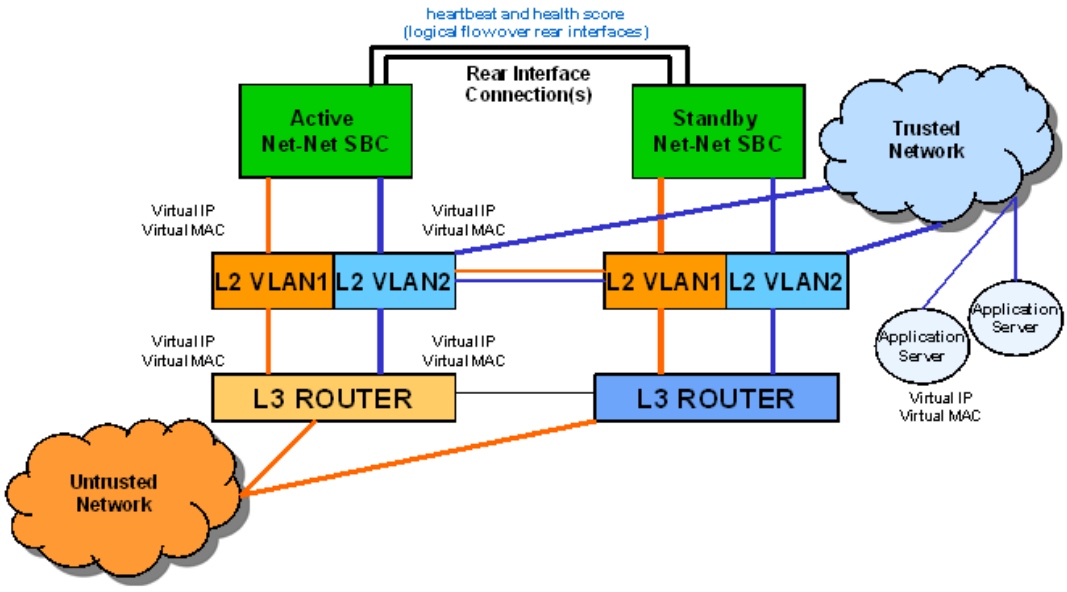
> **Note:**
>
> The target name is case sensitive.

5. Devise virtual MAC addresses so that, if a switchover happens, existing sessions are not interrupted.

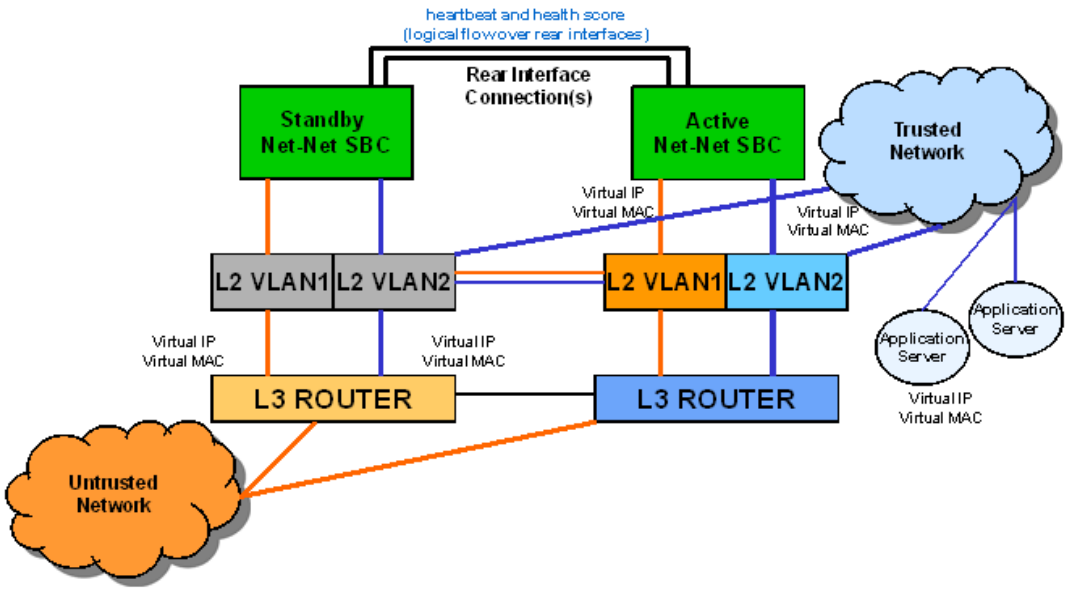
HA Node Connections

To use High Availability (HA), you must establish Layer 2 and Layer 3 networks that interconnect two Oracle Communications Session Border Controllers (SBC) and support HA with the required physical network connections. The basic network set-up in the following diagram shows an HA node deployment where each system is connected to its own Layer 2 switch. This set-up provides a measure of added redundancy in the event that one of the switches fails.

Here, the active system is using the virtual MAC and IP addresses.



In the second diagram, the same network is shown with the HA node having experienced a switchover. The previously standby SBC has taken over the active role in the HA node and is using the virtual IP and MAC addresses.



> **Note:**
>
> Switches should never be in primary-secondary mode. If they are, HA will not work correctly.

The following are hardware set-up and location considerations for placing an HA Node:

- You must set up each SBC according to the requirements and safety precautions set out in the *Oracle Communications System Hardware Installation Guide*.
- Each SBC's media interfaces must be connected to the same switches (or other network entities), as shown in the diagram above.
- The length of the shielded crossover 10/100 category 5 Ethernet cable that connects the SBCs from the rear interfaces must be able to reach from the configured rear interface on one Oracle Communications Session Border Controller to the configured rear interface on the other.

HA nodes use Oraclerder element redundancy protocol for its tasks. This protocol uses a connection between the rear interfaces of two SBCs to checkpoint the following information: health, state, media flow, signaling, and configuration.

We recommend that you use shielded category 5 (RJ45) crossover cables for all 10/100 Ethernet connections used for HA.

You can set up either single or multiple rear interface support for your HA node. For single interface support, one cable connects the two SBCs; for multiple interface support, two cables are used. However, the software configurations for each type of connection mode are different.

When you make these connections, do not use port 0 (wancom0) on the rear interface of the SBC chassis; that port should only be used for SBC management. Instead, use ports 1 and 2 (wancom1 and wancom2).

To cable Oracle Communications Session Border Controllers using single rear interface support:

1. Using a 10/100 category 5 crossover cable, insert one end into either port 1 (wancom1) or port 2 (wancom2) on the rear interface of the first SBC.
2. Insert the other end of the cable into port 1 or port 2 on the rear interface of the second SBC. We recommend that you use corresponding ports on the two systems. That is, use port 1 on both systems or use port 2 on both systems.
3. Perform software configuration for these interfaces as described in this chapter.

   To cable Oracle Communications Session Border Controllers using multiple rear interface support:

4. Using a 10/100 category 5 crossover cable, insert one end into port 1 on the rear interface of the first SBC.
5. Insert the other end of that cable into port 1 on the rear interface of the second SBC to complete the first physical connection.
6. Using a second 10/100 category 5 cable, insert one end into port 2 on the rear interface of the first SBC.
7. Insert the other end of this second cable in port 2 on the rear interface of the second SBC to complete the second physical connection.
8. Perform software configuration for these interfaces as described in this chapter.

Virtual MAC Addresses

In order to create the HA node, you need to create virtual MAC addresses for the media interfaces. You enter these addresses in virtual MAC address parameters for phy-interface configurations where the operation type for the interface is media.

The HA node uses shared virtual MAC (media access control) and virtual IP addresses for the media interfaces. When there is a switchover, the standby Oracle Communications Session Border Controller sends out an ARP message using the virtual MAC address, establishing that MAC on another physical port within the Ethernet switch. Virtual MAC addresses are actually unused MAC addresses that based on the Oracle Communications Session Border Controller's root MAC address.

The MAC address is a hardware address that uniquely identifies each Oracle Communications Session Border Controller. Given that, the virtual MAC address you configure allows the HA node to appear as a single system from the perspective of other network devices. To the upstream router, the MAC and IP are still alive, meaning that existing sessions continue uninterrupted through the standby Oracle Communications Session Border Controller.

Depending on the type of physical layer cards you have installed, you can create MAC addresses as follows: Four Ethernet (MAC) address for each configured four-port GigE physical interface card.

Virtual MAC Address Configuration

To create a virtual MAC address:

1. Determine the Ethernet address of the Oracle Communications Session Border Controller by using the ACLI **show interfaces** command. This command only works if you have already set up phy-interface configurations. Otherwise, you will get no output.

   The example below shows you where the Ethernet address information appears; this sample has been shortened for the sake of brevity. For each type of physical interface card, the Oracle Communications Session Border Controller displays the following:

   ```
   ORACLE# show interfaces
   f00 (media slot 0, port 0)
        Flags: UP BROADCAST MULTICAST ARP RUNNING
        Type: GIGABIT_ETHERNET
        Admin State: enabled
        Auto Negotiation: enabled
        Internet address: 10.10.0.10     Vlan: 0
        Broadcast Address: 10.10.255.255
        Netmask: 0xffff0000
        Gateway: 10.10.0.1
        Ethernet address is 00:08:25:01:07:64
   ```

2. Identify the root portion of the Ethernet (MAC) address.

   Each Oracle Communications Session Border Controller has MAC addresses assigned to it according to the following format: 00:08:25:XX:YY:ZN where:

   - 00:08:25 refers to Acme Packet
   - XX:YY:ZN refers to the specific Oracle Communications Session Border Controller
   - N is a 0-f hexadecimal value available for the Oracle Communications Session Border Controller

     In this example, the root part of this address is 00:08:25:XX:YY:Z.

3. To create an unused MAC address (that you will use as the virtual MAC address) take the root MAC address you have just identified. Replace this N value with unused hexadecimal values for the Oracle Communications Session Border Controller: 8, 9, e, or f.

   In other words, you change the last digit of the MAC address to either 8, 9, e, or f depending on which of those address are not being used.

   For example, for an HA node with MAC address bases of 00:08:25:00:00:00 and 00:08:25:00:00:10, the following addresses would be available for use at virtual MAC addresses:

   - 00:08:25:00:00:08
   - 00:08:25:00:00:09
   - 00:08:25:00:00:0e
   - 00:08:25:00:00:0f
   - 00:08:25:00:00:18
   - 00:08:25:00:00:19
   - 00:08:25:00:00:1e
   - 00:08:25:00:00:1f

     Corresponding media interfaces in HA nodes must have the same virtual MAC addresses. Given that you have various physical interface card options, the following points illustrate how virtual MAC address can be shared:

If you are using a four-port GigE physical interface card, both the active Oracle Communications Session Border Controllerand the standby Oracle Communications Session Border Controller might have the following virtual MAC address scheme for the slots:

- Slot 0 _ 00:08:25:00:00:0e and 00:08:25:00:00:0f
- Slot 1 - 00:08:25:00:00:1e and 00:08:25:00:00:1f

> **Note:**
>
> Note the virtual MAC addresses you have created so that you can reference them easily when you are configuring the phy-interfaces for HA.

Virtual MAC Addresses for VNFs

Virtual Network Functions (VNFs) rely on their hypervisor environment for MAC address establishment, advertisement and resolution. As such, you cannot derive these addresses using the same method as you do for Acme platforms. For VNFs, Oracle recommends establishing private MAC addressing for virtual MAC address configuration.

To support HA, you configure virtual Ethernet (MAC) address MAC addresses based on the Burned In Addresses (BIA) of the media interfaces. To determine what the virtual MAC addresses should be, you first identify a BIA and then calculate the virtual MACs based on that.

To define the virtual addresses you need to configure for each interface:

1. Identify the base MAC of eth0/wancom0 physical interface using the show interfaces command. For example, in the following display, you can see the base MAC is 00:50:56:C0:00:08:

```
eth(unit number 0):
Flags: (0x78843) UP BROADCAST MULTICAST ARP RUNNING INET_UP
Type: ETHERNET_CSMACD
inet: 111.22.0.123
Broadcast address: 111.22.255.255
Netmask 0xffff0000 Subnetmask 0xffff0000
Ethernet address is 00:50:56:C0:00:08
```

2. Set the bottom nibble of the first byte to 2 to define the address as locally administered.
3. Set the top nibble of the first byte to 0 and increment it for each interface.

For example, using the base-MAC for eth0, 00:50:56:C0:00:08, you assign the virtual addresses as follows:

- First media interface virtual MAC = 02:50:56:C0:00:08
- Second media interface virtual MAC = 12:50:56:C0:00:08
- Third media interface virtual MAC = 22:50:56:C0:00:08
- Forth media interface virtual MAC = 32:50:56:C0:00:08

HA Node Connections

You can begin software configuration for your HA node after you have:

- Completed the steps for physical set-up and connection.
- Noted the target name of the Oracle Communications Session Border Controllers that make up the HA node.
- Configured the virtual MAC addresses that you need, according to the type of physical interface cards installed on your Oracle Communications Session Border Controller.

HA Node Connection Configuration

If you are using HA, you need to set the phy-interface configuration parameters described in this section to establish successful connections. These parameters are for rear and media interfaces.

To access the phy-interface menu in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **phy-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# phy-interface
   ORACLE(phy-interface)#
   ```

   From this point, you can configure phy-interface parameters. To view all phy-interface parameters, enter a **?** at the system prompt.

Rear Interfaces

You can use port 1 (wancom1) or port 2 (wancom2) as interfaces to support HA. Do not use port 0 (wancom 0) as that port is reserved for carrying management traffic.

Make sure that the physical connections you have made on the rear panel of your Oracle Communications Session Border Controllers correspond to the configurations you enter for phy-interfaces. You can connect Oracle Communications Session Border Controllers through multiple rear interfaces. For multiple rear interface connectivity, cable both port 1 and port 2 (wancom1 and wancom2) on one Oracle Communications Session Border Controller to port1 and port 2 on the other Oracle Communications Session Border Controller in the HA node.

The Oracle Communications Session Border Controller's HA function depends heavily on health scores to determine the active and standby roles in an HA node. You can set the amount that will be subtracted from a Oracle Communications Session Border Controller's health score in the event that a management interface fails for any reason. For example, a connection might become invalid or a cable might be removed inadvertently.

The following example shows how a configured phy-interface will appear in the ACLI for an HA node:

```
phy-interface
        name                            wancom1
        operation-type                  Control
```

```
port                          1
slot                          0
virtual-mac
wancom-health-score           20
```

To establish rear interfaces for use in an HA node using the ACLI:

1. Access the phy-interface menu.
2. **name**—Set a name for the interface using any combination of characters entered without spaces. For example: wancom1.
3. **operation-type**—Set this parameter to **Control**.
4. **slot**—Set this parameter to **0**.
5. **port**—Set this parameter to **1** or **2**.
6. **wancom-health-score**—Enter the number value between 0 and 100. This value will be subtracted from the Oracle Communications Session Border Controller's health score in the event that a rear interface link fails. We recommend that you change this value from its default (**50**), and set it to **20**.

   This value you set here is compared to the active and emergency health score thresholds you establish in the Oracle Communications Session Border Controller HA node (redundancy) configuration.

   This parameter has no effect on a **phy-interface** set to **Media** as its **operation-type**.
7. For multiple rear interface support, configure the remaining, unused rear interfaces with the appropriate values.

   The following example shows configuration for multiple rear interface support.

```
ORACLE(system)# phy-interface
ORACLE(phy-interface)# name wancom1
ORACLE(phy-interface)# operation-type control
ORACLE(phy-interface)# port 1
ORACLE(phy-interface)# wancom-health-score 20
ORACLE(phy-interface)# done
ORACLE(phy-interface)# name wancom2
ORACLE(phy-interface)# operation-type control
ORACLE(phy-interface)# port 2
ORACLE(phy-interface)# wancom-health-score 20
ORACLE(phy-interface)# done
```

Media Interface Virtual MAC Addresses

To configure HA for the media interfaces in an HA node, you must set one or more virtual MAC addresses, according to the type of physical layer cards you have installed on your Oracle Communications Session Border Controller.

To set a virtual MAC address using the ACLI:

1. Access the phy-interface configuration.
2. Configure all relevant parameters as noted in the Phy-Interfaces section of this guide's *System Configuration* chapter.

   Since virtual MAC addresses are used for media interfaces only, verify that the operation type is set to media.

3. **virtual-mac**—Enter the virtual MAC address that you have created using the steps in the Virtual MAC Addresses section.

Alarms and Traps for Wancom1 and 2 Interfaces

You can configure the SBC to trigger alarms and traps when your HA wancom interfaces, including wancom1 and/or wancom2, go down. This allows you to more quickly recognize disruptions and re-establish your HA deployment's functionality.

HA systems synchronize themselves with each other regularly and often, providing short windows within which they can detect link failures, often caused by the switching infrastructure between them. When you configure this feature, the SBC monitors this synchronization and raises an alarm when synchronization fails. To detect synchronization failures in wancom links, the system utilizes a ping mechanism. Using the configured ping interval and retry settings, the system calculates the wait time before raising an alarm if it does not receive a response to the ping message.

You configure two parameters within your **redundancy** to enable and tune this feature, including:

- **wancom-ping-interval**—Sets the time interval in milleseconds between transmitting wancom ping messages. This parameter is disabled (0) by default.
- **wancom-ping-retry**—Sets the number of times the systems exchange ping retry messages between the wancom links before it raises an alarm and trap. A failed retry is defined as the active receiving no response from the standby.
  After ping retries expire, the system continues to exchange ping messages using the interval configuration. When the active receives a ping response again, the system clears the alarm.

```
ORACLE(redundancy)#wancom-ping-interval 50
ORACLE(redundancy)#wancom-ping-retry 5
```

The system calculates the wait period by multiplying the (**wancom-ping-interval**) times the retry times (**wancom-ping-retry**). When this timer expires, the active SBC raises the alarm.

When the active node beings to receive synchronization responses again, the SBC clears the alarm for the operational wancom interface. Note that you can also clear the alarm manually using the **clear-alarm** command.

Reporting

You can run the **show redundancy config** to monitor the round trip time of the wancom ping messages' request-response times on the Active node. You can refer to these values to identify incorrect wancom connectivity failure alarms. These delays can be caused by delays in network rather than link failures. Refer to the Request-Response Loss and the HW Timing Distribution values at the bottom of this output to determine whether an alarm was raised because of network delays rather than link failures.

```
ORACLEHA1# show redundancy config
11:10:44-198
Redundancy Statistics         -- Period -- -------- Lifetime --------
                    Active    High   Total      Total  PerMax     High
Queued Entries           0       0       0          3       2        2
Red Records              0       0       0          3       2        2
Records Dropped          -       -       0          0       0
Server Trans             9       9      83    4691627      49       16
```

```
Client Trans             0       0       0         20       20       1

11:10:44-198
Redundancy Transactions          ---- Lifetime ----
                          Recent      Total  PerMax
Requests received             83    4691627      49
Duplicate requests             0          0       0
Success responses             83    4691626      49
Error responses                0          1       1
Request sent                   0         20      20
Retransmissions sent           0          0       0
Success received               0         18      18
Errors received                0          1       1
Transaction timeouts           0          0       0
Avg Latency=0.000 for 0
Max Latency=0.000
Last redundant transaction processed: 3

11:10:44-198
Request-Response Loss %       ---- Lifetime ----
                        Recent      Total PerMax
                             0          0      0

11:10:44-198
HA  timing distribution:    0ns to    2ms to    4ms to    8ms to   16ms to
33ms to   67ms to
=======================        2ms       4ms       8ms      16ms
33ms      67ms       ---

Transaction Latency
   Request-Response RTT:         7         2         3         2
3         2         0
```

RTT is the time from which the active issues a wancom ping request and it receives a response. When RTT is long, due to a sluggish network, the system may issue false alarms. When referring to RTT values, note that low counts in higher latency ranges suggest good performance, while high counts might indicate network issues.

Recommended Configuration

Oracle recommends you configure your wancom ping settings having considered the following:

- Recommended values to configure **wancom-ping-interval** and **wancom-ping-retry** for raising wancom sync failure alarms/traps include 100 milliseconds and 2 retries, respectively. With these values, the SBC sends the ping messages at a frequency of 100 msec to each wancom interface and raises the alarm(s) if two consecutive messages from either or both wancom interface do not receive a response.
- Evaluate and consider other network factors, such as delay and round-trip time (RTT), when configuring these values.
- Adjusting the **wancom-ping-interval** to be more aligned with the advertisement-time allows for a quicker response to any synchronization failures because this reduces the time to identify and address them. Oracle recommends you keep the **wancom-ping-interval** lower than half of the ‘advertisement-time’ for more effective monitoring and timely detection of failures.

- Do not configure these parameters with very low values to minimize the number of transmissions. Choose your parameter values based on a comprehensive assessment of network conditions and requirements.

Alarms

The SBC raises the following alarms within the context of this feature:

- APP_WANCOM1_SYNC_FAILURE (327752)—Alarm is raised when wancom1 link between active and standby SBC is down.
- APP_WANCOM2_SYNC_FAILURE (327753)—Alarm is raised when wancom2 link between active and standby SBC is down.

These alarms will be logged in log.brokerd and acmelog files in /opt/logs/.

```
ID      Task  Severity    First Occurred        Last Occurred
327753  2800      4       2024-03-01 07:48:43   2024-03-01 07:48:43
Count Description
1 Sync failure for Wancom2
```

The SBC clears these alarms when the criteria that caused the alarm clears.

Traps

Assuming you have configured SNMP, the SBC issues the apWancomSyncFailTrap and apWancomSyncFailClearTrap traps at the same time as the synchronization failure alarms. The SBC sends the apWancomSyncFailTrap trap, along with the alarm, when the connection between either wancom1 or 2 fails. Similarly, the SBC sends the apWancomSyncFailClearTrap when the connection resumes. These traps include details about the applicable wancom interface. The numeric OIDs for these traps are:

- apWancomSyncFailTrap = 1.3.6.1.4.1.9148.3.16.2.2.7.0.1
- apWancomSyncFailClearTrap = 1.3.6.1.4.1.9148.3.16.2.2.7.0.2

The system manages these traps from the ap-apps.mib file.

HA Node Parameters

To establish a pair of Oracle Communications Session Border Controllers as an HA node, you need to configure basic parameters that govern how the Oracle Communications Session Border Controllers:

- Transition on switchover
- Share media and call state information
- Checkpoint configuration data

The following example shows what an HA configuration might look like in the ACLI.

```
redundancy-config
        state                                   enabled
        log-level                               INFO
        health-threshold                        75
        emergency-threshold                     50
        port                                    9090
        advertisement-time                      500
        percent-drift                           210
```

```
wancom-ping-interval                    0
wancom-ping-retry                       2
initial-time                            1250
becoming-standby-time                   180000
becoming-active-time                    100
cfg-port                                1987
cfg-max-trans                           10000
cfg-sync-start-time                     5000
cfg-sync-comp-time                      1000
gateway-heartbeat-interval              0
gateway-heartbeat-retry                 0
gateway-heartbeat-timeout               1
gateway-heartbeat-health                0
media-if-peercheck-time                 0
options
```

You need to configure the two Oracle Communications Session Border Controllers to be HA node peers. To enable configuration checkpointing, you must to configure two peers in the ACLI, one for the primary and one for the secondary Oracle Communications Session Border Controller. The HA node peers configuration also allows you to configure destinations for where to send health and state information. Unless you create Oracle Communications Session Border Controller peers and destinations configurations, HA will not work properly.

The following example shows what an HA configuration might look like in the ACLI.

```
peer
                name                            netnetsd1
                state                           enabled
                type                            Primary
                destination
                        address                         169.254.1.1:9090
network-interface               wancom1:0
peer
                name                            netnetsd2
                state                           enabled
                type                            Secondary
                destination
                        address                         169.254.1.2:9090
                        network-interface               wancom1:0
```

HA Node Parameter Configuration

To configure general HA node parameters using the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **redundancy** and press Enter.

```
ORACLE(system)# redundancy
```

From here, you configure basic HA node parameters. To view all basic HA node parameters, enter a **?** at the system prompt.

4. **state**—Leave this parameter set to **enabled** for HA to work. To stop HA operation, set this parameter to **disabled**. The default value is **enabled**. The valid values are:
   - enabled | disabled
5. **log-level**—Set the log level you want to use for the HA system process. The value you set in this field overrides any log level value you set for the entire Oracle Communications Session Border Controller in the system configuration process log level parameter. The default value is **INFO** which allows you to receive a moderate amount of detail. The valid values are:
   - emergency | critical | major | minor | warning | notice | info | trace | debug | detail
6. **health-threshold**—Enter a value between 0 and 100 to set the health score at which the Oracle Communications Session Border Controllers in the HA node gracefully exchange active-standby roles. The default value is **75**. The valid range is:
   - Minimum—1
   - Maximum—100

     For example, if this field is set to **75** and the active Oracle Communications Session Border Controller's health score falls below that point, the standby Oracle Communications Session Border Controller will take over the active role. However, Oracle Communications Session Border Controller will only take over the active role if its own health score is 75 or better.
7. **emergency-threshold**—Enter the health score for the standby Oracle Communications Session Border Controller to become active immediately. The default value is **50**. The valid range is:
   - Minimum—0
   - Maximum—100

     If the standby Oracle Communications Session Border Controller is initializing and the active Oracle Communications Session Border Controller's health score is below the health threshold, the standby Oracle Communications Session Border Controller will take the active role and there will be a graceful switchover. If the active Oracle Communications Session Border Controller's health score is below the emergency threshold, then the switchover will be immediate.

     If the standby Oracle Communications Session Border Controller has a health score below the emergency threshold and the active Oracle Communications Session Border Controlleris unhealthy, the active Oracle Communications Session Border Controller will not give up its active role.
8. **advertisement-time**—Enter the number of milliseconds to set how often Oracle Communications Session Border Controllers in an HA node inform each other of their health scores.

   We recommend you leave this parameter set to it's default, **500**. The valid range is:
   - Minimum—50
   - Maximum—999999999

9. **percent-drift**—Enter the percentage of the advertisement time that you want one member of the HA node to wait before considering the other member to be out of service. For the standby Oracle Communications Session Border Controller, this is the time it will wait before taking the active role in the HA node. The default value is **210**. The valid range is:
   - Minimum—100
   - Maximum—65535
10. **wancom-ping-interval**—Set the time between transmitting wancom ping messages (milliseconds).
   - Default: 0 (Disabled)
   - Values to enable: Min: 40 | Max: 999999999 ms

   > **Note:**
   >
   > This value must be less than your advertisement-time value.

11. **wancom-ping-retry**—Set the number of times the system generates wancom ping retries after there has been no response.
   - Default: 2
   - Values: Min: 1 | Max: 655

   > **Note:**
   >
   > This value must be less than or equal to percent-drift/100.

12. **initial-time**—Enter the number of milliseconds to set the longest amount of time the Oracle Communications Session Border Controller will wait at boot time to change its state from initial to either becoming active or becoming standby. The default value is **1250**. The valid range is:
   - Minimum—5
   - Maximum—999999999
13. **becoming-standby-time**—Enter the number of milliseconds the Oracle Communications Session Border Controller waits before becoming standby, allowing time for synchronization. If it is not fully synchronized within this time, it will be declared out of service.

   We recommend that you do not set this parameter below **45000**. If a large configuration is being processed, we recommend setting this parameter to **180000** to allow enough time for configuration checkpointing. The default value is **45000**. The valid range is:
   - Minimum—5
   - Maximum—999999999
14. **becoming-active-time**—Enter the number of milliseconds that the standby Oracle Communications Session Border Controller takes to become active in the event that the active Oracle Communications Session Border Controller fails or has an intolerably decreased health score. The default value is **100**. The valid range is:
   - Minimum—5
   - Maximum—999999999

HA Node Peer Configuration

To configure a Oracle Communications Session Border Controller as an HA node peer:

1. From the redundancy menu, type **peers** and press Enter.

```
ORACLE(system)# redundancy
ORACLE(redundancy)# peers
```

2. **state**—Enable or disable HA for this Oracle Communications Session Border Controller. The default value is **enabled**. The valid values are:
   - enabled | disabled
3. **name**—Set the name of the HA node peer as it appears in the target name boot parameter.

   This is also the name of your system that appears in the system prompt. For example, in the system prompt ORACLE1#, ORACLE1 is the target name for that Oracle Communications Session Border Controller.
4. **type**—These values refer to the primary and secondary utility addresses in the network interface configuration. To determine what utility address to use for configuration checkpointing, set the type of Oracle Communications Session Border Controller: primary or secondary.

> **Note:**
>
> You must change this field from unknown, its default. The valid values are:

   - **primary**—Set this type if you want the Oracle Communications Session Border Controller to use the primary utility address.
   - **secondary**—Set this type if you want the Oracle Communications Session Border Controller to use the secondary utility address.
   - **unknown**—If you leave this parameter set to this default value, configuration checkpointing will not work.

HA Node Health And State Configuration

To configure where to send health and state information within an HA node:

1. From the peers configuration, type destinations and press Enter.

```
ORACLE(rdncy-peer)# destinations
ORACLE(rdncy-peer-dest)#
```

2. **address**—Set the destination IPv4 address and port of the other Oracle Communications Session Border Controller in the HA node to which this Oracle Communications Session Border Controller will send HA-related messages. This value is an IPv4 address and port combination that you enter as: IPAddress:Port. For example, 169.254.1.1:9090.
   - The IPv4 address portion of this value is the same as the IPv4 address parameter set in a network interface configuration of the other Oracle Communications Session Border Controller in the HA node.

- The port portion of this value is the port you set in the Oracle Communications Session Border Controller HA Node/redundancy configuration for the other Oracle Communications Session Border Controller in the node.

3. **network-interface**—Set the name and subport for the network interface where the Oracle Communications Session Border Controller receives HA-related messages. Valid names are wancom1 and wancom2. This name and subport combination must be entered as name:subport; for example, **wancom1:0**.

   The network interface specified in this parameter must be linked to a phy-interface configured with rear interface parameters. The phy-interface's operation type must be control or maintenance, and so the subport ID portion of this parameter is 0. The subport ID is the VLAN tag.

Synchronizing Configurations

You can synchronize the Oracle Communications Session Border Controllers (SBC) in your High Availability (HA) node in the following ways:

- Automatically — Set up configuration checkpointing within the HA node.
- Manually — Check whether or not configurations in the HA node are synchronized, and then copy configuration data from one SBC to the other.

When you initially configure a new HA node, copy the configuration data manually from one SBC to the other. When you complete the process, you can configure your HA node to automatically synchronize configurations.

Oracle recommends that you configure the HA node for configuration checkpointing because that is the most reliable way to ensure that both systems have the same configuration.

Synchronize HA Peers

The process for synchronizing the peers in a High Availability (HA) node for the first time by way of the ACLI includes the following steps.

1. Create a complete configuration on the active Oracle Communications Session Border Controller (SBC). Include all HA node parameters and all rear interface configurations. Confirm that the rear interfaces are configured to send and receive information across the HA node.
2. On the active SBC, save the configuration.
3. On the active SBC, reboot to run the new configuration.

   Use the ACLI **show health** command to see that the active SBC booted without a peer. This changes after you copy the configuration to the standby SBC and activate the configuration.

4. On the standby SBC, perform the ACLI **acquire-config** command to copy the configuration from the active SBC. Use the **acquire-config** command with the IPv4 address of wancom 0 on the active SBC.

   ```
   ACMEPACKET2# acquire-config 192.168.12.4
   ```

The IPv4 address of wancom 0 on the active SBC is the IPv4 address portion of the value displayed for the **IP Address** boot parameter. The following codeblock shows an example of the**IP Address** value that the system displays when you view the boot parameters:

```
IP Address        : 192.168.12.4
```

5. When the copying process (**acquire-config**) is complete, reboot the standby SBC to activate the configuration. The system boots and displays start-up information.
6. Confirm that the HA node synchronized the configurations by using the ACLI **display-current-cfg-version** and **display-running-cfg-version** commands:

```
ORACLE# display-current-cfg-version
Current configuration version is 3
ORACLE# display-running-cfg-version
Running configuration version is 3
ORACLE# display-current-cfg-version
Current configuration version is 3
ORACLE# display-running-cfg-version
Running configuration version is 3
```

In the preceding example, all configuration versions—current and running—are the same number (3).

RTP Timestamp Synchronization

The Oracle Communications Session Border Controller maintains the continuity of egress transcoded media streams during HA switchover by synchronizing the RTP timestamps between active and standby systems.

For a new call, the transcoding resources are allocated and each session is configured with an initial RTP timestamp value. This process is repeated independently on both the active and standby systems to maintain approximately the same timestamps. This minimizes the difference between active and standby-side interpretation of the current RTP timestamp for a new session.

During HA operation, the active system maintains new timers that check for transcoded sessions lasting fifteen minutes or more. The active system re-synchronizes the RTP timestamp after fifteen minutes. This prevents the RTP timestamps from drifting due to clocking differences between active and standby hardware.

In addition, when the standby system boots, it performs a complete session sync with the active system for all currently active sessions.

Using Configuration Checkpointing

The Oracle Communications Session Border Controller's primary and secondary utility addresses support configuration checkpointing, allowing the standby Oracle Communications Session Border Controller to learn configuration changes from the active Oracle Communications Session Border Controller. This means that you only have to enter configuration changes on the active Oracle Communications Session Border Controller for the configurations across the HA node to be updated.

Configuration checkpointing uses parameters in the network interface and in the Oracle Communications Session Border Controller HA Nodes/redundancy configurations.

If you are using configuration checkpointing, you also need to set up two Oracle Communications Session Border Controller peer configurations: one the primary, and one for the secondary.

HA Configuration Checkpointing

You need to first set applicable network interface configuration parameters, and then establish applicable parameters in the Oracle Communications Session Border Controller HA node (redundancy) configuration.

We recommend that you do not change the configuration checkpointing parameters in the redundancy configuration. Using the defaults, this feature will function as designed.

> **Note:**
>
> Remember to set the appropriate type parameter in the HA node redundancy peers configuration.

For the network interface, these parameters appear as they do in the following example when you use the ACLI. This example has been shortened for the sake of brevity.

```
pri-utility-addr              169.254.1.1
sec-utility-addr              169.254.1.2
```

For the Oracle Communications Session Border Controller HA node (redundancy) configuration, these parameters appear as they do in the following example when you use the ACLI. This example has been shortened for the sake of brevity. You should not change these values without consultation from Oracle Technical Support or your Oracle Systems Engineer.

```
cfg-port                      1987
cfg-max-trans                 10000
cfg-sync-start-time           5000
cfg-sync-comp-time            1000
```

To configure HA configuration checkpointing in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **network-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# network-interface
   ORACLE(network-interface)#
   ```

   From here, you can configure network interface parameters. To view all network interfaces parameters, enter a **?** at the system prompt.

4. **pri-utility-addr**—Enter the utility IP address for the primary HA peer in an HA architecture.

   This address can be any unused IP address within the subnet defined for the network interface. For example, given a network interface of with the IPv4 address 168.0.4.15/24 (identifying the host associated with the network interface), the possible range of unused IPv4 addresses is 168.0.4.1 to 168.0.4.254. Your network administrator will know which IPv4 addresses are available for use.

5. **sec-utility-addr**—Enter the utility IP address for the secondary Oracle Communications Session Border Controller peer in an HA architecture.

   Usually, this IP address is usually the next in the sequence up from the primary utility address. It is also generated from the range of unused IP addresses within the subnet defined for the network interface.

6. Save your work and exit the network interface configuration.

```
ORACLE(network-interface)# done
ORACLE(network-interface)# exit
ORACLE(system)#
```

7. Access the system HA node/redundancy configuration by typing **redundancy** at the system prompt and then press Enter.

```
ORACLE(system)# redundancy
ORACLE(redundancy)#
```

> **Note:**
>
> We strongly recommend that you keep the default settings for the parameters Steps 8 through 11.

8. **cfg-port**—Enter the port number for sending and receiving configuration checkpointing messages. Setting this to zero (**0)** disables configuration checkpointing. The default value is **1987**. The valid values are:

   - Minimum—0, 1025
   - Maximum—65535

9. **cfg-max-trans**—Enter the number of HA configuration checkpointing transactions that you want to store. The active Oracle Communications Session Border Controller maintains the transaction list, which is acquired by the standby Oracle Communications Session Border Controller. Then the standby system uses the list to synchronize its configuration with active system. The default value is **10000**. The valid range is:

   - Minimum—0
   - Maximum—4294967295

     Transactions include: modifications, additions, and deletions. If the maximum number of stored transactions is reached, the oldest transactions will be deleted as new transactions are added.

10. **cfg-sync-start-time**—Enter the number of milliseconds before the Oracle Communications Session Border Controller tries to synchronize by using configuration checkpointing. On the active Oracle Communications Session Border Controller, this timer is continually reset as the Oracle Communications Session Border Controller checks to

see that it is still in the active role. If it becomes standby, it waits this amount of time before it tries to synchronize.

We recommend you leave this field at its default value, **5000**, so that configuration checkpointing can function correctly. The valid range is:

- Minimum—0
- Maximum—4294967295

11. **cfg-sync-comp-time**—Enter the number of milliseconds that the standby Oracle Communications Session Border Controller waits before checkpointing to obtain configuration transaction information after the initial checkpointing process is complete.

    We recommend you leave this field at its default value, **1000**, so that configuration checkpointing can function correctly. The valid range is:

    - Minimum—0
    - Maximum—4294967295

12. Save your work and exit the redundancy configuration.

```
ORACLE(redundancy)# done
ORACLE(redundancy)# exit
ORACLE(system)#
```

Manually Checking Configuration Synchronization

You can check that the current and active configurations are synchronized across the HA node. The current configuration is the one with which you are currently working, and the active configuration is the one active on the system.

To confirm that the systems in the HA node have synchronized configurations:

1. On the active Oracle Communications Session Border Controller in the Superuser menu, enter the following ALCI commands and press Enter. Note the configuration version numbers for comparison with those on the standby Oracle Communications Session Border Controller.

    - **display-current-cfg-version**—Shows the version number of the configuration you are currently viewing (for editing, updating, etc.).

```
ORACLE# display-current-cfg-version
Current configuration version is 30
```

    - **display-running-cfg-version**—Shows the version number of the active configuration running on the Oracle Communications Session Border Controller.

```
ORACLE# display-running-cfg-version
Running configuration version is 30
```

2. On the standby Oracle Communications Session Border Controller, enter the following ALCI commands and press Enter. Note the configuration version numbers for comparison with those on the active Oracle Communications Session Border Controller.

```
ORACLE# display-current-cfg-version
Current configuration version is 30
```

```
ORACLE# display-running-cfg-version
Running configuration version is 30
```

3. Compare the configuration numbers. If the version numbers on the active Oracle Communications Session Border Controller match those on the standby Oracle Communications Session Border Controller, then the systems are synchronized.

   If the version numbers do not match, you need to synchronize the Oracle Communications Session Border Controllers. You can do so using the ACLI **acquire-config** command.

Media Interface Link Detection and Gateway Polling

You can use media interface link detection and gateway polling globally on the Oracle Communications Session Border Controller, or you can override those global parameters on a per-network-interface basis.

- Use the Oracle Communications Session Border Controller HA node (redundancy) configuration to establish global parameters. When configured globally, they will appear like this in the ACLI:

```
gateway-heartbeat-interval      0
gateway-heartbeat-retry         0
gateway-heartbeat-timeout       1
gateway-heartbeat-health        0
```

- Use the network interface's gateway heartbeat configuration to override global parameters on a per-network-interface basis. When configured for the network interface, these parameters will appear like this in the ACLI:

```
gw-heartbeat
                state                           enabled
                heartbeat                       0
                retry-count                     0
                retry-timeout                   1
                health-score                    0
```

Media Interface Link Detection and Gateway Polling Configuration

To configure global media interface link detection and gateway polling:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **redundancy** and press Enter.

   ```
   ORACLE(system)# redundancy
   ```

   From here, you can configure gateway heartbeat parameters. To view all gateway heartbeat parameters, enter a **?** at the system prompt.

4. **gateway-heartbeat-interval**—Enter the number of seconds between heartbeats for the media interface gateway. Heartbeats are sent at this interval as long as the media interface is viable. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—65535
5. **gateway-heartbeat-retry**—Enter the number of heartbeat retries (subsequent ARP requests) to send to the media interface gateway before it is considered unreachable. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—65535
6. **gateway-heartbeat-timeout**—Enter the heartbeat retry time-out value in seconds. The default value is **1**. The valid range is:
   - Minimum—0
   - Maximum—65535

     This parameter sets the amount of time between Oracle Communications Session Border Controller ARP requests to establish media interface gateway communication after a media interface gateway failure.
7. **gateway-heartbeat-health**—Enter the amount to subtract from the Oracle Communications Session Border Controller's health score if a media interface gateway heartbeat fails. If the value you set in the gateway time-out retry field is exceeded, this amount will be subtracted from the system's overall health score. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—100

Media Interface Link Detection and Gateway Polling Configuration 2

To configure media interface link detection and gateway polling on a per-network-interface basis in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ACMEPACKET(configure)# system
   ```

3. Type **network-interface** and press Enter.

   ```
   ACMEPACKET(system)# network-interface
   ```

4. Type **gw-heartbeat** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(network-interface)# gw-heartbeat
   ACMEPACKET(gw-heartbeat)#
   ```

From here, you can configure gateway heartbeat parameters for the network interface. To view all gateway heartbeat parameters, enter a **?** at the system prompt.

5. **state**—Enable or disable the gateway heartbeat feature. The default value is **disabled**. The valid values are:
   - enabled | disabled
6. **heartbeat**—Enter the number of seconds between heartbeats for the media interface gateway. Heartbeats are sent at this interval as long as the media interface is viable. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—65535

     The value you configure in this field overrides any globally applicable value set in the gateway heartbeat interval parameter in the Oracle Communications Session Border Controller HA node (redundancy) configuration.
7. **retry-count**—Enter the number of heartbeat retries that you want sent to the media interface gateway before it is considered unreachable. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—65535
8. **retry-timeout**—Enter the heartbeat retry time-out value in seconds. The default value is **1**. The valid range is:
   - Minimum—1
   - Maximum—65535

     This parameter sets the amount of time between system ARP requests to establish media interface gateway communication after a media interface gateway failure.
9. **health-score**—Enter the amount to subtract from the system's health score if a media interface gateway heartbeat fails; this parameter defaults to 0. If the value you set in the retry-time-out field is exceeded, this amount will be subtracted from the system's overall health score. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—100

Signaling Checkpointing

You can configure your HA node to checkpoint signaling for SIP.

SIP Signaling Checkpointing

In the SIP configuration, you can set parameters that enable SIP signaling checkpointing across an HA node.

When configured, these parameters will appear in the ACLI as they do in example below.

> **Note:**
>
> This example shows the default values being used, and we recommend that you do not change these values from their defaults.

```
red-sip-port                    1988
red-max-trans                   10000
red-sync-start-time             5000
red-sync-comp-time              1000
```

Signaling Checkpointing Configuration

To configure SIP signaling checkpointing across an HA node in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-router** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ```

   From here, you can configure SIP parameters for HA nodes. To view all SIP configuration parameters, enter a **?** at the system prompt.

   When configuring SIP for HA, you only need to set the parameters addressed in this procedure.

4. **red-sip-port**—Enter the port on which SIP signaling checkpointing messages are sent and received. The default value is **1988**. A value of **0** disables the SIP signaling checkpointing. The valid range is:
   - Minimum—0, 1024
   - Maximum—65535
5. **red-max-trans**—Enter the maximum size of the transaction list, or how many SIP transactions you want to store in memory at one time. Oldest transactions will be discarded first in the event that the limit is reached. The default value is **10000**. The valid range is:
   - Minimum—0
   - Maximum—999999999
6. **red-sync-start-time**—Enter the number of milliseconds before the Oracle Communications Session Border Controller will try to synchronize its signaling state checkpointing.

   If the active Oracle Communications Session Border Controller is still adequately healthy, this timer will simply reset itself. If for any reason the active Oracle Communications Session Border Controller has become the standby, it will start to checkpoint with the newly active system when this timer expires.

We recommend that you leave this parameter set to its default, **5000**. The valid range is:

- Minimum—0
- Maximum—999999999

7. **red-sync-comp-time**—Enter the number of milliseconds representing how frequently the standby Oracle Communications Session Border Controller checkpointing with the active Oracle Communications Session Border Controller to obtain the latest SIP signaling information. The first interval occurs after initial synchronizations of the systems.

   We recommend that you leave this parameter set to its default, **1000**. The valid range is:

   - Minimum—0
   - Maximum—999999999

Media State Checkpointing

By default, the Oracle Communications Session Border Controller performs media checkpointing across the HA node for all signaling protocols. You can keep the default port set for redundancy media flows.

H.323 media high availability is supported through a TCP socket keep-alive, which determines whether or not the other end of a TCP/IP network connection is still in fact connected. This type of checkpointing prevents the listening side of a connection from waiting indefinitely when a TCP connection is lost. When there is a switchover in the HA node, the system that has just become active takes over sending TCP keep-alives. Media continues to flow until the session ends or the flow guard timers expire.

This parameter will appear in the ACLI as follows:

```
red-flow-port                   1985
```

Media State Checkpointing Configuration

To configure media state checkpointing across an HA node in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **media-manager-config** and press Enter.

   ```
   ORACLE(media-manager)# media-manager-config
   ```

4. **red-flow-port**—Enter the port number for checkpointing media flows associated with the HA interface. This is the port where media flow checkpoint message are sent and received.

   Setting this field to **0** disables media state checkpointing. The default value is **1985**. The valid range is:

   - Minimum—0, 1025
   - Maximum—65535

Limiting the Rate of Gratuitous ARP (GARP)

You can configure the Oracle Communications Session Border Controller (SBC) to minimize the rate of IPv4 GARP and IPv6 Neighbor Discovery (ND) traffic it sends out during a failover. This can prevent address resolution errors, caused by slow switching infrastructure. If you do not configure this rate limiting the system sends these messages as fast as possible, based on system load.

When the SBC executes an HA switchover, the new active SBC sends out GARP and/or ND messages for every configured IP address/VLAN pair. To limit the volume of these messages, configure the message window and the number of messages the system sends in this window using **redundancy** options. No reboot is required. The number of messages is the aggregate of GARPs and NDs. When configured, the system sends the configured number of messages, pauses for the remaining time (configured time minus elapsed time), and repeats the process until all messages are sent.

The **garps-per-interval** option, specifies the number of messages, from 10 to 1000. If the configured rate exceeds the rate at which the SBC can transmit these messages, the SBC transmits these messages at its maximum rate, without pacing.

The **garp-interval** option, specifies the message window, from 5 ms 100 milliseconds.

The ACLI example below configures this function to send 500 messages every 50 milliseconds.

```
ORACLE(redundancy-config)# options +garps-per-interval 500
ORACLE(redundancy-config)# options +garp-interval 50
```

Setting the **garps-per-interval** to 0, the default, disables this function.

Additional configuration considerations include:

- If you set either parameters to a value smaller than the allowed value, the system sets it to the minimum value.
- If you set either parameters to a value greater than the allowed value, the system sets it to the maximum value.
- If you set the **garps-per-interval** to a nonzero value and do not set the **garp-interval**, the system uses the minimum value (5ms) for the **garp-interval**.

Additional operational considerations include:

- The SBC does not re-transmit messages for any interface until it has sent the initial message for every interface.
- If configured to send more messages than it can, the SBC ignores the configuration and sends as many as it can.

HA Media Interface Keepalive

In an HA node, it is possible for the two systems in the node to lose communication via the management (rear, wancom) interfaces. For example, wancom 1 and wancom 2 might become disconnected, and cause the heartbeat synchronization to fail. This type of failure causes communication errors because both systems try to assume the active role and thereby access resources reserved for the active system.

To avoid these types of conditions, you can enable an option instructing the standby system to take additional time before going to the active state. This check occurs through the system's media interfaces. Using it, the standby can determine whether or not there has been a true active failure.

> **Note:**
>
> This media interface keepalive configuration is invalid for cloud deployments.

In cases when the standby determines the active system has not truly failed, it will go out of service because it will have determined it no longer has up-to-date data from its active counterpart. You can restore functionality by re-establishing management (rear) interface communication between the system in the node, and then re-synchronizes the standby by rebooting it.

When you enable the media interface keepalive, the standby system in the HA node sends ARP requests to determine if the media interfaces' virtual IP address are active. There are two possible outcomes:

- If it receives responses to its ARP requests, the standby takes itself out of service—to prevent a conflict with the active.
- If it does not receive responses to its ARP requests within a timeout value you set, then standby assumes the active role in the HA node.

Impact to Boot-Up Behavior

With the HA media interface keepalive enabled, the Oracle Communications Session Border Controller might be in the initial state longer than if the feature were disabled because it requires more information about the media (front) interfaces.

HA Media Interface Keepalive Configuration

You turn the HA media interface keepalive on by setting a timeout value for the standby to receive responses to its ARP requests before it assumes the active role in the HA node. Keeping this parameter set to 0, its default, disables the keepalive

To enable the HA media interface keepalive:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **redundancy** and press Enter.

   ```
   ORACLE(session-router)# redundancy
   ORACLE(redundancy)#
   ```

If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **media-if-peercheck-time**—Enter the amount of time in milliseconds for the standby system in an HA node to receive responses to its ARP requests via the media interface before it takes over the active role from its counterpart.

   The default is 0, which turns the HA media interface keepalive off. The maximum value is 500 milliseconds.

5. Save and activate your configuration.

Critical Memory Switchover

You can configure a high availability deployment of the SBC to switch to the standby when the system detects memory utilization that is persistently high. Over-utilization of memory can trigger a system crash. This function reduces the risk of those crashes.

The SBC includes an alarm mechanism that raises alarms and sends traps when memory utilization has exceeded your configured critical threshold. This function uses and extends upon that feature to trigger an HA switchover when memory utilization exceeds your configured critical threshold for 2 consecutive alarm iterations. Each utilization alarm consumes 15 seconds, resulting in a switchover when the condition exists for a maximum of 30 seconds.

The SBC uses the existing health score function to trigger this switchover. When the system experiences two consecutive memory alarm conditions, it subtracts 100 from the health score of the active machine. This value would exceed any HA trigger setting and cause the system to switchover to the standby. The SBC also uses the existing alarm/trap implemented for other alarm threshold functions.

You set the **critical-memory-abort** option in the **system-config** to **switchover** to configure this switchover function.

```
ORACLE(system-config)# options +critical-memory-abort=switchover
```

Valid values for this option include **enabled**, **disabled**, and **switchover**.

If you type options and then the option value without the plus sign, you overwrite any previously configured options. To add a new option to an options list, prepend the new option with a plus sign as shown above.

This function requires that you configure the critical memory **alarm-threshold** so that the system has a value to use. The example below sets the critical memory alarm threshold to 80 percent.

```
ORACLE(system-config)# alarm-threshold
alarm-threshold
 type        memory
 volume
 severity    critical
 value       80
```

When the SBC first reaches this critical value, the system:

- Raises an ACME Alarm
- Issues an SNMP trap

The alarm and trap remain valid as long as the system remains in a critical state. If the SBC reaches this critical value twice within 30 seconds, the system decreases its health score by 100, triggering a switchover to the new Active,

At this point, the original Active is not available for any ensuing switchover. But after this switchover, you can perform a reboot to the SBC on the former active device. This reboot clears memory and resets the health score, making the new standby available for any ensuing switchover.

The system does not automatically clear the alarm or issue a clear trap for this feature. You use the **clear-alarm** command manually or reboot the system to clear this alarm.

> **Note:**
>
> If you configure this option on a system that is not operating in an HA deployment, the SBC stops processing calls after it reaches the critical memory threshold.

Switchover Behavior During Traffic Spikes

The SBC includes an additional check function to ensure this switchover behavior does not take place if excessive memory utilization is triggered by temporary traffic spikes. To accomplish this, the system stores the exact memory usage (excluding the free list) whenever it hits the critical memory threshold the first time. The system then compares this memory in subsequent iterations to determine whether the system is remaining over the critical memory threshold. The system triggers the switchover in the following cases to ensure a temporary traffic spike did not cause that first detection:

1. The stored memory value in 1st iteration is less than the subsequent iteration value.
2. The subsequent iteration value is greater than the critical threshold configured value.

Related Configuration

This feature interacts with the **memory-utilization-threshold** and **heap-threshold** memory options in the **system-config**.

- The system allows you to configure the values of the **alarm-threshold** of severity critical in conjunction with the **heap-threshold** option. This feature, however always refers to the critical **alarm-threshold** to trigger its functionality. The system uses **heap-threshold** memory options in the **system-config** to determine whether it can accept new SIP requests.
- On the SLB, the system considers the minimum **heap-threshold** and **memory-utilization-threshold** values to determine when to stop throttling traffic. The lower of these values becomes the value for both.

RTC Notes

Starting in Release 4.1, the HA configuration is supported for real-time configuration (RTC). However, not all of the HA-related parameters are covered by RTC because of the impact on service it would cause to reconfigure these parameters dynamically.

This section sets out what parameters you should not dynamically reconfigure, or should dynamically reconfigure with care.

HA

Changes to the following ACLI parameters will have the noted consequences when dynamically reconfigured:

- **cfg-max-trans**—Changing this value could cause the activation time to lengthen slightly
- **init-time**, **becoming-standby-time**, and **becoming-active-time**—Changes take place only if the system is not transitioning between these states; otherwise the system waits until the transition is complete to make changes
- **percent-drift** and **advertisement-time**—Changes are communicated between nodes in the HA pair as part of regular health advertisements

In addition, the following parameters are not part of the RTC enhancement, for the reason specified in the right-hand column.

Parameter	Impact
state	Disrupts service
port	Disrupts service; leaves systems in an HA node without a means of communicating with each other
cfg-port	Disrupts service; leaves systems in an HA node without a means of communicating with each other
cfg-max-trans	Disrupts service
cfg-sync-start-time	Disrupts configuration replication
cfg-sync-comp-time	Disrupts configuration replication

Protocol-Specific Parameters and RTC

In addition, you should not change any of the parameters related to HA that are part of protocol or media management configurations that are used for protocol/media checkpointing. These are:

- SIP configuration
  - **red-max-trans**
  - **red-sync-start-time**
  - **red-sync-comp-time**
- Media Manager configuration
  - **red-flow-port**
  - **red-max-trans**
  - **red-sync-start-time**
  - **red-sync-comp-time**

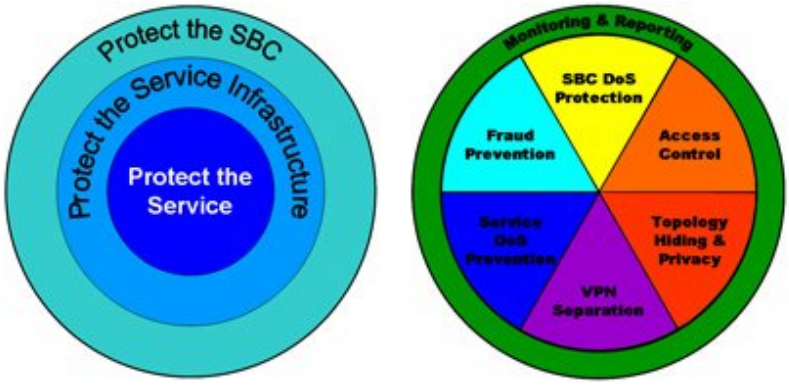
14
Security

This chapter explains Oracle Communications Session Border Controller security, which is designed to provide security for administrative security, VoIP and other multimedia services. It includes Admin Security, access control, DoS attack, and overload protection, which help secure service and protect the network infrastructure (including the Oracle Communications Session Border Controller). In addition, Oracle Communications Session Border Controller security lets legitimate users still place calls during attack conditions; protecting the service itself.

Security Overview

Oracle Communications Session Border Controller security includes the Net-SAFE framework's numerous features and architecture designs. Net-SAFE is a requirements framework for the components required to provide protection for the Session Border Controller (SBC), the service provider's infrastructure equipment (proxies, gateways, call agents, application servers, and so on), and the service itself.

The following diagrams illustrate Net-SAFE:



Each of Net-SAFE's seven functions consists of a collection of more specific features:

- Session border controller DoS protection: autonomic, SBC self-protection against malicious and non-malicious DoS attacks and overloads at Layers 2 to 4 (TCP, SYN, ICMP, fragments, and so on) and Layers 5 to 7 (SIP signaling floods, malformed messages, and so on).
- Access control: session-aware access control for signaling and media using static and dynamic permit/deny access control lists (ACLs) at layer 3 and 5.
- Topology hiding and privacy: complete infrastructure topology hiding at all protocol layers for confidentiality and attack prevention security. Also, modification, removal or insertion of call signaling application headers and fields. Includes support for the SIP Privacy RFC.
- VPN separation: support for Virtual Private Networks (VPNs) with full inter-VPN topology hiding and separation, ability to create separate signaling and media-only VPNs, and with optional intra-VPN media hair-pinning to monitor calls within a VPN.
- Service infrastructure DoS prevention: per-device signaling and media overload control, with deep packet inspection and call rate control to prevent DoS attacks from reaching

service infrastructure such as SIP servers, softswitches, application servers, media servers or media gateways.

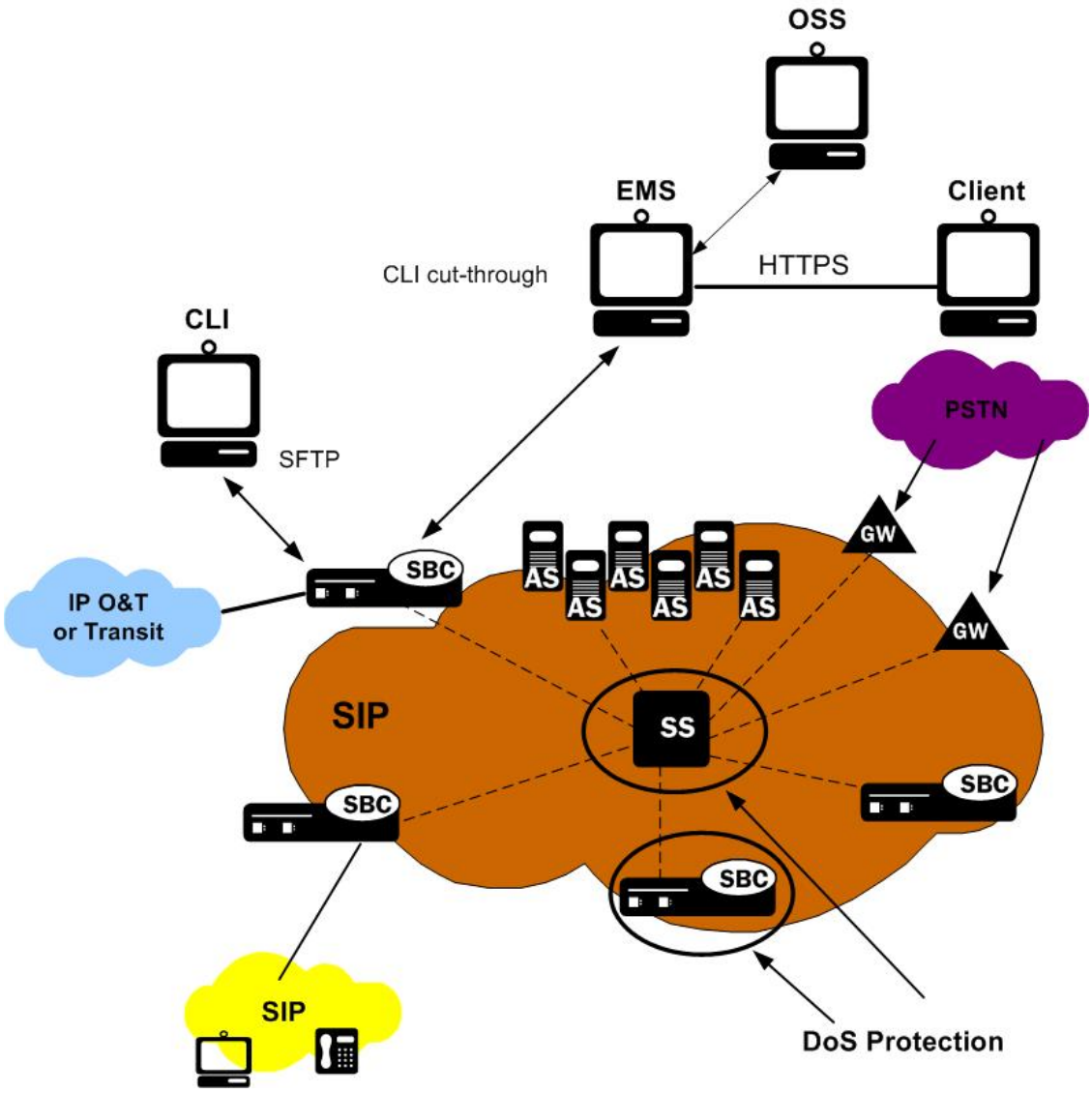
- Fraud prevention: session-based authentication, authorization, and contract enforcement for signaling and media; and service theft protection.
- Monitoring and reporting: audit trails, event logs, access violation logs and traps, management access command recording, Call Detail Records (CDRs) with media performance monitoring, raw packet capture ability and lawful intercept capability. The monitoring method itself is also secured, through the use of SSH and SFTP, and through the ability to use a separate physical Ethernet port for management access.

Denial of Service Protection

This section explains the Denial of Service (DoS) protection for the Oracle Communications Session Border Controller. The Oracle Communications Session Border Controller DoS protection functionality protects softswitches and gateways with overload protection, dynamic and static access control, and trusted device classification and separation at Layers 3-5. The Oracle Communications Session Border Controller itself is protected from signaling and media overload, but more importantly the feature allows legitimate, trusted devices to continue receiving service even during an attack. DoS protection prevents the Oracle Communications Session Border Controller host processor from being overwhelmed by a targeted DoS attack from the following:

- IP packets from an untrusted source as defined by provisioned or dynamic ACLs
- IP packets for unsupported or disabled protocols
- Nonconforming/malformed (garbage) packets to signaling ports
- Volume-based attack (flood) of valid or invalid call requests, signaling messages, and so on.
- Overload of valid or invalid call requests from legitimate, trusted sources

The following diagram illustrates DoS protection applied to the softswitch and to the Oracle Communications Session Border Controller.



Levels of DoS Protection

The multi-level Oracle Communications Session Border Controller DoS protection consists of the following strategies:

- Fast path filtering/access control: access control for signaling packets destined for the Oracle Communications Session Border Controller host processor as well as media (RTP) packets. The Oracle Communications Session Border Controller performs media filtering by using the existing dynamic pinhole firewall capabilities. Fast path filtering packets destined for the host processor require the configuration and management of a trusted list and a deny list for each Oracle Communications Session Border Controller realm (although the actual devices can be dynamically trusted or denied by the Oracle Communications Session Border Controller based on configuration). You do not have to provision every endpoint/device on the Oracle Communications Session Border Controller, but instead retain the default values.
- Host path protection: includes flow classification, host path policing and unique signaling flow policing. Fast path filtering alone cannot protect the Oracle Communications Session Border Controller host processor from being overwhelmed by a malicious attack from a

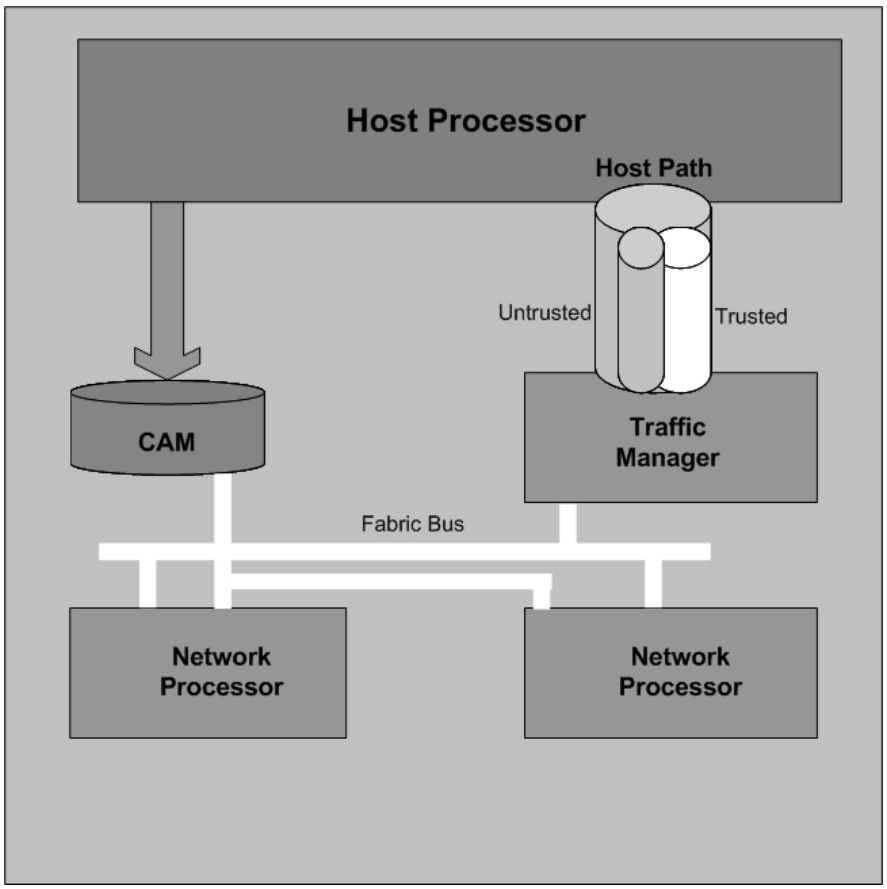
trusted source. The host path and individual signaling flows must be policed to ensure that a volume-based attack will not overwhelm the Oracle Communications Session Border Controller's normal call processing; and subsequently not overwhelm systems beyond it. The Oracle Communications Session Border Controller must classify each source based on its ability to pass certain criteria that is signaling- and application-dependent. At first each source is considered untrusted with the possibility of being promoted to fully trusted. The Oracle Communications Session Border Controller maintains two host paths, one for each class of traffic (trusted and untrusted), with different policing characteristics to ensure that fully trusted traffic always gets precedence.

- Host-based malicious source detection and isolation – dynamic deny list. Malicious sources can be automatically detected in real-time and denied in the fast path to block them from reaching the host processor.

About the Process

DoS attacks are handled in the Oracle Communications Session Border Controller's host path. The Oracle Communications Session Border Controller uses NAT table entries to filter out undesirable IP addresses; creating a deny list. After a packet from an endpoint is accepted through NAT filtering, policing is implemented in the Traffic Manager subsystem based on the sender's IP address. NAT table entries distinguish signaling packets coming in from different sources for policing purposes. The maximum number of policed calls that the Oracle Communications Session Border Controller can support is 16K (on 32K CAM / IDT CAM).

The Traffic Manager has two pipes, trusted and untrusted, for the signaling path. Each signaling packet destined for the host CPU traverses one of these two pipes.



Trusted Path

Packets from trusted devices travel through the trusted pipe in their own individual queues. In the Trusted path, each trusted device flow has its own individual queue (or pipe). The Oracle Communications Session Border Controller can dynamically add device flows to the trusted list by promoting them from the Untrusted path based on behavior; or they can be statically provisioned.

Trusted traffic is put into its own queue and defined as a device flow based on the following:

- source IP address
- source UDP/TCP port number
- destination IP address
- destination UDP/TCP port (SIP interface to which it is sending)
- realm it belongs to, which inherits the Ethernet interface and VLAN it came in on

For example, SIP packets coming from 10.1.2.3 with UDP port 1234 to the Oracle Communications Session Border Controller SIP interface address 11.9.8.7 port 5060, on VLAN 3 of Ethernet interface 0:1, are in a separate Trusted queue and policed independently from SIP packets coming from 10.1.2.3 with UDP port 3456 to the same Oracle Communications Session Border Controller address, port and interface.

Data in this flow is policed according to the configured parameters for the specific device flow, if statically provisioned. Alternatively, the realm to which endpoints belong have a default policing value that every device flow will use. The defaults configured in the realm mean each device flow gets its own queue using the policing values. As shown in the previous example, if both device flows are from the same realm and the realm is configured to have an average rate limit of 10K bytes per second (10KBps), each device flow will have its own 10KBps queue. They are not aggregated into a 10KBps queue.

The individual flow queues and policing lets the Oracle Communications Session Border Controller provide each trusted device its own share of the signaling, separate the device's traffic from other trusted and untrusted traffic, and police its traffic so that it can't attack or overload the Oracle Communications Session Border Controller (therefore it is trusted, but not completely).

Address Resolution Protocol Flow

The Address Resolution Protocol (ARP) packets are given their own trusted flow with the bandwidth limitation of 8 Kbps. ARP packets are able to flow smoothly, even when a DoS attack is occurring.

Untrusted Path

Packets (fragmented and unfragmented) that are not part of the trusted or denied list travel through the untrusted pipe. In the untrusted path, traffic from each user/device goes into one of 2048 queues with other untrusted traffic. Packets from a single device flow always use the same queue of the 2048 untrusted queues, and 1/2048th of the untrusted population also uses that same queue. To prevent one untrusted endpoint from using all the pipe's bandwidth, the 2048 flows defined within the path are scheduled in a fair-access method. As soon as the Oracle Communications Session Border Controller decides the device flow is legitimate, it will promote it to its own trusted queue.

All 2048 untrusted queues have dynamic sizing ability, which allows one untrusted queue to grow in size, as long as other untrusted queues are not being used proportionally as much. This dynamic queue sizing allows one queue to use more than average when it is available. For example, in the case where one device flow represents a PBX or some other larger volume device. If the overall amount of untrusted packets grows too large, the queue sizes rebalance, so that a flood attack or DoS attack does not create excessive delay for other untrusted devices.

In the usual attack situations, the signaling processor detects the attack and dynamically demotes the device to denied in the hardware by adding it to the deny ACL list. Even if the Oracle Communications Session Border Controller does not detect an attack, the untrusted path gets serviced by the signaling processor in a fair access mechanism. An attack by an untrusted device will only impact 1/1000th of the overall population of untrusted devices, in the worst case. Even then there's a probability of users in the same 1/1000th percentile getting in and getting promoted to trusted.

IP Fragment Packet Flow

All fragment packets are sent through their own 1024 untrusted flows in the Traffic Manager. The first ten bits (LSB) of the source address are used to determine which fragment-flow the packet belongs to. These 1024 fragment flows share untrusted bandwidth with already existing untrusted-flows. In total, there are 2049 untrusted flows: 1024-non-fragment flows, 1024 fragment flows, and 1 control flow.

Fragmented ICMP packets are qualified as ICMP packets rather than fragment packets. Fragment and non-fragmented ICMP packets follow the trusted-ICMP-flow in the Traffic Manager, with a bandwidth limit of 8Kbs.

Fragment Packet Loss Prevention

You can set the maximum amount of bandwidth (in the **max-untrusted-signaling** parameter) you want to use for untrusted packets. However, because untrusted and fragment packets share the same amount of bandwidth for policing, any flood of untrusted packets can cause the Oracle Communications Session Border Controller to drop fragment packets.

To prevent fragment packet loss on the Acme Packet 3820 and Acme Packet 4500, you can set the **fragment-msg-bandwidth**. When you set any value other than 0 (which disables it), the Oracle Communications Session Border Controller:

- Provides for a separate policing queue for fragment packets (separate from that used for untrusted packets)
- Uses this new queue to prevent fragment packet loss when there is a flood from untrusted endpoints.

When you set up a queue for fragment packets, untrusted packets likewise have their own queue—meaning also that the **max-untrusted-signaling** and **min-untrusted-signaling** values are applied to the untrusted queue.

Static and Dynamic ACL Entry Limits

When deployed over its own hardware, the Oracle Communications Session Border Controller can simultaneously police a maximum of 250,000 trusted device flows, while at the same time denying an additional 32,000 attackers. If list space becomes full and additional device flows need to be added, the oldest entries in the list are removed and the new device flows are added.

Static Trusted and Untrusted ACL Limits for virtual platforms

When deployed on virtual platforms, the SBC supports static ACL entry counts based on virtual machine memory. Deployments under 8G of memory support 8K trusted and 4K untrusted entries. When memory is:

- Between 8G and 64G, supported entries include:
  - Trusted static ACLs is 1024 per Gig
  - Untrusted static ACLs is 512 per Gig
- Greater than 64G, supported entries include:
  - Trusted static ACLs is 65536
  - Untrusted static ACLs is 32768

> **Note:**
>
> Dynamic ACL entries are independent of this support.

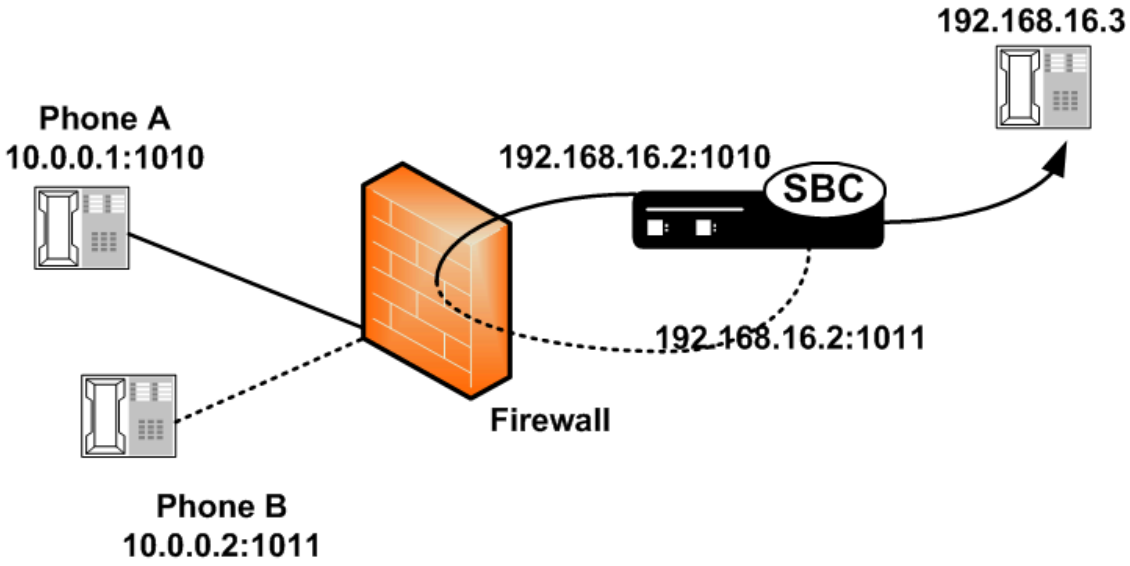
System capacities vary across the range of platforms that support the SBC. To query the current system capacities for the platform you are using, execute the **show platform limits** command.

Dynamic Deny for HNT

Dynamic deny for HNT has been implemented on the Oracle Communications Session Border Controller for cases when callers are behind a NAT or firewall. Without this feature, if one caller behind a NAT or firewall were denied, the Oracle Communications Session Border Controller would also deny all other users behind the same NAT or firewall. This would be true even for endpoints behind the firewall that had not crossed threshold limits you set for their realm; all endpoints behind the firewall would go out of service. In the following diagram, both Phone A and Phone B would be denied because their IP addresses would be translated by the firewall to the same IPv4 address (192.168.16.2).

However, dynamic deny for HNT allows the Oracle Communications Session Border Controller to determine, based on the UDP/TCP port, which endpoints should be denied and which should be allowed. The Oracle Communications Session Border Controller can determine that even though multiple endpoints originating behind a firewall appear with the same IPv4 address, those addresses use different ports and are unique.

As shown in the diagram below, the ports from Phone A and Phone B remain unchanged. This way, if Phone A violates the thresholds you have configured, the Oracle Communications Session Border Controller can block traffic from Phone A while still accepting traffic from Phone B.



Host and Media Path Protection Process

The Oracle Communications Session Border Controller Network Processors (NPs) check the deny and permit lists for received packets, and classify them as trusted, untrusted or denied (discard). Only packets to signaling ports and dynamically signaled media ports are permitted. All other packets sent to Oracle Communications Session Border Controller ports are filtered. Only packets from trusted and untrusted (unknown) sources are permitted; any packet from a denied source is dropped by the NP hardware. The Traffic Manager manages bandwidth policing for trusted and untrusted traffic, as described earlier. Malicious traffic is detected in the host processor and the offending device is dynamically added to denied list, which enables early discard by the NP. Devices become trusted based on behavior detected by the Signaling Processor, and dynamically added to the trusted list. This process enables the proper classification by the NP hardware. All other traffic is untrusted (unknown).

SBC Access Control

You an create static trusted/untrusted/deny lists with source IP addresses or IP address prefixes, UDP/TDP port number or ranges, and based on the appropriate signaling protocols. Furthermore, the Oracle Communications Session Border Controller can dynamically promote and demote device flows based on the behavior, and thus dynamically creates trusted, untrusted, and denied list entries.

Access Control for Hosts

ACLs are supported for all VoIP signaling protocols on the Oracle Communications Session Border Controller: SIP and H.323. The Oracle Communications Session Border Controller loads ACLs so they are applied when signaling ports are loaded. The following rules apply to static NAT entries based on your configuration:

- If there are no ACLs applied to a realm that have the same configured trust level as that realm, the Oracle Communications Session Border Controller adds a default NAT entry using the realm parameters.
- If you configure a realm with none as its trust level and you have configured ACLs, the Oracle Communications Session Border Controller only applies the ACLs.

- If you set a trust level for the ACL that is lower than the one you set for the realm, the Oracle Communications Session Border Controller will not add a separate NAT entry for the ACL.

ACLs provide access control based on destination addresses when you configure destination addresses as a way to filter traffic. You can set up a list of access control exceptions based on the source or the destination of the traffic.

For dynamic ACLs based on the promotion and demotion of endpoints, the rules of the matching ACL are applied.

Access Control Endpoint Classification Capacity and DoS

To view endpoint classification capacity limits for your current platform, use the **show platform limits** command. The output is dependent on the combination of hardware and software you are running.

Media Access Control

The media access control consists of media path protection and pinholes through the firewall. Only RTP and RTCP packets from ports dynamically negotiated through signaling (SIP and H.323) are allowed, which reduces the chance of RTP hijacking. Media access depends on both the destination and source RTP/RTCP UDP port numbers being correct, for both sides of the call.

Host Path Traffic Management

The host path traffic management consists of the dual host paths discussed earlier:

- Trusted path is for traffic classified by the system as trusted. You can initially define trusted traffic by ACLs, as well as by dynamically promoting it through successful SIP registration, or a successful call establishment. You can configure specific policing parameters per ACL, as well as define default policing values for dynamically-classified flows. Traffic for each trusted device flow is limited from exceeding the configured values in hardware. Even an attack from a trusted, or spoofed trusted, device cannot impact the system.
- Untrusted path is the default for all unknown traffic that has not been statically provisioned otherwise. For example, traffic from unregistered endpoints. Pre-configured bandwidth policing for all hosts in the untrusted path occurs on a per-queue and aggregate basis.

Traffic Promotion

Traffic is promoted from untrusted to trusted list when the following occurs:

- successful SIP registration for SIP endpoints
- successful session establishment for SIP calls

Malicious Source Blocking

Malicious source blocking consists of monitoring the following metrics for each source:

- SIP transaction rate (messages per second)
- SIP call rate (call attempts per second)
- Nonconformance/invalid signaling packet rate

Device flows that exceed the configured invalid signaling threshold, or the configured valid signaling threshold, within the configured time period are demoted, either from trusted to untrusted, or from untrusted to denied classification.

Blocking Actions

Blocking actions include the following:

- Dynamic deny entry added, which can be viewed through the ACLI.
- SNMP trap generated, identifying the malicious source

Dynamically added deny entries expire and are promoted back to untrusted after a configured default deny period time. You can also manually clear a dynamically added entry from the denied list using the ACLI.

Protecting Against Session Agent Overloads

You can prevent session agent overloads with registrations by specifying the registrations per second that can be sent to a session agent.

ARP Flood Protection Enhancements

Enhancements have been made to the way the Oracle Communications Session Border Controller provides ARP flood protection. In releases prior to Release C5.0, there is one queue for both ARP requests and responses, which the Oracle Communications Session Border Controller polices at a non-configurable limit (eight kilobytes per second). This method of ARP protection can cause problems during an ARP flood, however. For instance, gateway heartbeats the Oracle Communications Session Border Controller uses to verify (via ARP) reachability for default and secondary gateways could be throttled; the Oracle Communications Session Border Controller would then deem the router or the path to it unreachable, decrement the system's health score accordingly. Another example is when local routers send ARP requests for the Oracle Communications Session Border Controller's address are throttled in the queue; the Oracle Communications Session Border Controller never receives the request and so never responds, risking service outage.

The solution implemented to resolve this issue is to divide the ARP queue in two, resulting in one ARP queue for requests and a second for responses. This way, the gateway heartbeat is protected because ARP responses can no longer be flooded from beyond the local subnet. In addition, the Oracle Communications Session Border Controllers in HA nodes generate gateway heartbeats using their shared virtual MAC address for the virtual interface.

In addition, this solution implements a configurable ARP queue policing rate so that you are not committed to the eight kilobytes per second used as the default in prior releases. The previous default is not sufficient for some subnets, and higher settings resolve the issue with local routers sending ARP request to the Oracle Communications Session Border Controller that never reach it or receive a response.

As a security measure, in order to mitigate the effect of the ARP table reaching its capacity, configuring the media-manager option, **active-arp**, is advised. Enabling this option causes all ARP entries to get refreshed every 20 minutes.

Dynamic Demotion for NAT Devices

In addition to the various ways the Oracle Communications Session Border Controller already allows you to promote and demote devices to protect itself and other network elements from DoS attacks, it can now block off an entire NAT device. The Oracle Communications Session

Border Controller can detect when a configurable number of devices behind a NAT have been blocked off, and then shut off the entire NAT's access.

This dynamic demotion of NAT devices can be enabled for an access control (ACL) configuration or for a realm configuration. When you enable the feature, the Oracle Communications Session Border Controller tracks the number of endpoints behind a single NAT that have been labeled untrusted. It shuts off the NAT's access when the number reaches the limit you set.

The demoted NAT device then remains on the untrusted list for the length of the time you set in the **deny-period**.

DoS Counter Notifications

The SBC provides ACL and DDOS statistics that track events for ARP, trusted, and untrusted traffic. These statistics include notifications about ARP watermarks and trusted and untrusted queue metrics to provide visibility into traffic management rates, based on traffic patterns in normal and peak times. You configure these thresholds as a percentage of the configured traffic rates within the media-manager configuration element. This provides you with early notification of traffic congestion so you can better tune the global media settings for DDOS. The SBC does not drop the packets affected through threshold events. Instead, it forwards them to a traffic manager for making permit/drop decisions prior to sending it to the host. In addition to host bound events, the SBC generates SMNP traps and alarms for TCAs that monitor ARP, trusted, untrusted and max-signaling rates. You can collect statistics on related traffic using the ACLI, SNMP walks, HDR and REST.

Threshold boundary configurations per counter type (ARP, untrusted and trusted) are configuration in the media-manager element. You perform this configuration within the **media-manager** element. These thresholds allow you to establish minor, major and critical levels upon which you receive notifications.

For example, a SBC, when configured with a **max-signaling** committed rate of 1MB/sec, and a trusted rate of 90%, allows up to 3 threshold levels from 8K – 800K (10% below the trusted drop rate of 900K in this example). You may configure watermark levels of, for example, 50% and 75% to notify you that the media manager settings for this peak rate of traffic may be insufficient.

You can retrieve these configurations using the **show dos threshold** command, which includes further command arguments to see more specific information. You can also configure the SBC to capture event counters using SNMP, HDR, and REST.

Statistics Output on the ACLI

You use the **show dos threshold counters** command to display DOS and ACL statistics, immediate threshold level alarms and the number of times each state changes. The Cleared column presents the number of times the DOS threshold dropped below its water mark since the last time the DOS timer fired.

```
ORACLE# show dos threshold counters
Traffic              Current Threshold level
Trusted traffic      No threshold crossed
Untrusted traffic    No threshold crossed
ARP traffic          Major threshold
Counters             ---------Lifetime ---------
                     Overloaded          Cleared
     ARP minor            574                0
     ARP major            574                0
```

```
ARP critical          325              60
untrusted minor       30                3
untrusted major       30                3
untrusted critical    24                2
trusted minor         28                3
trusted major         28                3
trusted critical      27                3
```

Additional arguments to this command includes **show dos threshold reset**, which resets the dos threshold counters.

SNMP

The SBC provides the same data via SNMP using the DoS threshold counter objects within the ap-apps.mib. Similar to the CLI stats, these objects provide counters for the number of times traffic crosses each threshold.

See the SBC *MIB Reference Guide* for further detail about SNMP function, configuration, and DoS threshold OID detail.

HDR

You can also use the HDR group, **dos-threshold-counters**, to retrieve the same counter statistics available using the ACLI and SNMP. These objects provide counters for the number of times traffic crosses each threshold.

See the SBC *HDR Resource Guide* for further detail about HDR function, configuration, and DoS traffic objects.

REST API Interface

The SBC includes a KPI type called dosThresholdCounters that includes further objects you use to get the number of times the statistic crosses the applicable threshold. These counters are available from the Statistics' REST endpoints.

- trustedMinorCounter—Counter incremented when trusted bandwidth crossed the minor threshold percentage
- trustedMajorCounter—Counter incremented when trusted bandwidth crossed the major threshold percentage
- trustedCriticalCounter—Counter incremented when trusted bandwidth crossed the critical threshold percentage
- untrustedMinorCounter—Counter incremented, when untrusted bandwidth crossed the minor threshold percentage
- untrustedMajorCounter—Counter incremented, when untrusted bandwidth crossed the major threshold percentage
- untrustedCriticalCounter—Counter incremented, when untrusted bandwidth crossed the critical threshold percentage
- arpMinorCounter—Counter incremented, when arp bandwidth crossed the minor threshold percentage
- arpMajorCounter—Counter incremented, when arp bandwidth crossed the major threshold percentage
- arpCriticalCounter—Counter incremented, when arp bandwidth crossed the critical threshold percentage

See the SBC *REST API Documentation* for further detail about these DoS threshold counters.

DDoS Alarms and SNMP Traps

Alarms on DoS threshold traffic are tightly aligned with SNMP traps. The SBC issues both simultaneously when the applicable traffic, trusted, untrusted or arp, exceed your configured thresholds. They both also clear as soon as the traffic falls below that threshold.

The SBC sends two SNMP traps that alert you when traffic crosses each threshold, and clear when the traffic falls back below the threshold:

- apDosThresholdCrossTrap
- apDosThresholdClearTrap

These traps break out into additional SNMP objects, one to specify the traffic type and one to specify the threshold cross status. You can see traffic type from the apDosThresholdTraffic object; a second object identifies cross status. The values that specify the detail include:

- apDosThresholdTraffic—Three types of possible values. (1 = trusted, 2 = untrusted , 3 = ARP)
- apDosThresholdMinorCrossed—Two types of possible values (0 = threshold not crossed, 1 = threshold crossed)
- apDosThresholdMajorCrossed—Two types of possible values (0 = threshold not crossed, 1 = threshold crossed)
- apDosThresholdCriticalCrossed—Two types of possible values (0 = threshold not crossed, 1 = threshold crossed)

The example below indicates that ARP traffic has exceeded your minor threshold:

- apDosThresholdTraffic—Set to 3
- apDosThresholdMinorCrossed—Set to 1

Similarly, the SBC issues alarms to present this same detail to you. These alarms align with the systems traffic queues, and include include:

- Trusted traffic crosses DOS threshold
- Untrusted traffic crosses DOS threshold
- ARP traffic crosses DOS threshold

Unlike SNMP, these present type and 'threshold crossed' in a single alarm object.

DoS Protection at the Session Level

You can configure the SBC to implement DoS protection when any individual session appears to be conducting an attack. You can configure this protection on a **realm-config** or a **session-agent**, with the **session-agent** configuration taking precedence when applicable.

Without this feature, the SBC does not prevent cases where traffic overload or malicious attacks are generated by a trusted source IP within the context of an active session. The system simply forwards as much of this traffic as the system's hardware and software capabilities can support.

Furthermore, the system normally does not demote any source IP when you configure the **access-control-trust-level** parameter in the applicable **realm-config** to **high**. Instead, the system allocates all of these packets to the trusted queue for processing.

This feature extends upon the system's existing DoS Prevention functionality by providing DoS attack detection and mitigation at the session level for an endpoint, thereby providing:

- Session-based protection against traffic overload and malicious attacks sourced from trusted access or core devices in realms where you have configured the **access-control-trust-level** to **high**.
- The implementation for preventing DoS attack especially from trusted parties on the core side of the system.
  The system triggers this DOS attack protection based on the rate of request/responses received per session and takes appropriate action, using your DOS configuration.

  With this feature enabled, the system can take action on the existing session only, rather than blocking the entire endpoint for further calls.

You enable this feature using the **dos-action-at-session** parameter, which allows you to specify the desired behavior, including:

- Terminate sessions that are generating a detected DoS attack
- Temporarily demote the endstation that is generating a detected DoS attack to untrusted and terminate that session. The system uses the untrusted queue for any further traffic from that system and promotes the system back to trusted after the standard demotion processes expire.
- Demote the endstation that is generating a detected DoS attack to untrusted and terminate that session. If the system detects a subsequent DoS attack before promoting that endstation back to trusted, the system further demotes that endstation to the denied state. The system waits for the deny timer to expire, then promotes the endstation to untrusted, then promotes the system back to trusted after the standard demotion processes expire.

> **Note:**
>
> The system does not perform this feature's functionality if the attacking endpoint is behind a NAT.

Processing Session Based DoS Attacks

When the SBC receives an initial request, it creates a new session. The SBC processes session based DoS attack mitigation for all endstations with a trust level of high using the steps below:

1. Identifies the associated endstation (or session agent) and realm for that session to monitor the traffic level.
2. Creates an address variable for that endstation (or session agent) and sets the default state of that variable to permit that traffic through the trusted queue (MBCD_ACL_PERMIT).
3. Tracks the burst rate of incoming requests and responses for that session.
   You configure packet rate and monitoring window size for a session at either **session-agent** or **realm-config** using the **max-inbound-per-session-burst-rate** and **burst-rate-window-per-session** parameters. For each session, the system maintains separate counters for ingress and egress endpoints. For this feature, the system calculates the packet rate of the received messages for a configured time period.
4. Detects a DoS attack when a request or response rate in a session exceeds the configured **max-inbound-per-session-burst-rate** threshold value.
5. Performs DoS mitigation per your configuration.

When the DoS attack is detected from either the ingress or egress endpoint in a session, the SBC controls the DoS attack in that session based on call state:

- If the call is already established in a session when the DoS attack occurs, the SBC terminates the session by sending a BYE to both the UAC and UAS side.
  For forked calls, if the ingress UAC initiates a DoS attack at session, the SBC terminates the call by sending a 503 error response towards the UAC and a CANCEL towards all the forked UAS endpoints.
- If the call session is in connecting state and the SBC has received only 1xx responses when the DoS attack occurs, the SBC disconnects the session by sending a CANCEL request towards the terminating UE side and a “503 DDoS Request Cancelled" response towards the originating UE side.
  Although there are multiple egress dialogs or multiple egress endpoints involved in a forked call, if the system detects a DoS attack within one of these sessions, it terminates the entire call session.

  For forked calls, the SBC only demotes the egress endpoint that initiates the DoS attack.

The SBC makes further decisions to demote or deny the IP address based on the applicable **dos-action-at-session** parameter setting. Behavior is further refined based on whether you have configured the feature at the **session-agent** or the **realm-config**.

- If the value of **dos-action-at-session** is **permit**, the SBC can demote the endpoint from trusted to untrusted. If the DoS attack resumes, the system can then deny all traffic from the endpoint until the deny-period timer expires:
  1. If the endpoint makes the first DoS attack in a session, the system demotes this endpoint to the untrusted queue and starts the promotion timer, “UNTRUST_TMO timer”, which lasts for 180 seconds. The system sends all of that session's traffic through the untrusted queue.
  2. If the UNTRUST_TMO timer expires and there is no DoS attack in that time-period, then the system promotes that endpoint back to the trusted list.
     If the endpoint executes another session level DoS attack within the 180 second timer window, the system further demotes that endpoint from untrusted to the deny list, and starts your configured **deny-period** timer.
  3. The system drops all requests or responses from a denied endpoint.
  4. The system promotes the endpoint back to the untrusted list from denied list when the **deny-period** timer expires.
  5. The system promotes the endpoint back to the trusted from the untrusted list when the UNTRUST_TMO timer expires.
- If the value of **dos-action-at-session** is **no-deny**, the SBC can demote the endpoint, but cannot deny the endpoint.
  1. If the endpoint sends messages that cross the DoS thresholds within a session, the SBC demotes the endpoint from trusted to untrusted queue and terminates the current session.
     At this point, the SBC can accept new calls from that endpoint.

     The SBC handles these calls through the untrusted queue, even though the **access-control-trust-level** is high, until the UNTRUST_TMO timer expires at 180 seconds
  2. When the UNTRUST_TMO timer expires, the SBC moves the endpoint back to the trusted list.
- If the value of **dos-action-at-session** is **session-drop**, the SBC does not demote or deny the trusted endpoint. Instead, the SBC terminates the current DOS session and allows new calls from this trusted endpoint.

- The **dos-action-at-session** parameter has an additional value, **inherit**, when configured at a **session-agent**. You use this value to instruct the **session-agent** to inherit its behavior from the **max-inbound-per-session-burst-rate**, **burst-rate-window-per-session** and **dos-action-at-session** settings on the **realm-config**.
  If **dos-action-at-session** at the **realm-config** is **none**, there is no inheritance process.

When the SBC receives multiple INVITE requests from an endpoint within the same timeframe, it saves the current DoS state. This prevents the system from repeating the same DoS action for each session initiated within that timeframe.

If you have configured the feature to **inherit** or **no-deny** and a trusted endpoint sends multiple initial INVITEs requests that eventually lead to a DoS attack, the SBC moves the endpoint from trusted to untrusted during first call's DoS detection. For all the other INVITE sessions, the system takes no further action because the endpoint is already moved to the untrusted state. The system terminates these calls based on the current DoS state.

The table below presents actions the SBC takes when applying this feature to mitigate a DoS attack perpetrated within a normal call.

Request DOS attack	Response DOS attack	Action at originating UAC (ingress side)	Action at terminating UAS (egress side)
PRACK, UPDATE or any subsequent request DoS attack from originating UE	N/A	Send 503 response towards originating UAC	Send CANCEL towards terminating UAS
N/A	200 OK of PRACK, UPDATE or any subsequent request's response DOS attack from terminating UE	Send 503 response towards originating UAC	Send CANCEL towards terminating UAS
UPDATE or any subsequent request DoS attack from terminating UE	N/A	Send 503 response towards originating UAC	Send CANCEL towards terminating UAS.
N/A	200 OK of UPDATE or any subsequent request's response DoS attack from originating UE	Send 503 response towards originating UAC	Send CANCEL towards terminating UAS
N/A	18x response of INVITE DOS attack from terminating UE	Send 503 response towards originating UAC	Send CANCEL towards terminating UAS
N/A	200 OK response of INVITE DOS attack from terminating UE	Send BYE towards originating UAC	• First send the ACK. • Send BYE towards terminating UAS
Re-INVITE DoS attack from originating UE	N/A	Send BYE towards originating UAC	Send BYE towards terminating UAS
N/A	200 OK of re-INVITE DoS attack from term UE	Send BYE towards originating UAC	Send BYE towards terminating UAS
Re-INVITE DoS attack from terminating UE	N/A	Send BYE towards originating UAC	Send BYE towards terminating UAS
N/A	200 OK of re-invite DOS attack from originating UE	Send BYE towards originating UAC	SEND BYE towards terminating UAS

Reporting

This feature generates SNMP standard traps and updates the applicable statistic counters when it demotes an endstation from trusted to untrusted and from untrusted to deny. You must enable the **trap-on-demote-to-deny** and **trap-on-demote-to-untrusted** parameters in "**media-manager-config** to get the traps.

When the system generates these traps based on this feature, it uses the following reason strings:

- If **dos-action-at-session** is **permit**, the reason string is "Too many in session (permit) inbound messages".
- If **dos-action-at-session** is **no-deny**, the reason string is "Too many in session (nodeny) inbound messages".

Session Based DoS Protection and Static ACLs

Static ACLs, which can have a trust level of none (default), low, medium or high, interact with this feature based on their and their respective realm's trust levels:

- Static ACLs and their Realms trust level is low/medium—Static ACLs at these levels follow standard DoS operation regardless of your **dos-action-at-session** configuration. The system does not monitor these ACLs SigAddress at the session level.
- Static ACLs and their Realm's trust level is high—This configuration is not recommended. In these cases, you should delete these ACLs. After deletion, your **dos-action-at-session** configuration applies to the entire realm.
- Static ACLs trust level is high and their Realm's trust level is low/medium—Your **dos-action-at-session** configuration is not applicable.

The following table explains whether or not this DoS feature is applicable based on the trust levels or your static ACLs and their applicable realms:

Access-control trust level (Static ACL)	Realm-config trust level	DDoS feature at session (dos-action-at-session parameter value)	DoS detected on the ACL IP	DoS detected on the Other IP	Comments
high	low	OFF	NO	YES (Standard)	Set **dos-action-at-session** to **none**
low	high	ON	YES (Standard)	Yes (Session)	
high	low	OFF	NO	YES (Standard)	Referring to the row above, the trust-level of above ACL was modified from low to high, trust-level of realm was modified from high to low, **dos-action-at-session** was set to none and SBC was not rebooted

Access-control trust level (Static ACL)	Realm-config trust level	DDoS feature at session (dos-action-at-session parameter value)	DoS detected on the ACL IP	DoS detected on the Other IP	Comments
delete ACL	high	ON	YES (Session)	Yes (Session)	Referring to the two rows above, the ACL was deleted and SBC was not rebooted
high	high	ON	Configuration not recommended	Configuration not recommended	It is not recommended to set the trust-level of both ACL and realm as high together, because of a legacy issue resulting in dead calls
delete ACL	high	ON	YES (Session)	Yes (Session)	Referring to the row above, the ACL was deleted and SBC was not rebooted
low	high	ON	YES (Standard)	Yes (Session)	Referring to the two rows above, the trust-level of above ACL was modified from high to low and SBC was not rebooted
off (not configured)	low	OFF	YES (Standard)	YES (Standard)	Set **dos-action-at-session** to **none**
off (not configured)	high	ON	YES (Session)	YES (Session)	

Session Based DoS Mitigation Examples

This section presents specific examples of the SBC performing session-based DoS mitigation.

The examples below presents the SBC performing Session Based DoS mitigation at the inbound and the outbound side of a session.

DOS Attack at the Ingress

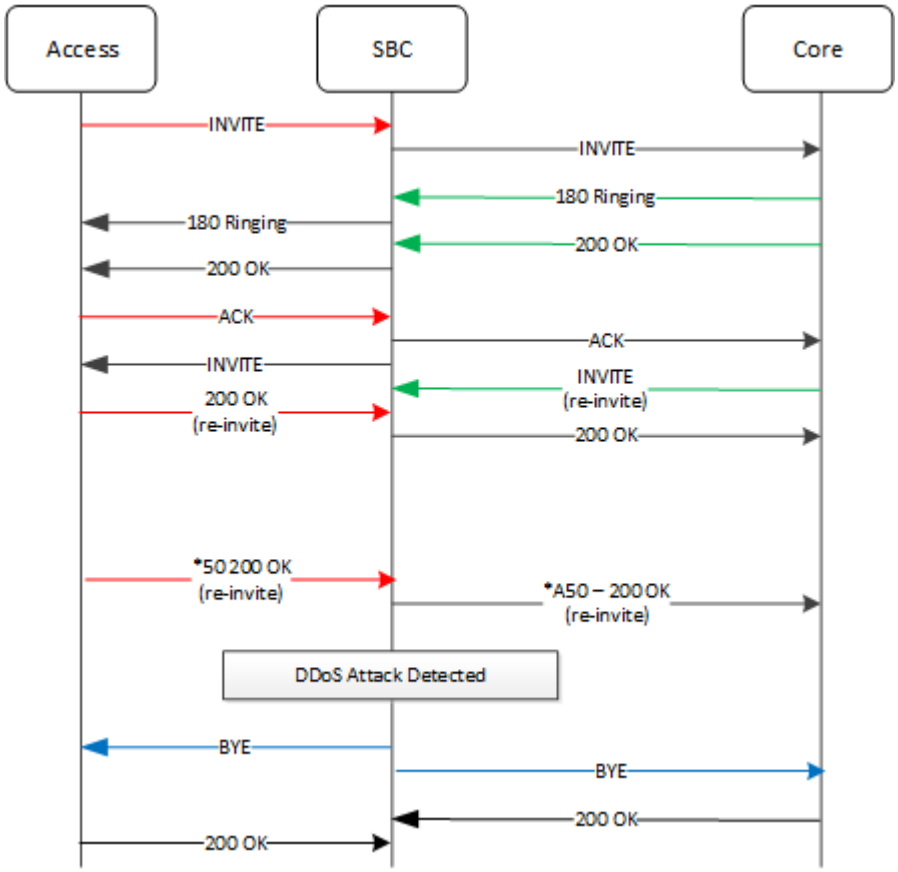
When an endpoint sends an initial INVITE request to the SBC, it creates a new session. For every request or response received at the ingress side in the particular session, the system increments the ingress packet count for that session.

The SBC increments the ingress count counter until the **burst-rate-window-per-session** timer configured expires.

If the ingress count in a session reaches the threshold value, which you configure at either the **realm-config** or **session-agent** using the **max-inbound-per-session-burst-rate** parameter, the SBC detects traffic exceeding that value as a DOS attack from trusted endpoint.

The system resets the ingress count to 0 after the **burst-rate-window-per-session** timer window times out. The **burst-rate-window-per-session** timer window runs in a loop and the system monitors the ingress packet counters in that time frame against the threshold value.

The image below shows the SBC performing this feature with the **max-inbound-per-session-burst-rate** set to 50.



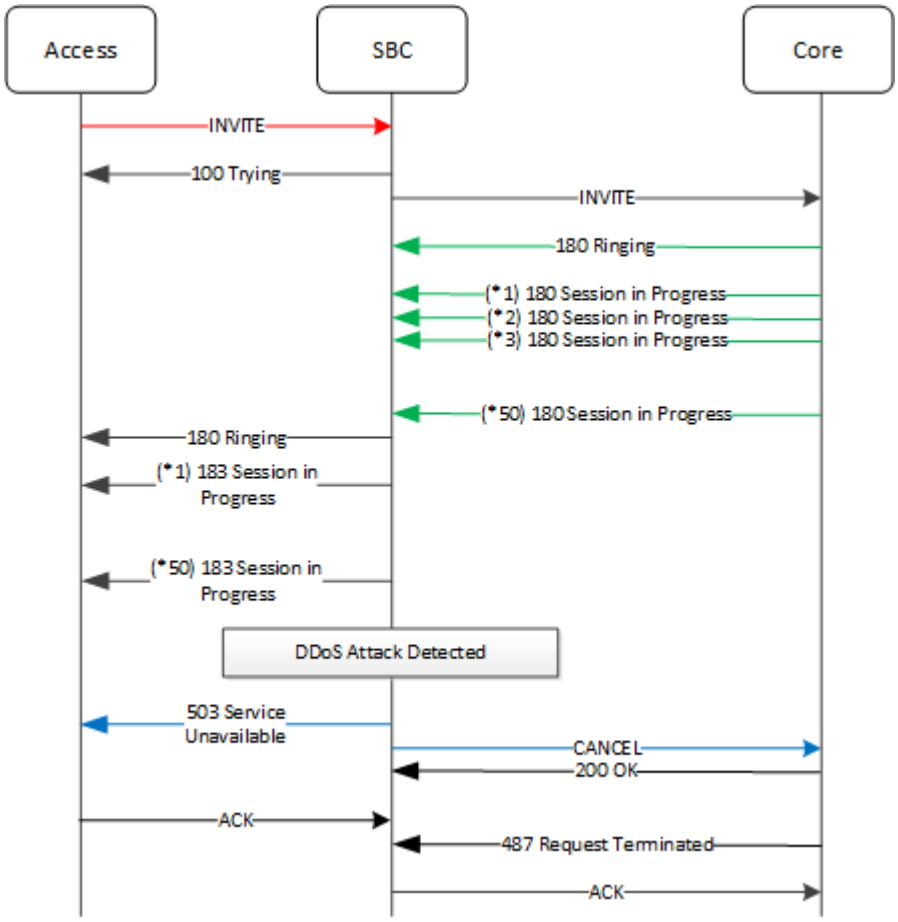
DOS Attack at the Egress

If the egress endpoint is trusted and is executing a DOS attack by pumping multiple requests/responses in a call, the SBC detects this attack at the session level.

The SBC associates any response to initial INVITE, or any subsequent request received at the egress side of a call with an existing session. It increments the egress count for that session for every request or response it receives at the egress side. The SBC increments this egress counter until the **burst-rate-window-per-session** timer expires. If this count reaches the maximum threshold value within that window, the SBC flags that call as generating a DOS attack.

The system resets the egress count to 0 after the **burst-rate-window-per-session** timer expires. For the duration of the session, the **burst-rate-window-per-session** timer window runs in a loop, and the system monitors the egress packet counters in that time frame against the threshold value.

The image below shows the SBC performing this feature with the **max-inbound-per-session-burst-rate** set to 50.



Session Based DOS Mitigation Configuration

You can configure this feature on an applicable **realm-config** or **session-agent**. The **dos-action-at-session** parameter is key to understanding and specifying operation of this feature.

Example syntax for the **dos-action-at-session** parameter on a realm is presented below.

```
ORACLE(realm-config)#dos-action-at-session no-deny
```

Two other **realm-config** parameters, which the system uses for security functions, are also required for this feature:

- You must set **access-control-trust-level** for the **realm-config** to **high**. This feature is not applicable for realms where **access-control-trust-level** is **none**, **medium** or **low**.
  If configuring **dos-action-at-session** on a **session-agent**, that agent must be associated with a realm that has **access-control-trust-level** set to **high**
- You must set the **deny-period** for the **realm-config**. This feature uses this value as the timer that specifies the length of time before denied endpoints can try to start sessions again.

Required parameters that apply only to this feature include:

- **dos-action-at-session**—Defines the feature mode for either the applicable **session-agent** or **realm-config**. On a realm. the default value is none; on a session agent the default value is inherit.

- **max-inbound-per-session-burst-rate**—Defines the maximum allowed burst rate of requests/responses received in a session. The default value is 30.
- **burst-rate-window-per-session**—Defines the total window size (in seconds) for which the system monitors packet burst rate in a session. The system run this timer window using your configured value, and resets it to 0 on a timeout. This timer window runs in a loop after each expiry. The default value is 1.

> **Note:**
>
> If you try to enable the **dos-action-at-session** feature on a realm that is not set to the high trust level, the system rejects the configuration. If you enable the **dos-action-at-session** feature on a session agent that operates in a realm that is not set to high trust level, the system allows the configuration, but provides verify messages indicating that the system is ignoring that configuration.

Configuration Precedence Behavior

For this feature, the **session-agent** configuration takes precedence over the **realm-config** configuration. The precedence will be decided based on **dos-action-at-session** parameter first.

- If you set **dos-action-at-session** in **session-agent** to **permit**, **no-deny** or **session-drop** then **session-agent** config parameters will be used.
- If **dos-action-at-session** in a **session-agent** is **inherit**, the system checks the**dos-action-at-session** in the applicable **realm-config**.
  If you have set the **realm-config** to a valid value, the system uses your **realm-config** values for **max-inbound-per-session-burst-rate**, **burst-rate-window-per-session** and **dos-action-at-session** on this **session-agent** traffic.
- If you set the **dos-action-at-session** parameter on a **session-agent** to **none**, the feature is disabled for this **session-agent**.
- Based on feature inheritance, this feature is not enabled if you leave **dos-action-at-session** set to:
  - **inherit** (default) on the applicable **session-agent** and
  - **none** (default) on the applicable **realm-config**

Configuring DoS Security

This section explains how to configure the Oracle Communications Session Border Controller for DoS protection.

Configuration Overview

Configuring Oracle Communications Session Border Controller DoS protection includes masking source IP and port parameters to include more than one match and configuring guaranteed minimum bandwidth for trusted and untrusted signaling path. You can also configure signaling path policing parameters for individual source addresses. Policing parameters are defined as peak data rate (in bytes/sec), average data rate (in bytes/sec), and maximum burst size.

You can configure deny list rules based on the following:

- ingress realm
- source IP address
- source port
- transport protocol (TCP/UDP)
- application protocol (SIP or H.323)

Exercise caution when configuring ACLs, noting that the syntax of your entry is correct. The SBC sets ACL fields with incorrect syntax to their defaults.

For example, the default source IP address for an ACL is 0.0.0.0. If using dynamic ACLs, this default address can overwrite the applicable realm's default ACL. If this ACL also has the default trust level of **none**, it would prevent the SBC from promoting any traffic on that realm to trusted.

Confirm the syntax of your configured ACLs before you save and activate them.

Changing the Default Oracle Communications Session Border Controller Behavior

The Oracle Communications Session Border Controller automatically creates permit untrusted ACLs that let all sources (address prefix of 0.0.0.0/0) reach each configured realm's signaling interfaces, regardless of the realm's address prefix. To deny sources or classify them as trusted, you create static or dynamic ACLs, and the global permit untrusted ACL to specifically deny sources or classify them as trusted. Doing this creates a default permit-all policy with specific deny and permit ACLs based on the realm address prefix.

You can change that behavior by configuring static ACLs for realms with the same source prefix as the realm's address prefix; and with the trust level set to the same value as the realm. Doing this prevents the permit untrusted ACLs from being installed. You then have a default deny all ACL policy with specific static permit ACLs to allow packets into the system.

Example 1 Limiting Access to a Specific Address Prefix Range

The following example shows how to install a permit untrusted ACL of source 12.34.0.0/16 for each signalling interface/port of a realm called access. Only packets from within the source address prefix range 12.34.0.0/16, destined for the signaling interfaces/port of the realm named access, are allowed. The packets go into untrusted queues until they are dynamically demoted or promoted based on their behavior. All other packets are denied/dropped.

- Configure a realm called access and set the trust level to low and the address prefix to 12.34.0.0/16.
- Configure a static ACL with a source prefix of 12.34.0.0/16 with the trust level set to low for the realm named access.

Example 2 Classifying the Packets as Trusted

Building on Example 1, this example shows how to classify all packets from 12.34.0.0/16 to the realm signaling interfaces as trusted and place them in a trusted queue. All other packets from outside the prefix range destined to the realm's signaling interfaces are allowed and classified as untrusted; then promoted or demoted based on behavior.

You do this by adding a global permit untrusted ACL (source 0.0.0.0) for each signaling interface/port of the access realm. You configure a static ACL with a source prefix 12.34.0.0/16 and set the trust level to high.

Adding this ACL causes the Oracle Communications Session Border Controller to also add a permit trusted ACL with a source prefix of 12.34.0.0/16 for each signaling interface/port of the access realm. This ACL is added because the trust level of the ACL you just added is high and the realm's trust level is set to low. The trust levels must match to remove the global permit trusted ACL.

Example 3 Installing Only Static ACLs

This example shows you how to prevent the Oracle Communications Session Border Controller from installing the global permit (0.0.0.0) untrusted ACL.

- Configure a realm with a trust level of none.
- Configure static ACLs for that realm with the same source address prefix as the realm's address prefix, and set the trust level to any value.

The system installs only the static ACLs you configure.

Access Control List Configuration

To configure access control lists:

1. Access the access-control configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# access-control
ACMEPACKET(access-control)#
```

2. Type select to choose and configure an existing object.

```
ACMEPACKET(access-control)# select
<src-ip>:
1: src 0.0.0.0; 0.0.0.0; realm01; ; ALL
```

3. **realm-id**—Enter the ID of the host's ingress realm.
4. **source-address**—Enter the source IPv4 address and port number for the host in the following format:

```
<IP address>[/number of address bits>][:<port>][/<port bits>]
```

For example:

```
10.0.0.1/24:5000/14
10.0.0.1/16
10.0.0.1/24:5000
10.0.0.1:5000
```

You do not need to specify the number of address bits if you want all 32 bits of the address to be matched. You also do not need to specify the port bits if you want the exact port number matched. If you do not set the port mask value or if you set it to 0, the exact port number will be used for matching. The default value is **0.0.0.0**.

5. **destination-address**—(Is ignored if you configure an application protocol in step 6.) Enter the destination IPv4 address and port for the destination in the following format:

   ```
   <IP address>[/number of address bits>][:<port>[/<port bits>]]
   ```

   You do not need to specify the number of address bits if you want all 32 bits of the address to be matched. You also do not need to specify the port bits if you want the exact port number matched. If you do not set the port mask value or if you set it to 0, the exact port number will be used for matching. The default value is **0.0.0.0**.

6. **application-protocol**—Enter the application protocol type for this ACL entry. The valid values are:

   - SIP | H.323 | None

   > **Note:**
   >
   > If application-protocol is set to none, the destination-address and port will be used. Ensure that your destination-address is set to a non-default value (0.0.0.0.)

7. **transport-protocol**—Select the transport-layer protocol configured for this ACL entry. The default value is **ALL**. The valid values are:

   - ALL | TCP | UDP

8. **access**—Enter the access control type or trusted list based on the trust-level parameter configuration for this host. The default value is **permit**. The valid values are:

   - **permit**—Puts the entry into the untrusted list. The entry is promoted or demoted according to the trust level set for this host.
   - **deny**—Puts the entry in the deny list.

9. **average-rate-limit**—Indicate the sustained rate in bytes per second for host path traffic from a trusted source within the realm. The default value is **0**. A value of **0** means policing is disabled. The valid range is:

   - Minimum—0
   - Maximum—999999999

10. **trust-level**—Indicate the trust level for the host with the realm. The default value is **none**. The valid values are:

    - **none**—Host is always untrusted. It is never promoted to the trusted list or demoted to the deny list.
    - **low**—Host can be promoted to the trusted list or demoted to the deny list.
    - **medium**—Host can be promoted to the trusted list but is only demoted to untrusted. It is never added to the deny list.
    - **high**—Host is always trusted.

11. **invalid-signal-threshold**— Enter the number of invalid signaling messages that trigger host demotion. The value you enter here is only valid when the trust level is low or medium. Available values are:

    - Minimum—Zero (0) is disabled.
    - Maximum—999999999

If the number of invalid messages exceeds this value based on the tolerance window parameter, configured in the media manager, the host is demoted.

The tolerance window default is 30 seconds. Bear in mind, however, that the system uses the same calculation it uses for specifying "recent" statistics in show commands to determine when the number of signaling messages exceeds this threshold. This calculation specifies a consistent start time for each time period to compensate for the fact that the event time, such as a user running a show command, almost never falls on a time-period's border. This provides more consistent periods of time for measuring event counts.

The result is that this invalid signal count increments for two tolerance windows, 60 seconds by default, within which the system monitors whether or not to demote the host. The signal count for the current tolerance window is always added to the signal count of the previous tolerance window and compared against your setting.

12. **maximum-signal-threshold**—Set the maximum number of signaling messages the host can send within the tolerance window. The value you enter here is only valid when the trust level is low or medium. The default value is **0**, disabling this parameter. The valid range is:

    - Minimum—0
    - Maximum—999999999

    If the number of messages received exceeds this value within the tolerance window, the host is demoted.

13. **untrusted-signal-threshold**—Set the maximum number of untrusted messages the host can send within the tolerance window. Use to configure different values for trusted and untrusted endpoints for valid signaling message parameters. Also configurable per realm. The default value is **0**, disabling this parameter. The valid range is:

    - Minimum—0
    - Maximum—999999999

14. **deny-period**—Indicate the time period in seconds after which the entry for this host is removed from the deny list. The default value is **30**. The valid range is:

    - Minimum—0
    - Maximum—999999999

15. **nat-trust-threshold**—Enter the number of endpoints behind a NAT that must be denied for the Oracle Communications Session Border Controller to demote the NAT device itself to denied (dynamic demotion of NAT devices). The default is 0, meaning dynamic demotion of NAT devices is disabled. The range is from 0 to 65535.

    The following example shows access control configured for a host in the external realm.

```
access-control
        realm-id                        external
        source-address                  192.168.200.215
        destination-address             192.168.10.2:5000
        application-protocol            SIP
        transport-protocol              ALL
        access                          permit
        average-rate-limit              3343
        trust-level                     low
        invalid-signal-threshold        5454
        maximum-signal-threshold        0
```

```
        untrusted-signal-threshold      0
        deny-period                     0
```

The following example of how to configure a blocklist entry:

```
access-control
        realm-id                        external
        source-address                  192.168.200.200
        destination-address             192.168.10.2:5000
        application-protocol            SIP
        transport-protocol              ALL
        access                          deny
        average-rate-limit              0
        trust-level                     none
        invalid-signal-threshold        0
        maximum-signal-threshold        0
        untrusted-signal-threshold      0
        deny-period                     0
```

Packet Loss Alarms for Access Control Lists

The Oracle Communications Session Border Controller reports packet loss on traffic associated with Access Control Lists (ACLs) using alarms. These alarms use the Oracle Communications Session Border Controller's system's alarm management and user display mechanisms. The user can configure three **media-manager** parameters to set thresholds for these alarms.

Packet drops occur when the system enforces its traffic management rules. These alarms require user configuration using media manager threshold parameters as a percentage of traffic for each type of ACL. When traffic volume exceeds these thresholds, the Oracle Communications Session Border Controller generates these alarms every 30 seconds until the traffic falls back below the threshold. The system synchronizes this configuration and current status with High Availability (HA) processes, maintaining these alarms across failover events.

The Oracle Communications Session Border Controller ACL traffic categories that have complementary alarms are Untrusted ACL and Trusted ACL.

The applicable media manager threshold parameters include:

- **untrusted-drop-threshold**
- **trusted-drop-threshold**

Applicable alarms include:

- ACL_UNTRUSTED_DROP_OVER_THRESHOLD
- ACL_TRUSTED_DROP_OVER_THRESHOLD

The syntax below shows how to set the untrusted drop threshold.

```
OC-SBC>enable
OC-SBC#configure terminal
OC-SBC(config)#media-manager
OC-SBC(media-manager)#media-manager-config
OC-SBC(media-manager-config)#select
```

```
OC-SBC(media-manager-config)#untrusted-drop-threshold 70
OC-SBC(media-manager-config)#done
```

Refer to the platform support table above. The range for all thresholds is 0 to 100, with defaults of 0. The default of 0 disables the threshold. All of these parameters are real-time configurable.

Packet Loss Trap for Access Control Lists

You can configure the Oracle Communications Session Border Controller (SBC) to generate an SNMP trap upon the expiration of a configurable time period during which the ACL packet drop ratio has exceeded a configured drop threshold. This trap reports the total number of dropped packets in that time period. The feature is disabled by default, and requires SNMP traps and DoS enabled.

To enable this feature, set either the **untrusted-drop-threshold** or the **trusted-drop-threshold** field in **media-manager** to a non-zero value between 1-100, then save and activate configuration changes. To disable, set both back to zero. Changes to these parameters do not require a reboot.

The feature also uses the **media-manager**'s **acl-monitor-window** to specify the monitoring window. The default value is 30 seconds, and the range is 5 to 3600 seconds. To change the monitoring window, set the **acl-monitor-window** to the desired value in seconds. Changes to the **acl-monitor-window** requires a reboot.

This SNMP trap includes the following information:

- The drop type (which is the same as the ACL type)
- The number of dropped packets during the monitored time period.
- The drop ratio during the monitored time period, reported as permillage (per thousand).

If the monitoring period expires and the percentage of dropped packets in that period is below the configured percentage value, the system does not send the trap, but does create a log entry in **log.platform** with the dropped packet value.

The following MIB objects are included in the **ap-agentcapability.mib** to support this feature.

```
apAclDropMibCapabilities 1.3.6.1.4.1.9148.2.1.31
apAclDropCap 1.3.6.1.4.1.9148.2.1.31.1
description "Acme Packet Agent Capability for ACL drop monitoring MIB"
```

The **ap-apps.mib** includes the following MIB objects to support this feature.

```
apAppsAclNotif 1.3.6.1.4.1.9148.3.16.2.2.4
apAppsAclNotifications 1.3.6.1.4.1.9148.3.16.2.2.4.0
apAclDropOverThresholdTrap 1.3.6.1.4.1.9148.3.16.2.2.4.0.1
description "The trap will be generated when acl drop ratio has exceeded the
configured threshold"

apAclDropOverThresholdClearTrap 1.3.6.1.4.1.9148.3.16.2.2.4.0.2
description "The trap will be generated when acl drop ratio has gone below
the configured threshold"

apAclNotificationGroups 1.3.6.1.4.1.9148.3.16.3.2.4
apAclDropNotificationsGroup 1.3.6.1.4.1.9148.3.16.3.2.4.1
description "Traps to monitor acl drops"
```

```
apAppsAclObjects 1.3.6.1.4.1.9148.3.16.4
apAclDropObjects 1.3.6.1.4.1.9148.3.16.4.1
apAclDropType 1.3.6.1.4.1.9148.3.16.4.1.1
description "ACL drop type"

apAclDropCount 1.3.6.1.4.1.9148.3.16.4.1.2
description "ACL drop count within monitor time window"

apAclDropRatio 1.3.6.1.4.1.9148.3.16.4.1.3
description "ACL drop ratio as permillage of current time window. Valid range
0-1000"
```

This feature is supported on HA configurations.

Host Access Policing

You can configure the Oracle Communications Session Border Controller to police the overall bandwidth of the host path.

To configure host access policing:

1. Access the **media-manager-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

2. Type select to begin editing.

   ```
   ORACLE(media-manager-config)# select

   ORACLE(media-manager-config)#
   ```

3. **max-signaling-bandwidth**—Set the maximum bandwidth available for the host path in bytes per second, which includes signaling messages from trusted sources, untrusted sources, and any management traffic on media ports. This setting applies to the following platforms, only: Acme Packet 4600, Acme Packet 6100, Acme Packet 6300, and Acme Packet 6350. Default: 1000000. Range: 71000-10000000.
4. **max-signaling-packet**—Set the maximum bandwidth available for the host path in packets per second, which includes signaling messages from trusted sources, untrusted sources, and any management traffic on media ports. This setting applies to the following platforms, only: Acme Packet 1100, Acme Packet 3900, Acme Packet 3950, Acme Packet 4900, and virtual. The default setting corresponds to the maximum value allowed by the platform, as follows:
   - Acme Packet 1100: 10000
   - Acme Packet 3900: 40000
   - Acme Packet 3950/4900: 110000
   - Virtual: System dependent.

5. **max-untrusted-signaling**—Set the percentage of the maximum signaling bandwidth you want to make available for messages coming from untrusted sources. This bandwidth is available only when not being used by trusted sources. Default: 100. Range:1-100.
6. **min-untrusted-signaling**—Set the percentage of the maximum signaling bandwidth you want reserved for untrusted sources. The rest of the bandwidth is available for trusted resources, but can also be used for untrusted sources per max-untrusted-signaling. Default: 30. Range: 1-100.
7. **fragment-msg-bandwidth**—(Only available on the Acme Packet 3820 and Acme Packet 4500) Enter the amount of bandwidth to use for the fragment packet queue. When set to 0, the Oracle Communications Session Border Controlleruses the same queue for and sharing bandwidth between untrusted packets and fragment packets. Default: zero. Range: 0-10000000.
8. **tolerance-window**—Set the size of the window used to measure host access limits for measuring the invalid message rate and maximum message rate for the realm configuration. Default: 30. Range: 0-999999999.
9. Save and activate the configuration.

Configuring ARP Flood Protection

You do not need to configure the Oracle Communications Session Border Controller to enable the use of two separate ARP queues; that feature is enabled automatically.

If you want to configure the ARP queue policing rate, you can do so in the media manager configuration.

> **Note:**
>
> this feature is not RTC-supported, and you must reboot your Oracle Communications Session Border Controller in order for your configuration changes to take effect.

To set the ARP queue policing rate:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Enter **media-manager** and press <Enter:.

```
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

4. **arp-msg-bandwidth**—Enter the rate at which you want the Oracle Communications Session Border Controller to police the ARP queue; the value you enter is the bandwidth limitation in bytes per second. The default value is **32000**. The valid range is:
   - Minimum—2000
   - Maximum—200000

5. Save your configuration.
6. Reboot your Oracle Communications Session Border Controller.

Access Control for a Realm

Each host within a realm can be policed based on average rate, peak rate, and maximum burst size of signaling messages. These parameters take effect only when the host is trusted. You can also set the trust level for the host within the realm. All untrusted hosts share the bandwidth defined for the media manager: maximum untrusted bandwidth and minimum untrusted bandwidth.

To configure access control for a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **addr-prefix**—Set the IP address prefix used to determine if an IP address is associated with the realm. This value is then associated with the ACLs you create to determine packet access. The default value is **0.0.0.0.**
5. **average-rate-limit**—Set the sustained rate for host path traffic from a trusted source within the realm in bytes per second. The default value is zero (**0**), disabling this parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295
6. **access-control-trust-level**—Set the trust level for the host within the realm. The default value is **none**. The valid values are:
   - **none**—Host is always untrusted. It is never promoted to the trusted list or demoted to the deny list.
   - **low**—Host can be promoted to the trusted list or demoted to the deny list.
   - **medium**—Host can be promoted to the trusted list but is only demoted to untrusted. It is never added to the deny list.
   - **high**—Host is always trusted.
7. **invalid-signal-threshold**— Enter the number of invalid signaling messages that trigger host demotion. The value you enter here is only valid when the trust level is low or medium. Available values are:
   - Minimum—Zero (0) is disabled.
   - Maximum—999999999

If the number of invalid messages exceeds this value based on the tolerance window parameter, configured in the media manager, the host is demoted.

The tolerance window default is 30 seconds. Bear in mind, however, that the system uses the same calculation it uses for specifying "recent" statistics in show commands to determine when the number of signaling messages exceeds this threshold. This calculation specifies a consistent start time for each time period to compensate for the fact that the event time, such as a user running a show command, almost never falls on a time-period's border. This provides more consistent periods of time for measuring event counts.

The result is that this invalid signal count increments for two tolerance windows, 60 seconds by default, within which the system monitors whether or not to demote the host. The signal count for the current tolerance window is always added to the signal count of the previous tolerance window and compared against your setting.

8. **maximum-signal-threshold**—Set the maximum number of signaling messages one host can send within the window of tolerance. The host is demoted if the number of messages received by the Oracle Communications Session Border Controller exceeds the number set here. Valid only when the trust level is set to low or medium. The default value is zero (**0**), disabling this parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295
9. **untrusted-signal-threshold**—Set the maximum number of untrusted messages the host can send within the tolerance window. Use to configure different values for trusted and untrusted endpoints for valid signaling message parameters. Also configurable per realm. The default value is zero (**0**), disabling the parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295
10. **deny-period**—Set the length of time an entry is posted on the deny list. The host is deleted from the deny lost after this time period. The default value is **30**. A value of **0** disables the parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295
11. **nat-trust-threshold**—Enter the number of endpoints behind a NAT that must be denied for the Oracle Communications Session Border Controller to demote the NAT device itself to denied (dynamic demotion of NAT devices). The default is 0, meaning dynamic demotion of NAT devices is disabled. The range is from 0 to 65535.

    The following example shows a host access policing configuration.

```
realm-config
        identifier                      private
        addr-prefix                     192.168.200.0/24
        network-interfaces
                                        prviate:0
        mm-in-realm                     disabled
        mm-in-network                   enabled
        msm-release                     disabled
        qos-enable                      disabled
        max-bandwidth                   0
        ext-policy-svr
        max-latency                     0
```

```
max-jitter                      0
max-packet-loss                 0
observ-window-size              0
parent-realm
dns-realm
media-policy
in-translationid
out-translationid
class-profile
average-rate-limit              8000
access-control-trust-level      medium
invalid-signal-threshold        200
maximum-signal-threshold        0
untrusted-signal-threshold      500
deny-period                     30
symmetric-latching              disabled
pai-strip                       disabled
trunk-context
```

Configuring Overload Protection for Session Agents

The Oracle Communications Session Border Controller offers two methods to control SIP registrations to smooth the registration flow.

You can limit the:

- number of new register requests sent to a session agent (using the **max-register-sustain-rate** parameter)
- burstiness which can be associated with SIP registrations

The first method guards against the Oracle Communications Session Border Controller's becoming overwhelmed with register requests, while the second method guards against a transient registration that can require more than available registration resources.

SIP registration burst rate control allows you to configure two new parameters per SIP session agent—one that controls the registration burst rate to limit the number of new registration requests, and a second to set the time window for that burst rate. When the registration rate exceeds the burst rate you set, the Oracle Communications Session Border Controller responds to new registration requests with 503 Service Unavailable messages.

Note that this constraint is not applied to re-registers resulting from a 401 Unauthorized challenge request.

To configure overload protection for session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. **constraints**—Enable this parameter to set the sustained rate window constraint you configure in the next step. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. **sustain-rate-window**—Enter a number to set the sustained window period (in milliseconds) that is used to measure the sustained rate. The default value is zero (**0**). The valid range is:
   - Minimum—10
   - Maximum—4294967295

   The value you set here must be higher than or equal to the value you set for the burst rate window.

   > **Note:**
   >
   > If you are going to use this parameter, you must set it to a minimum value of 10.
6. **max-register-sustain-rate**—Enter a number to set the maximum number of registrations per second you want sent to the session agent. The default value is zero (**0**), disabling the parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295
7. **register-burst-window**—Define the window size in seconds for the maximum number of allowable SIP registrations. 0 is the minimum and default value for this parameter; the maximum value is 999999999.
8. **max-register-burst-rate**—Enter the maximum number of new registrations you want this session agent to accept within the registration burst rate window. If this threshold is exceeded, the Oracle Communications Session Border Controller will respond to new registration requests with 503 Service Unavailable messages. 0 is the minimum and default value for this parameter; the maximum value is 999999999.
9. Save and activate your configuration.

DDoS Protection from Devices Behind a NAT

A DDoS attack could be crafted such that multiple devices from behind a single NAT could overwhelm the Oracle Communications Session Border Controller. The Oracle Communications Session Border Controller would not detect this as a DDoS attack because each endpoint would have the same source IP but multiple source ports. Because the Oracle Communications Session Border Controller allocates a different CAM entry for each source IP:Port combination, this attack will not be detected. This feature remedies such a possibility.

Restricting the Number of Endpoints behind a NAT

Each new source IP address and source IP port combination now counts as an endpoint for a particular NAT device. After the configured value of a single NAT's endpoints is reached, subsequent messages from behind that NAT are dropped and the NAT is demoted. This is set with the max-endpoints-per-nat parameter located in both the access-control and realm-config configuration elements.

Counting Invalid Messages from Endpoints behind a NAT

The Oracle Communications Session Border Controller also counts the number of invalid messages sent from endpoints behind the NAT. Once a threshold is reached, that NAT is demoted. Numerous conditions are counted as Errors/Invalid Messages from an endpoint. The aggregate of all messages from endpoints behind the NAT are counted against the NAT device, in addition to the existing count against the endpoint. This threshold is set with the nat-invalid-message-threshold parameter located in both the access-control and realm-config configuration elements.

As a unique case, the absence of a REGISTER message following a 401 response is counted as an invalid message from the end point. And if that endpoint is behind a NAT, this scenario will be counted as invalid message from that NAT device as well. You set a timeout period in which the REGISTER message must arrive at the Oracle Communications Session Border Controller. This period is set with the wait-time-for-invalid-register parameter located in the realm config.

DDoS Protection Configuration realm-config

To set the DDoS Protection from devices behind NATs in the realm-config:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **max-endpoints-per-nat**— Set the maximum number of endpoints that can exist behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.
4. **nat-invalid-message-threshold**—Set the maximum number of invalid messages that may originate behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.

5. **wait-time-for-invalid-register**—Set the period (in seconds) that the Oracle Communications Session Border Controller counts before considering the absence of the REGISTER message as an invalid message.
6. Type **done** to save your configuration.

DDoS Protection Configuration access-control

To set the DDoS Protection from devices behind NATs in the access-control configuration element:

1. Access the access-control configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# access-control
ORACLE(access-control)#
```

2. Type select to choose and configure an existing object.

```
ORACLE(access-control)# select
<src-ip>:
1: src 0.0.0.0; 0.0.0.0; realm01; ; ALL
selection:1
```

3. **max-endpoints-per-nat**— Set the maximum number of endpoints that can exist behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.
4. **nat-invalid-message-threshold**—Set the maximum number of invalid messages that may originate behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.
5. Type **done** to save your work and continue.

SNMP Trap support

The Oracle Communications Session Border Controller sends the following trap when a NAT device is blocklisted due to the triggers listed. The trap does not include reasons, which are available from the syslogs.

```
apSysMgmtExpDOSTrap     NOTIFICATION-TYPE
        OBJECTS         { apSysMgmtDOSIpAddress,  apSysMgmtDOSRealmID ,
                          apSysMgmtDOSFromUri }
        STATUS          deprecated
        DESCRIPTION
              "This trap is generated when an IP is placed on a blocklist due
              to denial-of-service attempts, and provides the IP address that
              has been demoted, the realm-id of that IP, and (if available)
              the URI portion of the SIP From header of the message that
              caused the demotion."
        ::= { apSysMgmtDOSNotifications 2 }
```

Ensure that the enable-snmp-monitor-traps parameter in the system-config configuration element is enabled for the Oracle Communications Session Border Controller to send out this trap.

Syslog Support

Set the syslog-on-demote-to-deny parameter in the media-manager-config to enabled to generate syslog on endpoint demotion from untrusted to deny. NAT device demotion will also generate a unique syslog message with accompanying text explaining that it is the NAT device demotion event.

Debugging

The show sip acl command now includes counts of Blocked NAT devices.

```
ACMEPACKET# show sipd acl
13:57:28-71
SIP ACL Status            -- Period -- -------- Lifetime --------
                Active    High   Total       Total  PerMax     High
Total Entries        0       0       0           0       0        0
Trusted              0       0       0           0       0        0
Blocked              0       0       0           0       0        0
Blocked NATs         0       0       0           0       0        0
ACL Operations         ---- Lifetime ----
                Recent      Total  PerMax
ACL Requests         0          0       0
Bad Messages         0          0       0
Promotions           0          0       0
Demotions            0          0       0
Trust->Untrust       0          0       0
Untrust->Deny        0          0       0
```

DoS Threshold Configuration

This section describes how to configure DoS traffic thresholds to alert you of high traffic conditions.

1. Access the media-manager configuration element

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)# select
   ```

2. **dos-guard-window**—Specifies the window of time for measuring traffic volume within which the DoS alert thresholds may be triggered. When the window expires, the system checks to see if the traffic has fallen below the configured thresholds. If so, the system sets the counters back to zero. If not, the system waits for the duration of the window and checks again. The system does this 5 times.
3. **untrusted-minor-threshold**—Defines the percentage of the untrusted bandwidth that triggers a minor alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.

4. **untrusted-major-threshold**—Specifies the percentage of the untrusted bandwidth that triggers a major alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
5. **untrusted-critical-threshold**—Specifies the percentage of the untrusted bandwidth that triggers a critical alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
6. **trusted-minor-threshold**—Specifies the percentage of the trusted bandwidth that triggers a minor alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
7. **trusted-major-threshold**—Specifies the percentage of the trusted bandwidth that triggers a major alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
8. **trusted-critical-threshold**—Specifies the percentage of the trusted bandwidth that triggers a critical alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
9. **arp-minor-threshold**—Specifies the percentage of the arp bandwidth that triggers a minor alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
10. **arp-major-threshold**— Specifies the percentage of the arp bandwidth that triggers a major alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
11. **arp-critical-threshold**— Specifies the percentage of the arp bandwidth that triggers a critical alert for this threshold. When triggered, the system sends an alarm and a trap. Set the value to zero to disarm this threshold for alert events.
12. Type **done** to save your configuration.

Configure DOS Protection at the Session Level

You configure Session Level DoS Protection on the SBC on either the **realm-config** or the **session-agent** elements. The **session-agent** configuration takes precedence. This procedure makes this configuration on a **realm-config**, which allows the procedure to include required realm configuration which is not available from a **session-agent**. These settings are still, however, required on the realm to which that **session-agent** belongs.

To set the **dos-action-at-session** parameter:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Access the session-router branch.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-config** and press Enter.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. Select the applicable agent or create a new one.

5. **access-control-trust-level**——For session-based DoS protection, you must set this parameter to high.
   - default - none
   - High
   - Medium
   - Low
6. **dos-action-at-session**—Specify the system's behavior for reacting to session-based DoS attacks.
   - none—(default) The system takes no action during a DoS attack.
   - permit—If the endpoint initiates the DoS attack at the session level, the system can demote or deny the endpoint. At first detection of a DoS attack, the system demotes the endpoint from trusted to untrusted. If there is a second DoS attack before the UNTRUST_TMO timer expires, the system further demotes the endpoint to deny.
   - no-deny—If the endpoint initiates the DoS attack at the session level, the system can demote the endpoint to untrusted. When the UNTRUST_TMO timer expires, the system promotes the endpoint back to the trusted state.
   - session-drop—If the endpoint initiates the DoS attack at the session level, the system takes action on that session only. Specifically, the system terminates the existing session but does not demote or deny the endpoint.

   When configuring this feature on a **session-agent**, the value "inherit" is also available. For **session-agent**, "inherit" is the default value for the parameter, and instructs the system to use your session-level DoS configuration on the applicable **realm-config**.
7. **max-inbound-per-session-burst-rate**—Defines the max allowed burst rate of requests/responses received in a session. Default value is 30. It can be configured to higher value based on the actual packet runrate.
   - default - 30
   - Range - 0 - 999999999
8. **burst-rate-window-per-session**—Defines the total window size (in seconds) for which the burst rate of packets will be monitored in a session. A timer window will start with value configured in burst-rate-window-per-session and the timer window will reset to 0 on timeout. The timer window will run in loop after each expiry. Default value is 1, which means 1 sec monitoring window will run.
   - default - 1
   - Range - 0 - 999999999
9. **deny-period**—The system uses this configured values as the window within which an endpoint can be promoted from denied to untrusted.
10. Save your configuration.

This feature is RTC-supported

Media Policing

Media policing controls the throughput of individual media flows in the Oracle Communications Session Border Controller, which in turn provides security and bandwidth management functionality. The media policing feature works for SIP, H.323 and SIP-H.323 protocols. The media policing feature also lets you police static flows and RTCP flows.

The term media policing refers to flows that go through the Oracle Communications Session Border Controller. Flows that are directed to the host application are not affected by media policing.

You can use media policing to protect against two potential security threats that can be directed against your Oracle Communications Session Border Controller:

- Media DoS—Once media flows are established through the Oracle Communications Session Border Controller, network resources are open to RTP media flooding. You can eliminate the threat of a media DoS attack by constraining media flows to absolute bandwidth thresholds.
- Bandwidth Piracy—Bandwidth policing ensures that sessions consume no more bandwidth than what is signaled for.

Policing Methods

The Oracle Communications Session Border Controller polices real-time traffic by using Constant Bit Rate (CBR) media policing. CBR policing is used when a media flow requires a static amount of bandwidth to be available during its lifetime. CBR policing best supports real-time applications that have tightly constrained delay variation. For example, voice and video streaming are prime candidates for CBR policing.

Session Media Flow Policing

Session media encompasses RTP and RTCP flows. In order to select policing constraints for these flows, the Oracle Communications Session Border Controller watches for the codec specified in an SDP or H.245 message. When a match is made between the codec listed in an incoming session request and a configured **media-profile** configuration element, the Oracle Communications Session Border Controller applies that **media-profile**'s bandwidth policing constraint to the media flow about to start.

If multiple codecs are listed in the SDP message, the Oracle Communications Session Border Controller will use the **media-profile** with the most permissive media policing constraints for all of the flows associated with the session. If a codec in the H.245/SDP message is not found in any configured **media-profile**, the Oracle Communications Session Border Controller uses the **media-profile** with the most permissive media policing constraints configured. If no **media-profiles** are configured, there will be no session media flow policing.

If a mid-call change occurs, bandwidth policing is renegotiated.

Configuration Notes

Review the following information before configuring your Oracle Communications Session Border Controller to perform media policing.

Session Media Flow Policing

Session media flow policing applies to both RTP and RTCP flows. Setting either of the parameters listed below to 0 disables media policing, letting RTP or RTCP flows pass through the system unrestricted.

- RTP Policing
  - Set in the **media-profile** configuration element's **average-rate-limit** parameter to police RTP traffic with the CBR policing method.
  - **average-rate-limit**—Establishes the maximum speed for a flow in bytes per second.

- RTCP Policing
  - Set in the **media-manager-config** configuration element's **rtcp-rate-limit** parameter to police RTCP traffic with the CBR policing method.
  - **rtcp-rate-limit**—Establishes the maximum speed for an RTCP flow in bytes per second.

Static Flow Policing

Static flow policing is configured with one parameter found in the **static-flow** configuration element. To configure CBR, you have to set the **average-rate-limit** parameter to a non-zero value. Setting the parameter listed below to 0 disables static flow policing, effectively letting the flow pass through the Oracle Communications Session Border Controller unrestricted.

In a CBR configuration, the **average-rate-limit** parameter determines the maximum bandwidth available to the flow.

- **average-rate-limit**—Establishes the maximum speed for a static flow in bytes per second.

> **Note:**
>
> Static flow policing is not necessarily tied to any type of media traffic, it can affect flows of any traffic type.

Media Policing Configuration for RTP Flows

You can configure media policing in the **media-profile** configuration element using the ACLI. In the following example, you will configure media policing for the G723 media profile.

To configure media policing for RTP flows:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ```

4. Select an existing media profile to which you will add policing constraints.

   ```
   ORACLE(media-profile)# select
   <name>:
   1: audio    4=G723      RTP/AVP 16 0 0 0
   selection:1
   ORACLE(media-profile)#
   ```

   From this point, you can configure media policing parameters. To view all **media-profile** parameters, enter a **?** at the system prompt

5. **average-rate-limit**—Enter the maximum rate in bytes per second for any flows that this **media-profile** polices. The default value is zero (**0**), disabling media policing. The valid range is:
   - Minimum—0
   - Maximum—125000000

   Average rate limit values for common codecs:
   - PCMU—80000 Bps
   - G729—26000 Bps

   The following example shows a **media-profile** configuration element configured for media policing.

```
media-profile
        name                            G723
        media-type                      audio
        payload-type                    4
        transport                       RTP/AVP
        req-bandwidth                   16
        frames-per-packet               0
        parameters
        average-rate-limit              15000
```

Media Policing Configuration for RTCP Flows

You can configure media policing for RTCP flows by using the ACLI.

To configure media policing for RTCP flows:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the **media-manager** path.

```
ORACLE(configure)# media-manager
```

3. Type **media-manager** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

4. **rtcp-rate-limit**—Enter the RTCP policing constraint in bytes per second. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—125000000

RTP Payload Type Mapping

The Oracle Communications Session Border Controller maintains a default list of RTP payload types mapped to textual encoding names as defined in RFC 3551.

The following table defines the preconfigured payload type for standard encodings.

Payload Type	Encoding Name	Audio (A) / Video (V)	Clock Rate
0	PCMU	A	8000
4	G723	A	8000
8	PCMA	A	8000
9	G722	A	8000
15	G728	A	8000
18	G729	A	8000

If you configure any payload type to encoding name mappings, the default mappings will be ignored. You must then manually enter all payload type mappings you use in the **media-profile** configuration element.

ITU-T to IANA Codec Mapping

The Oracle Communications Session Border Controller maintains a list of ITU-T (H.245) codecs that map to IANA RTP codecs. An ITU codec is directly mapped to an IANA Encoding Name for media profile lookups. All codecs are normalized to IANA codec names before any matches are made. New ITU-T codecs can not be added to the media profiles list.

The following table defines the ITU-T to IANA codec mappings.

ITU-T	IANA
g711Ulaw64k	PCMU
g711Alaw64k	PCMA
g726	G726
G7231	G723
g728	G728
g729wAnnexB	G729
g729	G729 fmtp:18 annexb=no
H261VideoCapability	H261
H263VideoCapability	H263
t38Fax	T38

SDP Anonymization

In order to provide an added measure of security, the Oracle Communications Session Border Controller's topology-hiding capabilities include SDP anonymization. Enabling this feature gives the Oracle Communications Session Border Controller the ability to change or modify certain values in the SDP so that malicious parties will be unable to learn information about your network topology.

To do this, the Oracle Communications Session Border Controller hides the product-specific information that can appear in SDP o= lines and s= lines. This information can include usernames, session names, and version fields. To resolve this issues, the Oracle Communications Session Border Controller makes the following changes when you enable SDP anonymization:

- Sets the session name (or the s= line in the SDP) to s=-
- Sets the username in the origin field to -SBC

- Sets the session ID in the origin field to an integer of incrementing value

Note that for mid-call media changes, the session identifier is not incremented.

To enable this feature, you set a parameter in the media manager configuration.

SDP Anonymization Configuration

To enable SDP anonymization:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **media-manager** again to access the media manager configuration, and press Enter.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **anonymous-sdp**—Set this parameter to enabled to use the SDP anonymization feature. When you leave this parameter empty the feature is turned off. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

Unique SDP Session ID

Codec negotiation can be enabled by updating the SDP session ID and version number. The media-manager option, **unique-sdp-id** enables this feature.

With this option enabled, the Oracle Communications Session Border Controller will hash the session ID and IP address of the incoming SDP with the current date/time of the Oracle Communications Session Border Controller in order to generate a unique session ID.

Unique SDP Session ID Configuration

To enable unique SDP session ID in media-manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. **options**—Set the options parameter by typing **options**, a Space, the option name **unique-sdp-id** with a plus sign in front of it, and then press Enter.

```
ORACLE(media-manager)# options +unique-sdp-id
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

4. Save and activate your configuration.

TCP Synchronize Attack Prevention

This section explains how the Oracle Communications Session Border Controller protects itself from a Transmission Control Protocol (TCP) synchronize (SYN) packet flooding attack sourced from a remote hostile entity.

SIP and H.323 signaling can be configured on the Oracle Communications Session Border Controller to be TCP protocol-based. In this configuration, the Oracle Communications Session Border Controller can be a target of a TCP SYN attack. The Oracle Communications Session Border Controller C is able to service new call requests throughout the duration of an attack

About SYN

SYN is used by TCP when initiating a new connection to synchronize the sequence numbers on two connecting computers. The SYN is acknowledged by a SYN-ACK by the responding computer. After the SYN-ACK, the client finishes establishing the connection by responding with an ACK message. The connection between the client and the server is then open, and the service-specific data can be exchanged between the client and the server.

A SYN flood is a series of SYN packets from forged IP addresses. The IP addresses are chosen randomly and do not provide any hint of the attacker's location. The SYN flood keeps the server's SYN queue full. Normally this would force the server to drop connections. A server that uses SYN cookies, however, will continue operating normally. The biggest effect of the SYN flood is to disable large windows.

Server Vulnerability

Vulnerability to attack occurs when the server has sent a SYN-ACK back to client, but has not yet received the ACK message; which is considered a half-open connection. The server has a data structure describing all pending connections built in its system memory. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections.

The attacking system sends SYN messages to the server that appear to be legitimate, but in fact reference a client that is unable to respond to the SYN-ACK messages. The final ACK message is never sent to the server.

The half-open connections data structure on the server fills and no new incoming connections are accepted until the table is emptied out. Typically there is a timeout associated with a pending connection (the half-open connections will eventually expire and the server will recover). But the attacking system can continue sending IP-spoofed packets requesting new connections faster than the server can expire the pending connections. The server has difficulty in accepting any new incoming network connections.

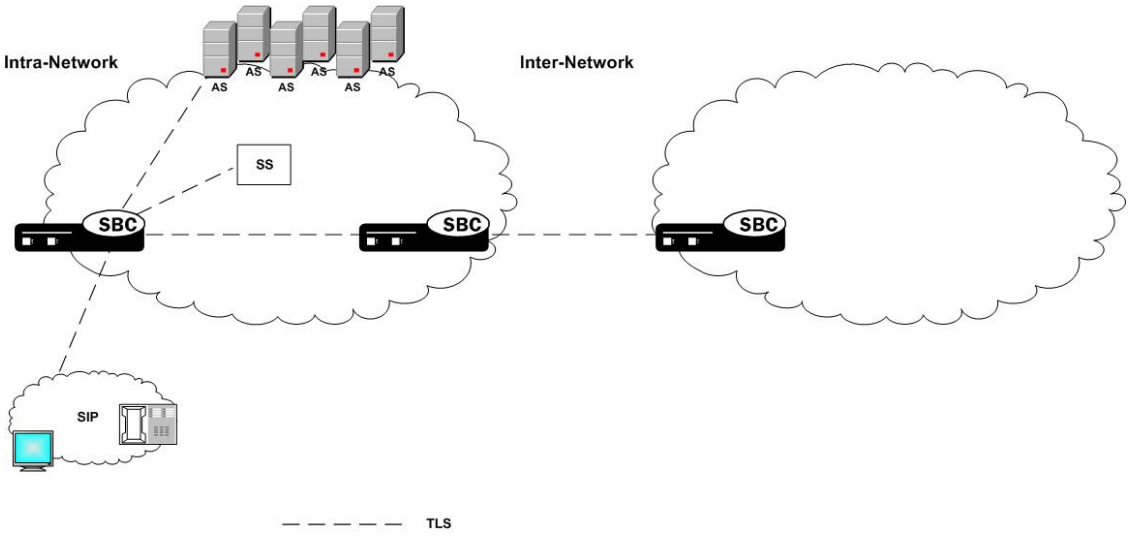
Configuring TCP SYN Attack Prevention

No configuration is necessary to enable TCP SYN attack prevention. Internal TCP protocol changes were made to provide protection.

Transport Layer Security

The Oracle Communications Session Border Controller provides support for Transport Layer Security (TLS) for SIP, which can be used to protect user and network privacy by providing authentication and guaranteeing the integrity for communications between the Oracle Communications Session Border Controller and the following:

- Another device in your network infrastructure (intra-network)
- Another Oracle Communications Session Border Controller when you are using a peering application (inter-network) for interior network signaling security
- An endpoint for authentication before allowing SIP messaging to take place



The SBC and TLS

Transport Layer Security (TLS) on the Oracle Communications Session Border Controller (SBC) depends on the presence of the Security Service Module (SSM) for hardware acceleration of encryption and decryption and random media generation. The SSM module is a plug-in that you can add to the SBC chassis given the installation of the necessary boot loader and minimum hardware revision levels.

With the required hardware revision levels, qualified field personnel can add the plug-in unit to the SBC onsite. This provision makes upgrades fast, and means that you do not need to return the SBC to Oracle manufacturing for a hardware upgrade. When you upgrade the SBC with the SSM card that supports TLS, field personnel will affix a new CLEI code label to the Oracle chassis. The code will also appear on the SSM card (also referred to as the plug-in unit) and is visible when the system's chassis cover is opened. On a new SBC provisioned with the SSM card, the code labels are already affixed in all required locations.

With the SSM card installed on the SBC, TLS support is enabled and the SSM accelerator performs:

- Diffie-Hellman
- AES256
- Random number generation

Supported Encryption

The SBC supports the following encryption:

- TLSv1.2
- TLSv1.3

The SBC requires the use of TLS Secure Renegotiation as described in RFC 5746 in order to counter the prefix attack described in CVE-2009-3555. If the devices attempting a TLS connection to the SBC don't support TLS Secure Renegotiation, the TLS handshake fails. Oracle recommends updating such devices to support TLS Secure Renegotiation.

Diffie-Hellman Key Size

In the context of TLS negotiations on SIP interfaces, the default Diffie-Hellman key size offered by the SBC is 1024 bits. The key size is set in the `diffie-hellman-key-size` attribute within the **tls-global** configuration element.

While the key size can be increased, setting the key size to 2048 bits significantly decreases performance.

Suite B and Cipher List Support

The Oracle Communications Session Border Controller (SBC) supports full control of selecting the ciphers that you want to use for Transport Layer Security (TLS). The system defaults to DEFAULT for the Cipher List parameter in the TLS Profile configuration. Oracle recommends that you delete ALL and add only the particular ciphers that you want, choosing the most secure ciphers for your deployment.

To support Suite B, the SBC certificate-record configuration includes the following parameters:

- Key Algor—Public key algorithm. Supports RSA and ECDSA. Default: RSA Security. You must select ECDSA to support suite B.
- ECDSA Key Size—ECDSA key size. Supports p256 and p384.

Configure the list of ciphers that you want to use from the **cipher-list** element in the **tls-profile** configuration. Press Tab to display the list of supported ciphers. One-by-one, you can add as many ciphers as your deployment requires.

TLS Ciphers

The **tls-profile** object contains the **cipher-list** parameter. For a complete list of supported ciphers for this release, see the *Release Notes*.

Minimum Advertised SSL/TLS Version

The sslmin option sets the minimum allowed TLS version. Use this option to mitigate the use of older, more vulnerable versions of TLS.

When the **tls-version** is set to **compatibility**, the **sslmin** option specifies the lowest TLS version allowed.

In **security-config**, the **sslmin** option values can be: tls1.2 or tls1.3.

Minimum Advertised TLS Version Configuration

The **sslmin** option sets the minimum advertised TLS version when the **tls-version** is set to **compatibility**.

1. Access the **security-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# security-config
ORACLE(security-config)#
```

2. Select the **security-config** object to edit.

```
ORACLE(security-config)#

ORACLE(security-config)#
```

3. **options**—Set the options parameter by typing **options**, a space, a plus sign, the option name **sslmin=** and then one of the valid values. Valid values are:
   - tls1.2
   - tls1.3

```
ORACLE(security-config)#options +sslmin=tls1.2
```

4. Type **done** to save your configuration.

Signaling Support

The Oracle Communications Session Border Controller ’s TLS functionality supports SIP and SIPS. In addition, the Oracle Communications Session Border Controller can accommodate a mixture of TLS and non-TLS sessions within a realm as because a request for TLS is controlled by the endpoint (TLS UA).

Endpoint Authentication

The Oracle Communications Session Border Controller does not operate as a CA. Instead, the Oracle Communications Session Border Controller ’s TLS implementation assumes that you are using one of the standard CAs for generating certificates:

- Verisign
- Entrust

- Thawte
- free Linux-based CA (for example, openssl)

> **Note:**
>
> Self-signed certificates are available only as an option for MSRP connections

The Oracle Communications Session Border Controller can generate a certificate request in PKCS10 format and to export it. It can also import CA certificates and a Oracle Communications Session Border Controller certificate in the PKCS7/X509 PEM format.

The Oracle Communications Session Border Controller generates the key pair for the certificate request internally. The private key is stored as a part of the configuration in 3DES encrypted form (with an internal generated password) and the public key is returned to the user along with other information as a part of PKCS10 certificate request.

The Oracle Communications Session Border Controller supports the option of importing CA certificates and marking them as trusted. However, the Oracle Communications Session Border Controller only authenticates client certificates that are issued by the CAs belonging to its trusted list. If you install only a specific vendor's CA certificate on the Oracle Communications Session Border Controller , it authenticates that vendor's endpoints. Whether the certificate is an individual device certificate or a site-to-site certificate does not matter because the Oracle Communications Session Border Controller authenticates the signature/ public key of the certificate.

Keeping Pinholes Open at the Endpoint

The Oracle Communications Session Border Controller provides configurable TCP NAT interval on a per-realm basis. You need to configure a NAT interval for the applicable realm to support either all conforming or all non-conforming endpoints.

- Conforming endpoints use the draft-jennings sipping-outbound-01. It describes how to keep the endpoint keeps the connection alive.

  > **Note:**
  >
  > Currently the endpoint uses REGISTER.

- Non-conforming endpoints have short NAT interval, where the HNT application with the TCP connection for TLS operates as it does for regular TCP. We give the UA a shorter expires time so that it refreshes frequently, implicitly forcing the UA to keep the TVP socket open and reuse it for further requests (in-dialog or out-of-dialog). Regular requests using TLS sent from the Oracle Communications Session Border Controller to the UA reuse the same TCP connection so that further TLS certificate exchanges are not required.

Key Usage Control

You can configure the role of a certificate by setting key usage extensions and extended key usage extensions. Both of these are configured in the certificate record configuration.

Key Usage List

This section defines the values you can use (as a list) in the **key-usage-list** parameter. You can configure the parameter with more than one of the possible values.

Value	Description
digitalSignature (default with keyEncipherment)	Used when the subject public key is used with a digital signature mechanism to support security services other than non-repudiation, certificate signing, or revocation information signing. Digital signature mechanisms are often used for entity authentication and data origin authentication with integrity.
nonRepudiation	Used when the subject public key is used to verify digital signatures that provide a non-repudiation service protecting against the signing entity falsely denying some action, excluding certificate or CRL signing.
keyEncipherment (default with digitalSignature)	Used with the subject public key is used for key transport. (For example, when an RSA key is to be used for key management.)
dataEncipherment	Used with the subject public key is used for enciphering user data other than cryptographic keys.
keyAgreement	Used with the subject public key is used key agreement. (For example, when a Diffie-Hellman key is to be used for a management key.)
encipherOnly	The keyAgreement type must also be set. Used with the subject public key is used only for enciphering data while performing key agreement.
decipherOnly	The keyAgreement type must also be set. Used with the subject public key is used only for deciphering data while performing key agreement.

Extended Key Usage List

This section defines the values you may use in the **extended-key-usage-list** parameter.

Value	Description
serverAuth (default)	Used while the certificate is used for TLS server authentication. In Oracle Communications Session Border Controller access-side deployments, the system typically acts as a TLS server accepting TLS connections. You might use this setting while generating the end-entity-cert.
clientAuth	Used while the certificate is used for TLS client authentication. In Oracle Communications Session Border Controller core-side deployments, the system typically acts as a TLS client initiating TLS connections. You might use this setting while generating the end-entity-cert.

When you enable a **tls-profile** for **mutual-authentication**, you must also configure the **extended-key-usage-list** parameter within the associated **end-entity-certificate** to both the **serverAuth** and **clientAuth** values. The default for **extended-key-usage-list** is **serverAuth** only.

4096-bit RSA Key Support

The Oracle Communications Session Border Controller (SBC) supports 4096-bit RSA keys for SIP Transport Layer Security (TLS) on all platforms. The 4096-bit support enables you to import root certificates for SIP communications secured with TLS into the SBC.

Use the **key-size** parameter in the certificate-record configuration to set the key size.

Reusing a TLS Connection

TheOracle Communications Session Border Controller supports TLS connection reuse if and when an alias is included in the Via header by the originator of the TLS connection. When this is the case, the Oracle Communications Session Border Controller reuses the same connection for any outgoing request from the Oracle Communications Session Border Controller .

TLS Configuration Process

Configuring Transport Layer Security (TLS) on the Oracle Communications Session Border Controller (SBC) includes the following steps.

1. Configure certificates. See "Configure Certificates."
2. Configure and apply the TLS profile. See "Configure a TLS Profile."

Certificate Configuration Process

The process for configuring certificates on the Oracle Communications Session Border Controller (SBC) includes the following steps:

1. Configure a certificate record on the SBC. See "Configure Certificates."
2. Generate a certificate request by the SBC. See "Generate a Certificate Request."
3. Import the certificate record into the SBC. See "Import a Certificate Using the ACLI" and "Import a Certificate Using SFTP."
4. Reboot the system.

Configure the Certificate Record

The certificate record configuration represents either the end-entity certificate or the CA certificate on the Oracle Communications Session Border Controller (SBC). When you use the certificate record for an end-entity certificate, associate a private key with the certificate record configuration by using the ACLI **generate-certificate-request** command. You can import a requested certificate provided by a CA into a certificate record configuration using the ACLI **import-certificate** command.

Do not associate a private key with the certificate record configuration, if it was issued to hold a CA certificate.

> **Note:**
>
> You do not need to create a certificate record when importing a CA certificate or certificate in PKCS #12 format.

1. Access the **certificate-record** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# certificate-record
ORACLE(certificate-record)#
```

2. For the Certificate Record configuration, do the following:
   - Name—(Required) Enter the name of this certificate record.
   - Country—Enter the country name abbreviation. For example, CA for Canada. Range: 2 characters.
   - State—Enter the region abbreviation. For example, Quebec. Range: 1-128 characters.
   - Locality—Enter the name of the locality in the region. For example, Montreal. Range:1-128 characters.
   - Organization—Enter the name of the organization. For example, Office of Information Technology. 1-64 characters.
   - Unit—Enter the name of the unit in the organization. For example, Global Network Security. 1-64 characters.
   - Common name—Enter the common name for the certificate record. For example, your name. Range: 1-64 characters.
   - Key algor—Set a key algorithm. Valid algorithms: rsa | ecdsa.
   - Digest algor—Set a digest algorithm. Valid values: sha1 | sha256 | sha384.
   - Key size—For the RSA key algorithm, set the RSA key size. Valid key size: 512 | 1024 | 2048 | 4096.
   - ECDSA key size—For the ECDSA key algorithm, set the ECDSA key size. Valid key size: p256 | p384.
   - Alternate name—(Optional) Enter one or more alternative names for the certificate holder.
   - Trusted—Do one of the following:
     - Select to make the certificate trusted. (Default)
     - Deselect to make the certificate un-trusted.
   - Key usage list—Set key the usage extensions you want to use with this certificate record. Multiple values allowed. Default: The combination of **digitalSignature** and **keyEncipherment**. For a list of possible values and their descriptions, see “Key Usage List.”
   - Extended key usage list—Set the extended key usage extensions you want to use with this certificate record. Default: **serverAuth**. For a list of possible values and their descriptions, see “Extended Key Usage List.”
   - Options—Set any optional features or parameters that you want.
3. Type **done** to save your configuration.

- Create TLS profiles, using your certificate records, to further define the encryption behavior and create the configuration element that you can apply to a SIP interface.

Generate a Certificate Request

Using the ACLI **generate-certificate-request** command allows you to generate a private key and a certificate request in PKCS10 PEM format. You take this step after you configure a certificate record.

The Oracle Communications Session Border Controller stores the private key that is generated in the certificate record configuration in 3DES encrypted form with in internally generated password. The PKCS10 request is displayed on the screen in PEM (Base64) form.

You use this command for certificate record configurations that hold end-entity certificates. If you have configured the certificate record to hold a CA certificate, then you do not need to generate a certificate request because the CA publishes its certificate in the public domain. You import a CA certificate by using the ACLI **import-certificate** command.

This command sends information to the CA to generate the certificate, but you cannot have Internet connectivity from the Oracle Communications Session Border Controller to the Internet. You can access the internet through a browser such as Internet Explorer if it is available, or you can save the certificate request to a disk and then submit it to the CA.

To run the applicable command, you must use the value you entered in the name parameter of the certificate record configuration. You run the command from main Superuser mode command line:

```
ORACLE# generate-certificate-request acmepacket
Generating Certificate Signing Request. This can take several minutes...
-----BEGIN CERTIFICATE REQUEST-----
MIIDHzCCAoigAwIBAgIIAhMCUACEAHEwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwNDEzMjEzNzQzWhcNMDgwNDEyMjEzNzQzWjBUMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCTUExEzARBgNVBAcTCkJ1cmxpbmd0b24xFDASBgNV
BAoTC0VuZ2luZWVyaW5nMQ0wCwYDVQQDEwRhY211MIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQCXjIeOyFKAUB3rKkKK/+59LT+rlGuW7Lgc1V6+hfTSr0co+ZsQ
bHFUWAA15qXUUBTLJG13QN5VfG96f7gGAbWayfOS9Uymold3JPCUDoGgb2E7m8iu
vtq7gwjSeKNXAw/y7yWy/c04FmUD2U0pZX0CNIR3Mns5OAxQmq0bNYDhawIDAQAB
o4HdMIHaMBEGA1UdEQQKMAiCBnBrdW1hcjAJBgNVHRMEAjAAMB0GA1UdDgQWBBTG
tpodxa6Kmmn04L3Kg62t8BZJHTCBmgYDVR0jBIGSMIGPgBRrRhcU6pR2JYBUbhNU
2qHjVBShtqF0pHIwcDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
ETAPBgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lw
aXQgVGVzdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHmCAQAwDQYJKoZIhvcNAQEFBQAD
gYEAbEs8nUCi+cA2hC/lM49Sitvh8QmpL81KONApsoC4Em24L+DZwz3uInoWjbjJ
QhefcUfteNYkbuMH7LAK0hnDPvW+St4rQGVK6LJhZj7/yeLXmYWIPUY3Ux4OGVrd
2UgV/B2SOqH9Nf+FQ+mNZOlL7EuF4IxSz9/69LuYlXqKsG4=
-----END CERTIFICATE REQUEST-----;
WARNING: Configuration changed, run save-config command.
ORACLE# save-config
Save-config received, processing.
waiting 1200 for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot-activate'
ORACLE# activate-config
Activate-Config received, processing.
waiting 12000 for request to finish
```

```
Add LI flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt: 0
Request to 'ACTIVATE-CONFIG' has finished
Activate Complete
ORACLE#
```

Import a Certificate Using the ACLI

After the Certificate Authority (CA) generates the certificate, you can import it into the Oracle Communications Session Border Controller (SBC) with the **import-certificate** command.

Use the following syntax:

```
ORACLE # import-certificate [try-all|pkcs7|x509] [certificate-record file-name]
```

1. When you use the **import-certificate** command, you can specify whether you want to use PKCS7 or X509v3 format, or try all. In the command line, you enter the command, the format specification, and the name of the certificate record.

   ```
   ORACLE# import-certificate try-all acme
   ```

   The SBC displays the following:

   ```
   Please enter the certificate in the PEM format.
   Terminate the certificate with ";" to exit.......
   -----BEGIN CERTIFICATE-----
   MIIDHzCCAoigAwIBAgIIAhMCUACEAHEwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
   BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
   DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
   dXRob3JpdHkwHhcNMDUwNDEzMjEzNzQzWhcNMDgwNDEyMjEzNzQzWjBUMQswCQYD
   VQQGEwJVUzELMAkGA1UECBMCTUExEzARBgNVBAcTCkJ1cmxpbmd0b24xFDASBgNV
   BAoTC0VuZ2luZWVyaW5nMQ0wCwYDVQQDEwRhY211MIGfMA0GCSqGSIb3DQEBAQUA
   A4GNADCBiQKBgQCXjIeOyFKAUB3rKkKK/+59LT+rlGuW7Lgc1V6+hfTSr0co+ZsQ
   bHFUWAA15qXUUBTLJG13QN5VfG96f7gGAbWayfOS9Uymold3JPCUDoGgb2E7m8iu
   vtq7gwjSeKNXAw/y7yWy/c04FmUD2U0pZX0CNIR3Mns5OAxQmq0bNYDhawIDAQAB
   o4HdMIHaMBEGA1UdEQQKMAiCBnBrdW1hcjAJBgNVHRMEAjAAMB0GA1UdDgQWBBTG
   tpodxa6Kmmn04L3Kg62t8BZJHTCBmgYDVR0jBIGSMIGPgBRrRhcU6pR2JYBUbhNU
   2qHjVBShtqF0pHIwcDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
   ETAPBgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lw
   aXQgVGVzdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHmCAQAwDQYJKoZIhvcNAQEFBQAD
   gYEAbEs8nUCi+cA2hC/lM49Sitvh8QmpL81KONApsoC4Em24L+DZwz3uInoWjbjJ
   QhefcUfteNYkbuMH7LAK0hnDPvW+St4rQGVK6LJhZj7/yeLXmYWIPUY3Ux4OGVrd
   2UgV/B2SOqH9Nf+FQ+mNZOlL7EuF4IxSz9/69LuYlXqKsG4=
   -----END CERTIFICATE-----;
   Certificate imported successfully....
   WARNING: Configuration changed, run "save-config" command.
   ```

2. Save the configuration.

   ```
   ORACLE# save-config
   Save-Config received, processing.
   waiting 1200 for request to finish
   ```

```
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
```

3. Synchronize and activate the configurations.

```
ORACLE# activate-config
Activate-Config received, processing.
waiting 120000 for request to finish
Add LI Flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt:  0
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
ORACLE#
```

4. Reboot the system.

Import a Certificate Using SFTP

1. Copy the certificate to the `/opt` directory of the Oracle Communications Session Border Controller using SFTP.
2. Import the certificate with the **import-certificate** command.

   Use the following syntax:

   ```
   import-certificate [try-all|pkcs7|x509] [certificate name] [filename]
   ```

   Use the value of the **name** parameter from the previously configured **certificate-record** configuration element for the certificate name argument.

   ```
   ORACLE# import-certificate x509 acme certificate.pem
   Certificate imported successfully....
   WARNING: Configuration changed, run "save-config" command.
   ORACLE#
   ```

3. Save the configuration.

   ```
   ORACLE# save-config
   ```

4. Activate the configurations.

   ```
   ORACLE# activate-config
   ```

5. Reboot the system.

Update a Certificate

When you need to renew an expiring certificate on the SBC, you can either create a new certificate-record or you can overwrite the existing certificate-record with the renewed certificate.

Before You Begin

- Send the original Certificate Signing Request to the Certificate Authority (CA) for renewal.

If you want to replace your expiring certificate which a new certificate-record:

1. Follow the procedure in "Configure the Certificate Record" to create a new certificate-record element.
2. Update your tls-profile to point to that new certificate-record.
3. Delete the old certificate-record.
4. Reboot.

If you want to replace your expiring certificate by updating your existing certificate-record:

1. Use the `import-certificate` command to import the new certificate into the certificate-record element that you want to renew.

   For example, if updating a certificate-record element called restless used by an http-server element:

   ```
   ORACLE# import-certificate try-all restless

   IMPORTANT:
            Please enter the certificate in the PEM format.
            Terminate the certificate with ";" to exit.......
   ```

2. Paste your certificate into the ACLI.
3. Type 'y' to overwrite the existing certificate.

   ```
   Warning: A certificate already exist for the cert record "restless".
   Do you want to overwrite it [y/n]?:
   ```

4. Save your configuration.
5. Reboot the SBC.

PKCS #12 Container Import and Export Capability

The SBC supports Public Key Cryptography Standard (PKCS) #12 for bundling a private key with the associated X.509 public key certificate in a file for archiving, importing, and exporting. The SBC does not support bundling all members of the chain of trust.

> **Note:**
>
> The SBC only supports PKCS12 files that are bundled with RSA private keys and their X.509 certificates.

SBC customers often need to use keys and certificates stored in the SBC for Transport Layer Security (TLS) packet analysis and network troubleshooting, or to share with another SBC or other device. The keys and certificates are packaged together and exchanged in the PKCS #12 archive file format.

> **Note:**
>
> The SBC supports this functionality only by way of the ACLI.

Export to a PKCS #12 File

You can export a local entity certificate from the Session Border Controller (SBC) to a PKCS #12 file by way of the ACLI. For the enterprise SBC, you cannot do so from the Web GUI.

> **Note:**
>
> When prompted for password and passphrase, use the ones that you entered in system-config.

- Run the export-certificate command.

  ```
  export-certificate <pkcs#12> <certificate-record-name> [pkcs-12-file-name]
  ```

  where

  - `Certificate-record-name`—the name of the local entity certificate record that you want to export.
  - `pkcs12-file-name`—the name of the target PKCS #12 file. The system creates the export file in the /opt directory. Use either .pfx or .p12 for the file extensions.

  ```
  ORACLE# export-pkcs12 masterca certificate.pfx
  Creating pkcs12 for certificate-record: (masterca)
  PKCS12 Certificate(s) exported successfully....
  ORACLE#
  ```

  This command is supported only when using the RSA key exchange, not when using the Diffie-Hellman key exchange.

Import a PKCS #12 File

You can import a PKCS #12 key and certificate file that was generated elsewhere into the Oracle Communications Session Border Controller (SBC) by way of the ACLI.

Make sure that your PKCS#12 file was generated either with the `-descert` flag or the `-keypbe` and `-certpbe` options. If `rsa.key` is a private key and `cert.crt` is an X.509 certificate, either of the following commands generates a PKCS#12 file.

```
# generate using -descert
openssl pkcs12 -export -in cert.crt -inkey rsa.key -out my_pkcs12.pfx -name
"Test Cert" -descert
# generate using -keypbe and -certpbe options
openssl pkcs12 -export -in cert.crt -inkey rsa.key -out my_pkcs12.pfx -name
"Test Cert" -keypbe PBE-SHA1-3DES -certpbe PBE-SHA1-3DES
```

1. Copy the PKCS#12 file to the /opt directory using SFTP.

2. Run the import-certificate command.

```
import-certificate <pkcs#12> <certificate-record-name> [pkcs-12-file-name]
```

where

- `certificate-record-name`—must be a new name that does not exist as PKCS #12. This is different from other certificate imports, where the certificate record must already exist in the target destination.
- `pkcs12-file-name`—the name of the PKCS #12 file that you want to import.

```
ORACLE# import-certificate pkcs12 newKey2 my2_pkcs12.pfx
The specified certificate-record: (newKey2) does not exist.
Creating one...
Enter Import Password:
Importing ee: newKey2
Certificate(s) imported successfully....

-----------------------------------------------
WARNING:
Configuration changed, run 'save-config' and
'activate-config' commands to commit the changes.
-----------------------------------------------
ORACLE#
```

> **Note:**
>
> 512-bit keys are not supported.

View Certificates

You can view either a brief version or a detailed version of your certificates.

- Brief Version
- Detailed Version

Brief Version

Obtaining the brief version uses this syntax, and will appear like the following example:

```
ORACLE# show security certificates brief acmepacket
certificate-record: restless
Certificate:
     Data:
         Version: 3 (0x2)
             Serial Number: 96 (0x60)
             Issuer:
                 C=US
                 ST=MA
                 L=Burlington
                 O=Engineering
                 CN=Acme Packet MA
                 emailAddress=bob@oracle.com
```

```
            Subject:
                C=US
                ST=MA
                O=Eng
                CN=Acme
ORACLE#
```

Detailed Version

Obtaining the detailed version uses this syntax, and will appear like the following example:

```
ORACLE# show security certificates detail restless
certificate-record: restless
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 96 (0x60)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer:
            C=US
            ST=MA
            L=Burlington
            O=Engineering
            CN=Acme Packet MA
            emailAddress=bob@oracle.com
        Validity
            Not Before: Feb 16 08:45:19 2024 GMT
            Not After : Feb 17 08:45:19 2024 GMT
        Subject:
            C=US
            ST=MA
            O=Eng
            CN=Acme
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                7A:0D:7B:49:48:45:86:1F:4E:1E:B8:13:D9:15:D0:CE:95:DC:53:5C
            X509v3 Authority Key Identifier:

keyid:D6:56:81:C0:7B:07:F5:3F:A2:18:A6:F5:70:8D:34:92:1B:73:EC:4C
                DirName:/C=US/ST=MA/L=Burlington/O=Engineering/CN=Acme Packet
MA/emailAddress=bob@oracle.com
                serial:DA:76:D0:86:25:E5:AB:E4
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
```

Configure a TLS Profile

The TLS profile configuration contains the information required to run SIP over TLS.

- Obtain the necessary certificates.
- Confirm that the system displays the Superuser mode.

When the Oracle Communications Session Border Controller (SBC) negotiates with TLS, it starts with the highest TLS version and works its way down until it finds a compatible version and cipher that works for the other side.

1. Access the **tls-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tls-profile
ORACLE(tls-profile)#
```

2. **name**—Enter the name of the TLS profile. Required.
3. **end-entity-certificate**—Enter the name of the entity certification record.
4. **trusted-ca-certificates**—Enter the names of the trusted CA certificate records.
5. **cipher-list**—Use either the default **DEFAULT**, or enter a list of ciphers that you want to support. For a complete list of supported ciphers, see the *Oracle Communications Session Border Controller Release Notes*.
6. **verify-depth**—Specify the maximum depth of the certificate chain to verify. Default: 10. Valid range: 0-10.
7. **mutual-authenticate**—Define whether or not you want the SBC to mutually authenticate the client. Valid values: enabled | disabled. Default: disabled.
8. **tls-version**—Enter the TLS version that you want to use with this TLS profile. Valid values are:
   - tlsv1.3 (default)
   - tlsv1.2
   - compatibility—When the Oracle Communications Session Border Controller negotiates on TLS, it starts with the highest TLS version and works its way down until it finds a compatible version and cipher that works for the other side.

> **Note:**
>
> The **sslmin** option in **security-config** specifies the lowest TLS version allowed when **tls-version** is set to **compatibility**.

> **Note:**
>
> As per the Curl functionality, Curl always publishes all the TLS versions from the configured TLS version to the maximum TLS version supported by Curl. It is applicable to all the modules where TLS is used with Curl.
> For an example, If the SBC is configured with TLSv1.2, then the SBC as a client will publish TLSv1.2 to the maximum TLS version supported by Curl in the client hello message.

9. Type **done** to save your configuration.

Applying a TLS Profile

To apply the TLS profile, you need to specify it for the SIP interface with which it will be used. You must take this step from within the SIP interface configuration.

1. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

2. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

3. Select the existing SIP interface to which you want to apply the TLS profile. If you do not know the same of the profile, press Enter again after you use the select command to see a list of all SIP interfaces. Type in the number corresponding to the SIP interface you want to select, and press Enter. You will then be modifying that SIP interface.

   ```
   ORACLE(sip-interface)# select
   ```

4. **Type sip-ports and** Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-interface)# sip-ports
   ORACLE(sip-port)#
   ```

5. **transport-protocol**—Change the transport protocol to **TLS**.

   ```
   ORACLE(sip-interface)# transport-protocol tls
   ```

6. **tls-profile**—Enter the name of the TLS profile you want applied. This is the same value you enter for the name parameter in the TLS profile configuration. This profile will be applied when the transport protocol is TLS.

   ```
   ORACLE(sip-interface)# tls-profile acmepacket
   ```

7. Save your updated SIP interface configuration.

Notifications for Certificate Expiration

Traps

When a security certificate is installed locally on the Oracle Communications Session Border Controller, you can poll the expiration of the certificate using the **apSecurityCertificateTable**.

You can configure the SBC to generate the apSecurityCertExpiredNotification trap once a certificate has expired. The number of minutes between notifications sent is configured in the **security-config** parameter **local-cert-exp-trap-int.**

To send a warning of expiration, you can set the **security-config** parameter **local-cert-exp-warn-period** to the number of days before the locally installed certificate expires in which you would like a warning. The number of minutes between notifications sent is configured in the**security-config** parameter **local-cert-exp-trap-int**.

Alarms

The SBC also generates an alarm when the certificate of a **tls-profile** is about to expire or has expired. The value of **local-cert-exp-warn-period** determines the number of days before a certificate expires in which the SBC generates a warning alarm.

When the certificate is about to expire:

```
ORACLE# display-alarms
1 alarms to show
ID      Task    Severity        First Occurred          Last Occurred
327731  3386    6       2019-01-29 21:47:55     2019-01-29 21:47:55
Count   Description
1       Certificate: tempCert expiring on Jan 30 20:58:29 2019 GMT,

done
ORACLE#
```

When the certificate has expired:

```
ORACLE# display-alarms
1 alarms to show
ID      Task    Severity        First Occurred          Last Occurred
327730  3386    6       2019-02-01 16:20:45     2019-02-01 16:20:45
Count   Description
1       Certificate: tempCert expired on Jan 30 20:58:29 2019 GMT,

done
ORACLE#
```

Configuring Notifications for Certificate Expiration

To configure the Oracle Communications Session Border Controller to generate traps and alarms when a certificate has or is about to expire:

1. Navigate to the **security-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security#
   ORACLE(security)# security-config
   ORACLE(security-config)#
   ```

2. Set the **local-cert-exp-warn-period** parameter to the number of days before the locally installed certificate expires in order to receive a warning trap and alarm.

   A value of 0 disables the trap and alarm.

   ```
   ORACLE(security-config)# local-cert-exp-warn-period 60
   ORACLE(security-config)#
   ```

> **Note:**
>
> The default value is 30. Consider this configuration carefully if you are using it in conjunction with the CA trust store feature.

3. Set the **local-cert-trap-int** parameter to the number of minutes between notifications sent once a certificate has expired. This value is also used for notification interval when in the pre-expiration warning period.

   A value of 0 disables the warning trap and alarm.

   ```
   ORACLE(security-config)# local-cert-exp-trap-int 15
   ORACLE(security-config)#
   ```

4. Use **done**, **exit**, and **verify-config** to complete required configuration.

Denial of Service for TLS

This section explains the DoS for TLS feature. With this feature, the Oracle Communications Session Border Controller can provide protection from TCP/TLS message flood by limiting the number of connections from an end point and by limiting the number of simultaneous TCP/TLS connections to a SIP interface.

The Oracle Communications Session Border Controller protects against a flood of invalid TLS messages and against end points establishing TCP/TLS connections or doing an initial registration without then sending any messages. The Oracle Communications Session Border Controller protects against:

- Too many simultaneous TLS connections being requested by a single IP address by limiting the number of TLS connections from a single IP address. There is a maximum simultaneous number of TCP/TLS connections a SIP interface will allow from a single IP address.
- Too many simultaneous TLS connections being requested by limiting the maximum number of connections for a SIP interface. There is a maximum number of simultaneous TCP/TLS connections a SIP interface will allow in aggregate from all IP addresses served by that signaling interface.
- End points establishing TCP/TLS connections without then sending any messages (application layer messages post TLS handshake complete). Triggered by inactivity as measured by lack of any message from this peer.
- End points doing an initial registration without then sending any messages.
  This timer could be used by the administrator to detect errors with the SIP configuration. It is expected that whenever an end point establishes a TCP/TLS connection, the end point will keep the connection active by sending messages with REGISTER or by using the NAT interval configuration. Whenever a connection is torn down because of inactivity, a log at the level ERROR is generated.)
- Malformed packets by counting and limiting the maximum number of malformed packets. Whenever an invalid TLS message is received, the internal counter corresponding to invalid-signal-threshold is incremented. When the invalid signal threshold reaches the configured value, the end point will be denied for the configured deny period. (Also requires configuration of the tolerance window in media manager.)
- The max-incoming-conns parameter is well under the maximum number of TLS connections supported by the system. You can set this parameter to it's maximum value of 20000. If you need more than 20000 TLS connections available on this SIP interface, you

must set max-incoming-conns to 0 which allows up to the system maximum number of TLS connections, taken on a first come first served basis, on this SIP interface.

DoS for TLS Configuration

You configure the SIP interface and the realm to support DoS for TLS.

DoS protection for TLS Connections on the SIP Interface Configuration

To configure the DoS protection for TCP/TLS connections on a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. **max-incoming-conns**—Enter the maximum number of simultaneous TCP/TLS connections for this SIP interface. The default value is zero (0) which disables any limit to the number of simultaneous TCP/TLS connections on this SIP interface. The valid range is:
   - Minimum—0
   - Maximum—20000
5. **per-src-ip-max-incoming-conns**—Enter the maximum number of connections allowed from an end point.The default value is zero (**0**). The default disables the parameter. The valid range is:
   - Minimum—0
   - Maximum—20000

   > **Note:**
   >
   > To make this parameter effective, you need to set the realm's access-control-trust-level to low or medium.

6. **inactive-conn-timeout**—Enter the time in seconds you want a connection from an endpoint discontinued. This provides protection from end points doing an initial registration without sending any messages. The default value is zero (**0**). The default disables the parameter. The valid range is:
   - Minimum—0
   - Maximum—999999999

7. Save and activate your configuration.

Configuring the SIP Configuration

To configure the SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# session-router
```

3. Type **sip-config** and press Enter. The system prompt changes.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

From this point, you can configure SIP configuration parameters. To view all sip-config parameters, enter a **?** at the system prompt.

4. **inactive-dynamic-conn**—Enter the time in seconds after which the Oracle Communications Session Border Controller tears down inactive dynamic TCP connections. Inactive is defined as not transporting any traffic. This protects against endpoints establishing TCP/TLS connections and then not sending messages. The default value is 32. The valid range is:
   - Minimum—0
   - Maximum—999999999

   > **Note:**
   >
   > Setting this parameter to 0 disables this parameter.

   Because the Oracle Communications Session Border Controller first establishes a TCP connection, then the TLS connection it waits twice the value entered here after the initiation of a TLS connection before tearing down the connection.

   After an endpoint establishes a TCP/TLS connection, it is supposed to keep the connection active by sending messages or by using the NAT interval configuration. Whenever a connection is torn down because of inactivity, a log at the level ERROR is generated.

Configuring the Realm

To configure the realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the media-related configurations.

```
ORACLE(configure)# media-manager
```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

   From this point, you can configure realm parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

4. **deny-period**—Indicate the time period in seconds after which the entry for this host is removed from the deny list. The default value is **30**. The valid range is:
   - Minimum—0
   - Maximum—4294967295
5. **invalid-signal-threshold**— Enter the number of invalid TLS signaling messages that trigger host demotion. The value you enter here is only valid when the trust level is low or medium. Available values are:
   - Minimum—Zero (0) is disabled.
   - Maximum—999999999

     If the number of invalid messages exceeds this value based on the tolerance window parameter, configured in the media manager, the host is demoted.

     The tolerance window default is 30 seconds. Bear in mind, however, that the system uses the same calculation it uses for specifying "recent" statistics in show commands to determine when the number of signaling messages exceeds this threshold. This calculation specifies a consistent start time for each time period to compensate for the fact that the event time, such as a user running a show command, almost never falls on a time-period's border. This provides more consistent periods of time for measuring event counts.

     The result is that this invalid signal count increments for two tolerance windows, 60 seconds by default, within which the system monitors whether or not to demote the host. The signal count for the current tolerance window is always added to the signal count of the previous tolerance window and compared against your setting.
6. **access-control-trust-level**—Set the trust level for the host within the realm. The default value is **none**. The valid values are:
   - **none**—Host is always untrusted. It is never promoted to the trusted list or demoted to the deny list.
   - **low**—Host can be promoted to the trusted list or demoted to the deny list.
   - **medium**—Host can be promoted to the trusted list but is only demoted to untrusted. It is never added to the deny list.
   - **high**—Host is always trusted.
7. Save and activate your configuration.

TLS Session Caching

Transport Layer Security (TLS) session caching allows the Oracle Communications Session Border Controller to cache key information for TLS connections, and to set the length of time that the information is cached.

When TLS session caching is not enabled, the Oracle Communications Session Border Controller and a TLS client perform the handshake portion of the authentication sequence in

which they exchange a shared secret and encryption keys are generated. One result of the successful handshake is the creation of a unique session identifier. When an established TLS connection is torn down and the client wants to reinstate it, this entire process is repeated. Because the process is resource-intensive, you can enable TLS session caching to avoid repeating the handshake process for previously authenticated clients to preserve valuable Oracle Communications Session Border Controller resources.

When TLS session caching is enabled on the Oracle Communications Session Border Controller, a previously authenticated client can request re-connection using the unique session identifier from the previous session. The Oracle Communications Session Border Controller checks its cache, finds the session identifier, and reinstates the client. This process reduces the handshake to three messages, which preserves system resources.

If the client offers an invalid session identifier, for example, one that the Oracle Communications Session Border Controller has never seen or one that has been deleted from its cache, the system does not allow the re-connection. The system negotiates the connection as a new connection.

TLS Session Caching Configuration

TLS session caching is global for all TLS functions on your Oracle Communications Session Border Controller. A new global TLS configuration (**tls-global**) has been added to the system for this purpose.

To enable global TLS session caching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **security** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **tls-global** and press Enter.

   ```
   ORACLE(security)# tls-global
   ORACLE(tls-global)#
   ```

4. **session-caching**—Set the state for TLS session caching to **enabled** if you want to turn this feature on. The default value is **disabled**, meaning the SBC does not send new session tickets. The valid values are:

   - enabled | disabled

   > **Note:**
   >
   > This parameter is not RTC supported.

5. **session-cache-timeout**—Enter the time in hours that you want the SBC to cache unique session identifiers so that previously authenticated clients can reconnect. The default value is **12**. A value of **0** disables this parameter. The valid range is:

   - Minimum—0
   - Maximum—24

If you set this parameter to 0, then cache entries will never age (and not be deleted from the cache unless you use the **clear-cache tls** command to delete all entries from the TLS cache). RFC 2246, *The TLS Protocol Version 1.0*, recommends that you set this parameter at the maximum, 24.

TLS Endpoint Certificate Data Caching

To provide a higher level of security for unified messaging (UM), the Oracle Communications Session Border Controller allows you configure enforcement profiles to cache data from TLS certificates. During the authentication process, the system caches the data so it can use that data in subsequent SIP message processing. Thus the Oracle Communications Session Border Controller can:

- Add custom SIP header populated with information from TLS certificates—When the Oracle Communications Session Border Controller receives an INVITE from a GW, it can write proprietary headers into the SIP message. It uses the certificate information the GW provided during the TLS authentication process with the Oracle Communications Session Border Controller to do so.
- Compare the host of the Request-URI with information from TLS certificates—When an INVITE is destined for the unified messaging server, the Oracle Communications Session Border Controller checks the domain of the Request-URI it has generated prior to HMR application. It does so to verify that the Request-URI matches the domain information the UM server provided during the TLS authentication process with the Oracle Communications Session Border Controller.

TLS endpoint certificate data caching can only applies to call-creating SIP INVITEs. The Oracle Communications Session Border Controller looks to the following configurations, in order, to apply an enforcement profile: session agent, realm, and SIP interface associated with the INVITE. As a final step, it checks the SIP profile for enforcement profile association.

Inserting Customized SIP Headers in an Outgoing INVITE

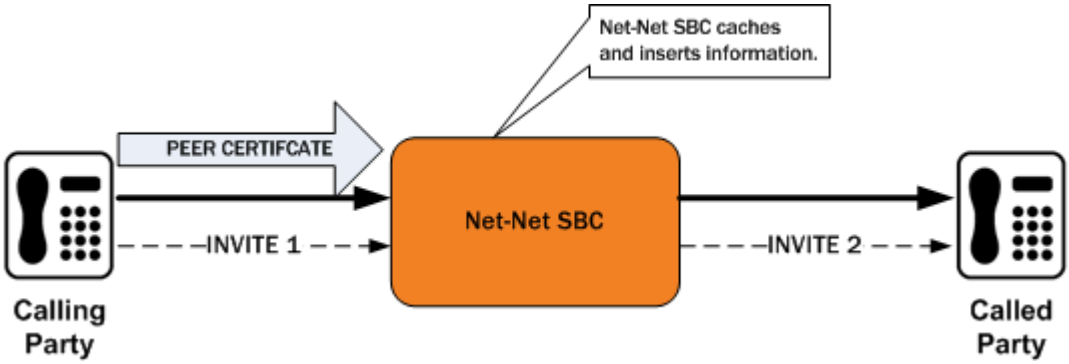
When the Oracle Communications Session Border Controller establishes a new TLS connection, it caches the following peer certificate attributes:

- Certificate Subject Name
- Certificate Subject Alternative Name (only DNS)

The Oracle Communications Session Border Controller constructs a customized P-Certificate-Subject-Common-Name SIP header and inserts the header into the outgoing INVITE with the Certificate Subject Name. The Oracle Communications Session Border Controller also constructs and inserts in the outgoing INVITE one or more P-Certificate-Subject-Alternative-Name SIP headers.

If you enable this capability and the incoming INVITE already has P-Certificate-Subject-Common-Name and P-Certificate-Subject-Alternative-Name headers, the Oracle Communications Session Border Controller strips them before inserting the new customized ones. It does so to avoid the risk of any attempt to spoof the headers and thereby gain unauthorized access to the UM server.

The following diagram shows a scenario where the calling party establishes a TLS connection with the Oracle Communications Session Border Controller. Because mutual authentication is enabled, the Oracle Communications Session Border Controller receives the peer certificate and caches required information from it. This information is inserted in the outgoing INVITE.



The peer certificate from the calling party during the TLS handshake with the Oracle Communications Session Border Controller looks like the following example.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 9 (0x9)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=MA, L=Woburn, O=Smith Securities, OU=Certificate
Authority Dept, CN=Smith Certificate Authority/emailAddress=Smith@CA.com
        Validity
            Not Before: Dec 10 21:14:56 2009 GMT
            Not After : Jul 11 21:14:56 2019 GMT
        Subject: C=US, ST=MA, L=Burlington, O=Acme Packet, OU=Certificate
Authority Dept, CN=*.acme.com/emailAddress=ph1Client@acme.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
        X509v3 extensions:
            X509v3 Basic Constraints:
            CA:FALSE
            X509v3 Issuer Alternative Name:
            email:Smith@CA.com
            X509v3 Subject Alternative Name:
            DNS:gw1.acme.com, DNS:gw3.ano.com, DNS:gw2.some.com
            X509v3 Key Usage: critical
            Digital Signature, Key Encipherment
    Signature Algorithm: sha256WithRSAEncryption
```

The outgoing SIP INVITE (INVITE 2 in the diagram) looks like the following sample. Bold text shows where the Oracle Communications Session Border Controller uses information from the certificate.

```
INVITE sip:222222@acme.com:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.27.113:5060;branch=z9hG4bK4jmg29cmm810cg7smmrn85o4q7
From: 111111 <sip:111111@acme.com>;tag=_ph1_tag
To: 222222 <sip:222222@acme.com>
Call-ID: _1-2_call_id-10147@acme.com-1-
CSeq: 1 INVITE
Contact: <sip:111111@172.16.27.113:5060;transport=udp>
P-Certificate-Subject-Common-Name: *.acme.com
P-Certificate-Subject-Alternative-Name: gw1.acme.com
P-Certificate-Subject-Alternative-Name: gw3.ano.com
P-Certificate-Subject-Alternative-Name: gw2.some.com
```

```
Max-Forwards: 69
Subject: TBD
Content-Type: application/sdp
Content-Length: 138
Route: <sip:222222@172.16.27.188:5060;lr>
v=0
o=user1 53655765 2353687637 IN IP4 172.16.27.113
s=-
c=IN IP4 172.16.27.113
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Validating the Request-URI Based on Certificate Information

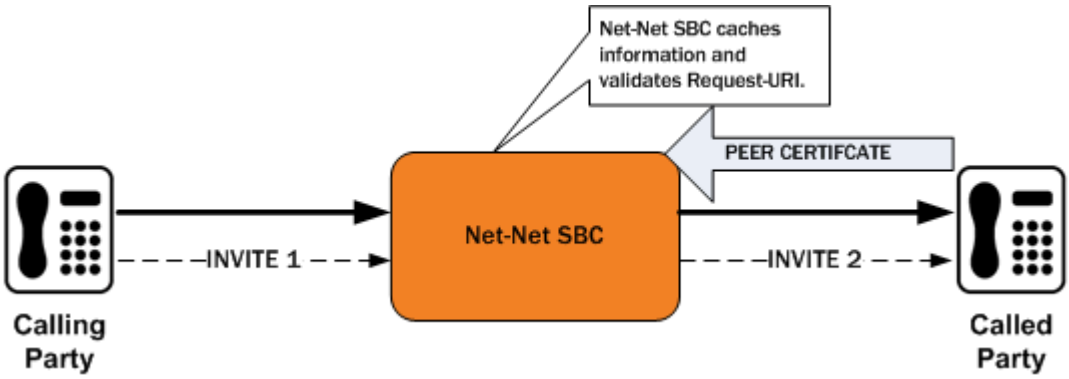
When you configure the Oracle Communications Session Border Controller to cache TLS certificate information to validate Request-URIs, it stores the Certificate Subject Name and Certificate Subject Alternative Name (only DNS) it learns from peer certificate attributes. It then takes these actions:

- Extracts the host from the Request-URI of the outgoing INVITE
- Compares this host (exact or wildcard match) with the Certificate Common Name or Certificate Subject Alternative name of the certificate it has received
- Sends out an INVITE if the Certificate Common Name or Certificate Subject Alternative name match; Sends a 403 Forbidden rejection to the endpoint from it received the INVITE if there is no match

Wildcard matching applies only to the prefix of the Request-URI:

```
*.acme.com
    *.*.acmepacket.com
```

This diagram shows a peering scenario where the Oracle Communications Session Border Controller receives an INVITE from the calling party, which it processes and prepares to send out INVITE 2. After establishing a TLS connection with the called party and caching the required information, the Oracle Communications Session Border Controller validates the Request-URI. Once validation occurs, the Oracle Communications Session Border Controller sends INVITE 2.



The peer certificate from the called party during the TLS handshake with the Oracle Communications Session Border Controller would look like this. Relevant information in the sample appears in **bold** font.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 9 (0x9)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=MA, L=Woburn, O=Smith Securities, OU=Certificate
Authority Dept, CN=Smith Certificate Authority/emailAddress=amith@CA.com
        Validity
            Not Before: Dec 10 21:14:56 2009 GMT
            Not After : Jul 11 21:14:56 2019 GMT
        Subject: C=US, ST=MA, L=Woburn, O=Acme Packet, OU=Certificate
Authority Dept, CN=*.acme.com/emailAddress=ph2Server@acme.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
        X509v3 extensions:
            X509v3 Basic Constraints:
            CA:FALSE
            X509v3 Issuer Alternative Name:
            email:Smith@CA.com
            X509v3 Subject Alternative Name:
            DNS:gw1.acme.com, DNS:*.ano.com, DNS:*.some.com
            X509v3 Key Usage: critical
            Digital Signature, Key Encipherment
    Signature Algorithm: sha256WithRSAEncryption
```

The outgoing SIP INVITE (INVITE 2 in the diagram) would then look like the sample below. The INVITE is sent because **smith.acme.com** matches the common name ***.acme.com**. Other valid SIP Request-URIs would be:

```
222222@gw1.acme.com
222222@smith.ano.com
222222@amith.some.com
```

You can see where the system uses information from the certificate; the text is **bold**.

```
INVITE sip:222222@smith.acme.com:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.27.113:5060;branch=z9hG4bK4jmg29cmm8l0cg7smmrn85o4q7
From: 111111 <sip:111111@acme.com>;tag=_ph1_tag
To: 222222 <sip:222222@acme.com>
Call-ID: _1-2_call_id-10147@acme.com-1-
CSeq: 1 INVITE
Contact: <sip:111111@172.16.27.113:5060;transport=udp>
Max-Forwards: 69
Subject: TBD
Content-Type: application/sdp
Content-Length: 138
Route: <sip:222222@172.16.27.188:5060;lr>
v=0
o=user1 53655765 2353687637 IN IP4 172.16.27.113
s=-
```

```
c=IN IP4 172.16.27.113
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

TLS Endpoint Certificate Data Caching Configuration

To configure SIP endpoint certificate data caching for an enforcement profile:

1. Access the **enforcement-profile** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# enforcement-profile
   ORACLE(enforcement-profile)#
   ```

2. Select the **enforcement-profile** object to edit.

   ```
   ORACLE(enforcement-profile)# select
   <name>:

   ORACLE(enforcement-profile)#
   ```

3. **add-certificate-info**—Enter a list of one or more certificate attribute names to enable TLS certificate information caching and insertion of cached certificate information into a customized SIP INVITEs. This parameter is empty by default.

   If you want to list more than one value, enclose the value in quotation marks (“ “) and separate the values with Spaces.

   ```
   ORACLE(enforcement-profile)# add-certificate-info "sub-common-name sub-alt-name-DNS"
   ```

4. **certificate-ruri-check**—Change this parameter from **disabled**, its default, to **enabled** if you want your Oracle Communications Session Border Controller to cache TLS certificate information and use it to validate Request-URIs. Enabling this parameter also means the Oracle Communications Session Border Controller will use the cached TLS certificate information in a customized SIP INVITE.
5. Type **done** to save your configuration.

Untrusted Connection Timeout for TCP and TLS

You can configure the Oracle Communications Session Border Controller for protection against starvation attacks for socket-based transport (TCP or TLS) for SIP access applications. During such an occurrence, the attacker would open a large number of TCP/TLS connections on the Oracle Communications Session Border Controller and then keep those connections open using SIP messages sent periodically. These SIP messages act as keepalives, and they keep sockets open and consume valuable resources.

Using its ability to promote endpoints to a trusted status, the Oracle Communications Session Border Controller now closes TCP/TLS connections for endpoints that do not enter the trusted state within the period of time set for the untrusted connection timeout. The attacking client is thus no longer able to keep connections alive by sending invalid messages.

This feature works by setting a value for the connection timeout, which the Oracle Communications Session Border Controller checks whenever a new SIP service socket for TCP or TLS is requested. If the timer's value is greater than zero, then the Oracle Communications Session Border Controller starts it. If the timer expires, then the Oracle Communications Session Border Controller closes the connection. However, if the endpoint is promoted to the trusted state, then the Oracle Communications Session Border Controller will cancel the timer.

Caveats

This connection timeout is intended for access applications only, where one socket is opened per-endpoint. This means that the timeout is not intended for using in peering applications; if this feature were enabled for peering, a single malicious SIP endpoint might cause the connection to be torn down unpredictably for all calls.

Untrusted Connection Timeout Configuration for TCP and TLS

The untrusted connection timer for TCP and TLS is set per SIP interface.

To set the untrusted connection timer for TCP and TLS:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   If you are adding support for this feature to a pre-existing SIP configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **untrusted-conn-timeout**—Enter the time in seconds that you want the Oracle Communications Session Border Controller to keep TCP and TLS connections open for untrusted endpoints. The default value is **0**, which will not start the timer. The valid range is:
   - Minimum—0
   - Maximum—999999999
5. Save and activate your configuration.

Securing Communications with TLS

You can use the Transport Layer Security (TLS) protocol to secure the communications link between the Oracle Communications Session Border Controller (SBC) and the Oracle Communications Session Delivery Manager. Note that the systems use the Acme Control Protocol (ACP) for this messaging.

To configure the SBC to use TLS for this ACP messaging:

1. Configure a TLS profile. The tls-profile object is located under security, where you add certificates, select cipher lists, and specify the TLS version for each profile.
2. Configure system-config element's `acp-tls-profile` parameter to specify this TLS profile.

The `acp-tls-profile` parameter is empty by default, which means that ACP over TLS is disabled. When ACP over TLS is disabled, the Session Delivery Manager establishes a TCP connection with the SBC. When the `acp-tls-profile` parameter specifies a valid TLS profile, the SBC negotiates a TLS connection with Session Delivery Manager.

You must reboot SBC after configuring ACP over TLS.

> **Note:**
>
> This feature requires Session Delivery Manager version 8.1 and above.

Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) is defined in RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. The protocol enables users to determine the revocation state of a specific certificate, and may provide a more efficient source of revocation information than is possible with Certificate Revocation Lists (CRL).

The protocol specifies the data exchanged between an OCSP client (for example, the Oracle Communications Session Border Controller) and an OCSP responder, the Certification Authority (CA), or its delegate, that issued the target certificate. An OCSP client issues a request to an OCSP responder and suspends acceptance of the certificate in question until the responder replies with a certificate status.

Certificate status is reported as

- good
- revoked
- unknown

good indicates a positive response to the status inquiry. At a minimum, this positive response indicates that the certificate is not revoked, but does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval.

revoked indicates that the certificate has been revoked, either permanently or temporarily.

unknown indicates that the responder cannot identify the certificate.

Caveats

OCSP is currently supported only on TLS interfaces; it is not currently supported for use with IKEv1 and IKEv2.

Online Certificate Status Protocol Configuration

OCSP configuration consists of:

- Configuring one or more certificate status profiles; each profile contains information needed to contact a specific OCSP responder.
- Enabling certificate revocation checking by assigning a certificate status profile to a previously configured TLS profile.

1. Access the **cert-status-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# cert-status-profile
ORACLE(cert-status-profile)#
```

2. **name**—Identify this cert-status-profile instance.

   Each profile instance provides configuration data for a specific OCSP responder.

3. **ip-address**—Specify the IPv4 address of the OCSP responder.

4. **port**—Specify the destination port.

   The default port number is 80.

5. **realm-id** —Specify the realm used to transmit OCSP requests.

   The default is the wancom0 interface.

   If the IP address resolved by DNS is not accessible from the identified realm, the SBC sends the OCSP request out on the wancom0 interface. If the address is not routeable, create a host-route configuration element.

6. **requester-cert**—Specify the requester certificate only if OCSP requests are signed; ignore this parameter if requests are not signed.

   RFC 2560 does not require signed requests; however, local or CA policies can mandate digital signatures.

7. **responder-cert**—Identify the certificate used to validate OCSP responses.

   This value is the public key of the OCSP responder.
   RFC 2560 requires that all OCSP responders digitally sign OCSP responses, and that OCSP clients validate incoming signatures.

   Provide the name of the certificate configuration element that contains the certificate used to validate the signed OCSP response.

8. **retry-count**—Specify the maximum number of times to retry an OCSP responder in the event of connection failure.

   If the retry counter is exceeded, the OCSP requester either contacts another responder (if multiple responders have been configured within this cert-status-profile) and quarantines the unavailable responder for the period defined in the **dead-time** parameter.

   In the absence of an explicitly configured value (an integer within the range 0 through 10), the default of 1 is used.

```
ORACLE(cert-status-profile)# retry-count 2
ORACLE(cert-status-profile)#
```

9. **dead-time**—Specify the quarantine period imposed on an unavailable OCSP responder.

   The range is 0 through 3600 seconds, and the default is 0.

   Customers using a single OCSP responder should retain the default value or specify a brief quarantine period to prevent lengthy service outages.

10. Retain the default values for **type** and **trans-protocol**.
11. Use **done**, **exit**, and **verify-config** to complete configuration of this cert-status-profile instance.
12. Repeat Steps 1 through 11 to configure additional certificate status profiles.

Enable Certificate Status Checking

After configuring certificate status profiles, enable checking certificates.

1. Access the **tls-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tls-profile
ORACLE(tls-profile)#
```

2. **cert-status-check**—Enable OCSP in conjunction with an existing TLS profile.
3. **cert-status-profile-list**—Assign one or more cert-status-profiles to the current TLS profile.

   Each assigned cert-status-profile provides the information needed to access a single OCSP responder.

   Use quotation marks to assign multiple OCSP responders. The following sequence assigns three cert-status-profiles (VerisignClass3Designate, Verisign-1, and Thawte-1) to the TLS-1 profile.
4. Use **done**, **exit**, and **verify-config** to complete configuration.

   Sample Configuration:

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# cert-status-profile
ORACLE(cert-status-profile)# name VerisignClass3Designate
ORACLE(cert-status-profile)# ip-address 192.168.7.100
ORACLE(cert-status-profile)# responder-cert VerisignClass3ValidateOCSP
ORACLE(cert-status-profile)# done
ORACLE(cert-status-profile)# exit
...
...
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tls-profile
ORACLE(tls-profile)# select
<name>:
1. TLS-1
2. TLS-2
3. TLS-3
selection: 1
ORACLE(tls-profile)# cert-status-check enabled
ORACLE(cert-status-profile)# cert-status-profile-list
"VerisignClass3Designate Verisign-1 Thawte-1"
ORACLE(cert-status-profile)# done
ORACLE(cert-status-profile)# exit
```

The **tls-profile** configuration element is not RTC supported for MSRP Online Certificate Status Protocol. To support MSRP OCSP, reboot the SBC.

Unreachable OCSP Responder

With OCSP enabled, the client implementation running on the SBC supports message exchange between the SBC and an OCSP responder as specified in RFC 2560, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. The SBC contacts the OCSP responder whenever a remote client attempts to establish an SSL/TLS connection with the SBC. The SBC sends a request to the OCSP responder to check the current status of the certificate presented by the remote client. The SBC suspends processing of the SSL/TLS connection request pending receipt of the OCSP response. In previous releases (prior to Version S-CX6.3.0), a good OCSP response resulted in the establishment of a secure SSL/TLS connection. A revoked or unknown OCSP response, or the failure to reach an OCSP responder, resulted in the rejection of the connection attempt.

This behavior, which adheres to the requirements of RFC 2560, conflicts with the requirements of Section 5.4.6.2.1.6.4.a.i of UCR 2008 which requires an OCSP client to attempt authentication of remote clients in the event of an unreachable OCSP responder.

The attribute **ignore-dead-responder** in the TLS profile configuration element provides compliance with DISA/DoD requirements specifying OCSP client operations when faced with unreachable OCSP responders. By default, the attribute is disabled meaning that all client connections will be disallowed in the event of unreachable OCSP responders.

In DISA/DoD environments **ignore-dead-responder** should be enabled, allowing local certificate-based authentication by the SBC in the event of unreachable OCSP responders. Successful authentication is achieved if the certificate presented by the remote client was signed by a Certificate Authority (CA) referenced by the **trusted-ca-certificates** attribute. If the local authentication succeeds, the secure TLS/SSL connection is established; otherwise the connection is rejected.

Unreachable OCSP Responder Configuration

The following sample configuration implements DISA/DoD-compliant client behavior in the event of an unreachable OCSP responder.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security#
ACMEPACKET(security)# tls-profile
ACMEPACKET(security)# show
tls-profile
        name                        DoD
        end-entity-certificate      sylarCert-2048
        trusted-ca-certificates     dod1 dod2 disaA disaB IBM1
        cipher-list                 all
        verify-depth                10
        mutual-authenticate         disabled
        tls-version                 tlsv13
        cert-status-check           enabled
        cert-status-profile-list    DoD
        ignore-dead-responder       enabled
        ...
        ...
ACMEPACKET(tls-profile)#
```

OCSP Status Monitoring

OCSP monitoring is provided to track the reachability of individual OCSP responders, and, in topologies containing multiple OCSP responders, the overall availability of the OCSP service.

If monitoring is enabled for individual OCSP responders, reachability is monitored by observing responder transactions.

Initially, all OCSP responders are considered reachable. If a previously reachable OCSP responder fails to respond to a certificate status request, the SBC marks the OCSP responder as unreachable, and generates an SNMP trap and log entry indicating that status. If a previously unreachable OCSP responder responds to a certificate status request, the SBC returns the OCSP responder to the reachable status, and generates an SNMP trap and log entry indicating that status change.

Use the following procedure to enable monitoring of individual OCSP responders.

1. Navigate to the new security-config configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security#
ORACLE(security)# security-config
ORACLE(security-config)#
```

2. Enable monitoring of individual OCSP responders by setting the **ocsr-monitoring-traps** attribute to **enabled**; this attribute is disabled by default.

```
ORACLE(security-config)# ocsr-monitoring-traps enabled
ORACLE(security-config)#
```

3. Use **done**, **exit**, and **verify-config** to complete required configuration.

   Reachability status of individual OCSP responders is aggregated to monitor the overall availability of the OCSP service. Using the procedure explained above, the Oracle Communications Session Border Controller maintains a count of all OCSP responders, and of all reachable OCSP responders.

   - If all OCSP responders are reachable, the SBC generates a trap and log entry noting this optimal state.
   - If all OCSP responders are unreachable, the SBC generates a trap and log entry noting this erroneous state.
   - When the SBC transitions from either of the two states described above (in the optimal state, when an OCSP responder becomes unreachable; in the erroneous state, when an unreachable OCSP responder becomes reachable), the SBC generates a trap and log entry indicating that an unspecified number of OCSP responders are reachable.

   Monitoring of the OCSP service availability is a by-product of enabling SNMP; no further configuration is required.

OCSP Access via FQDN

OCSP responders can be accessed via IPv4 addresses and port numbers as well as FQDNs. Since multiple public key infrastructure (PKI) elements capable of supporting OCSP requests can exist within a DISA/DoD environment, the Domain Name Service (DNS) lookup that resolves the FQDN can result in more than one OCSP responder IP address being returned to the SBC in its role of OCSP client. When processing a lookup that contains more than one IP

address, the SBC uses a
round-robin algorithm to select from the list of OCSP responder addresses.

OCSP access via FQDN is available on all media interfaces and on the wancom0 administrative interface. Note that support for FQDN-based access requires the configuration of DNS support.

If the **realm** attribute is configured in the certificate-status-profile configuration element, the required DNS query is issued on the corresponding network interface. This model requires configuration of the **dns-ip-primary** attribute, and optionally the **dns-ip-backup1** and **dns-ip-backup2** attributes for the realm's network interface.

If the **realm** attribute is not configured in the certificate-status-profile, the required DNS query is issued on the wancom0 interface. This model requires configuration of the **dns-ip-primary** attribute, and optionally the **dns-ip-backup1** and
dns-ip-backup2 attributes for the wancom0 interface.

Access via an FQDN is supported by the **hostname** attribute in the
cert-status-profile configuration element.

The SBC allows configuration of both an OCSP responder IP address and port number (using the **ip-address** and **port** attributes) and an OCSP responder domain (using the **hostname** attribute).

In such cases the **verify-config** command issues a warning and notes that
IP address-based access will be used.

OCSP Access Configuration via IP Address

The following sample configuration accesses an OCSP responder at 192.168.7.100:8080.

```
ORACLE# configure terminal
ORACLE(configure)# security#
ORACLE(secirity)# cert-status-profile#
ORACLE(cert-status-profile)# show
cert-status-profile
       name                    defaultOCSP
       ip-address              192.168.7.100
       hostname
       port                    8080
       type                    OCSP
       trans-proto             HTTP
       requestor-cert          ocspVerisignClient
       responder-cert          VerisignCA2
       trusted-cas
       realm-id                admin
       retry-count             1
       dead-time               0
       last-modified-by
       last-modified-date
ORACLE(cert-status-profile)#
```

OCSP Access Configuration via FQDN

The following sample configuration accesses one or more OCSP responders at example.disa.mil.

Note that in the absence of a specified domain, the wancom0 interface must be DNS-enabled.

```
ORACLE# configure terminal
ORACLE(configure)# security#
ORACLE(cert-status-profile)# show
cert-status-profile
        name                        DISAdomain2
        ip-address
        hostname                    example.disa.mil
        port
        type                        OCSP
        trans-proto                 HTTP
        requestor-cert
        responder-cert
        trusted-cas                 dod1 dod2 disaA disaB IBM1
        realm-id
        retry-count                 1
        dead-time                   0
        last-modified-by
        last-modified-date
ORACLE(cert-status-profile)#
```

Direct and Delegated Trust Models

RFC 2560 specifies that an OCSP responder must digitally sign OCSP responses, and that an OCSP client must validate the received signature. In prior releases, successful validation of the signed response served to authenticate the responder. Such an authentication method is referred to as a direct trust model in that it does not require confirmation from a trusted Certificate Authority (CA). Rather it requires that the OCSP client be in possession of the public counterpart of the private key used by the OCSP responder to sign the response. This certificate is identified by the **responder-cert** attribute in the cert-status-profile configuration element.

The SBC continues support for the direct trust model, while also supporting an alternative delegated trust model as described in Section 5.4.6.2.1.6.1.e.3.c of UCR 2010. The delegated trust model requires that the OCSP responder be authenticated by a trusted CA. Within the DISA/DoD delegated trust model, an OCSP responder certificate is appended to every response, thus eliminating the need for a pre-provisioned responder certificate. The appended certificate is a signing certificate issued and signed by a DoD-approved CA that issued the certificate that is being validated. These OCSP responder certificates have a short lifespan and are reissued regularly.

Direct Trust Model Configuration

The direct trust model is used in virtually all commercial/enterprise environments. Configuration of the direct trust model is unchanged from that contained in the latest version of your hardware or the Oracle Communications Session Border Controller*Configuration Guide*.

Delegated Trust Model Configuration

The delegated trust model is used exclusively in some strict DISA/DoD environments; other DISA/DoD environments may support both the direct and delegated trust models.

Use the following procedure to configure OCSP for DISA/DoD environments.

1. From superuser mode, use the following command sequence to access cert-status-profile configuration mode. While in this mode, you configure a cert-status-profile configuration element, a container for the information required to access a single, specific OCSP responder.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# cert-status-profile
   ORACLE(cert-status-profile)#
   ```

2. The **name** attribute differentiates cert-status-profile configuration elements one from another. Each cert-status-profile provides configuration information for a single, specific OCSP responder.
3. The **type** attribute selects the certificate revocation check methodology, the only currently supported methodology is OCSP.
4. Retain the default value (http) for **trans-protocol** attribute, which identifies the transport method used to access the OCSP responder.
5. The **ip-address** attribute works in conjunction with the **port** attribute to provide the IP address of the OCSP responder.

   ip-address identifies the OCSP responder by its IP address. **port** identifies the port monitored by the HTTP server for incoming OCSP requests.

   The **port** attribute can be safely ignored if the OCSP responder is specified as a FQDN by the **host-name** attribute, but is required if the OCSP responder is identified by the **ip-address** attribute.

   Allowable **port** values are integers within the range 1025 through 65535. In the absence of an explicitly configured value, the system provides a default value of 80, the well-known HTTP port.
6. Alternatively, use the **host-name** attribute to identify the OCSP responder.

   host-name identifies the OCSP responder by a FQDN.

   If you provide both an IPv4 address/port number and a FQDN, the Oracle Communications Session Border Controller uses the IP address/port number and ignores the FQDN.

   If values are provided for both attributes, the Security Gateway uses the IP address and ignores the **host-name** value.
7. The **realm-id** attribute specifies the realm used to access the OCSP responder.

   In the absence of an explicitly configured value, the Oracle Communications Session Border Controller provides a default value of wancom0, specifying OCSP transmissions across the wancom0 management interface.

   If the OCSP responder identified by a FQDN, the realm identified by **realm-id** must be DNS-enabled.
8. The **requester-cert** attribute is meaningful only if OCSP requests are signed; ignore this attribute if requests are not signed.

   RFC 2560 does not require the digital signature of OCSP requests. OCSP responders, however, can impose signature requirements.

   If a signed request is required by the OCSP responder, provide the name of the certificate configuration element that contains the certificate used to sign OCSP requests.
9. The **responder-cert** attribute identifies the certificate used to validate signed OCSP response — a public key of the OCSP responder.

In DISA/DoD environments that support the direct trust model, optionally provide the name of the certificate configuration element that contains the certificate used to validate the signed OCSP response.

If a **responder-cert** is provided, it is only used if the OCSP response has no appended certificates, in which case the OCSP client attempts to validate the response signature. Depending on the validation failure or success, the response is rejected or accepted.

If the OCSP response has an appended certificate or certificate chain, the **responder-cert** is ignored, and the trusted-cas list is used to validate the appended certificate(s).

10. The **trusted-cas** attribute (a list of certificate configuration objects) identifies the approved DoD-approved CAs that sign OCSP responder certificates.

    In DISA/DoD environments that support the delegated trust model, you must provide a list of CAs used to validate the received certificate.

    If a certificate or a certificate chain is appended to the OCSP response, the OCSP client verifies that the first certificate signed the response, and that the CA is trusted by the Oracle Communications Session Border Controller (that is, the CA certificate is contained in the **trusted-cas** list. The client then walks through each additional certificate (if any exist) ensuring that each certificate is also trusted. If all certificates are trusted, the OCSP response is accepted; otherwise, it is rejected.

11. The **retry-count** attribute specifies the maximum number of times to retry an OCSP responder in the event of connection failure.

    If the retry counter specified by this attribute is exceeded, the OCSP requester contacts another responder (if multiple responders have been configured) and quarantines the unavailable responder for a period defined the **dead-time** attribute.

    In the absence of an explicitly configured value (an integer within the range 0 through 10), the Oracle Communications Session Border Controller provides a default value of 1 (connection retries).

12. The **dead-time** attribute specifies the quarantine period imposed on an unavailable OCSP responder.

    In the absence of an explicitly configured value (an integer within the range 0 through 3600 seconds), the Oracle Communications Session Border Controller provides a default value of 0 (no quarantine period).

    Customer implementations utilizing a single OCSP responder are encouraged to retain the default value, or to specify a brief quarantine period to prevent lengthy service outages.

13. Use **done**, **exit**, and **verify-config** to complete configuration of this cert-status-profile instance.

14. Repeat Steps 1 through 13 to configure additional cert-status-profile configuration elements.

IPv6-IPv4 Internetworking

The Oracle Communications Session Border Controller supports the following internetworking environments:

SIP-TLS IPv6 endpoints with SIP-TLS IPv4 endpoints

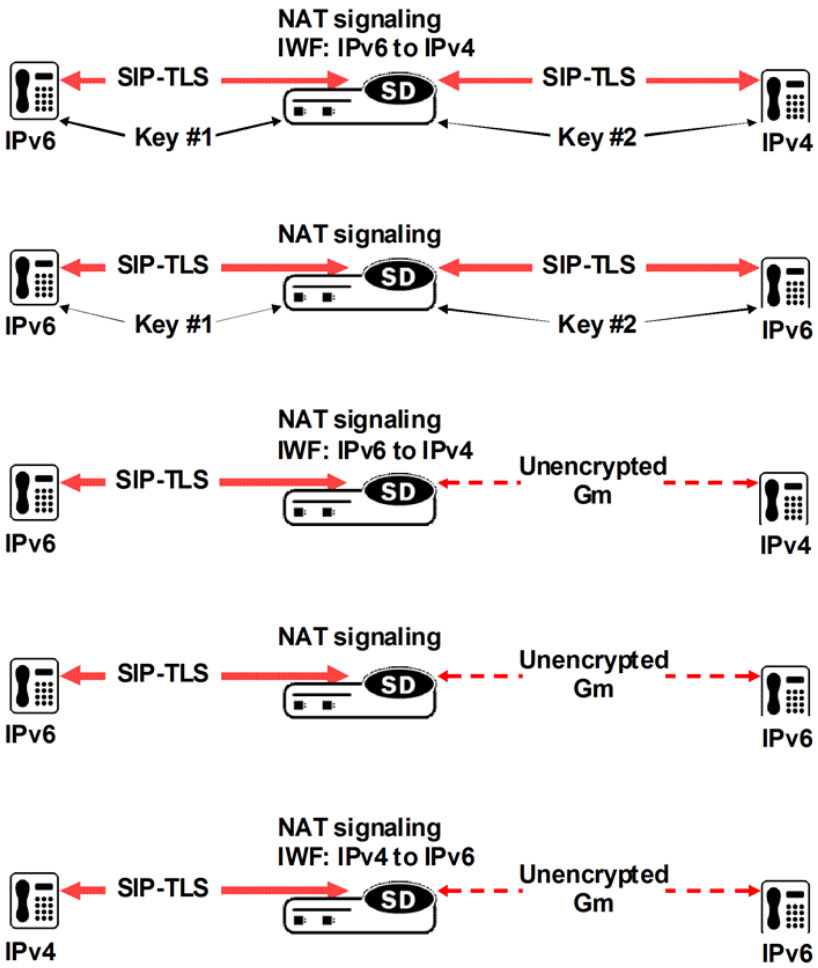
SIP-TLS IPv6 endpoints with SIP-TLS IPv6 endpoints

SIP-TLS IPv6 endpoints with non-SIP-TLS (unencrypted) IPv4 endpoints

SIP-TLS IPv6 endpoints with non-SIP-TLS (unencrypted) IPv6 endpoints

SIP-TLS IPv4 endpoints with non-SIP-TLS (unencrypted) IPv6 endpoints

> **Note:**
>
> Previously delivered TLS functionality, for example, support for certificate extensions, and support for certificate chain processing, is not affected by IPv6-IPv4 internetworking.



SRTP IPv4 IPv6 Internetworking

Internetworking IPv4 and IPv6 media while using SDES as the key exchange protocol is supported.

SRTP is defined in RFC 3711, *The Secure Real-time Transport Protocol (SRTP)*. It provides confidentiality, message authentication, and replay protection for RTP media and control traffic. SDES is defined in RFC 4568, *Session Description Protocol (SDP) Security Descriptions for Media Streams*. This RFC describes a new SDP cryptographic attribute that provides a secure method to provide security for unicast media streams.

Supported Topologies

The following internetworking topologies are supported and illustrated below

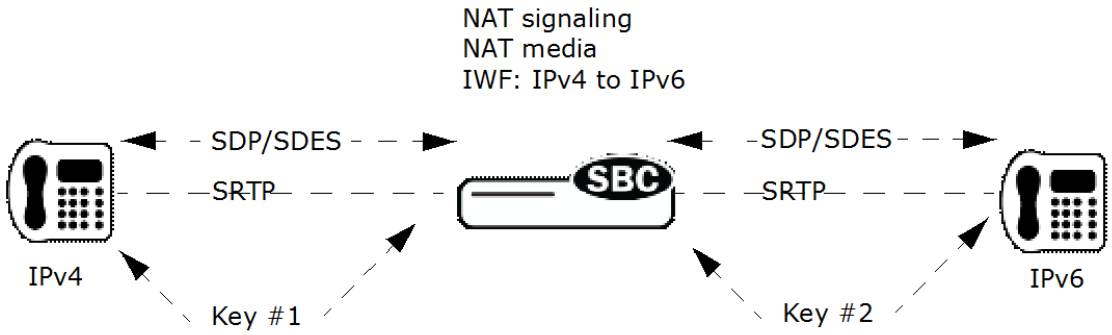
SRTP IPv4 endpoints with SRTP IPv4 endpoints

SRTP IPv4 endpoints with RTP (unencrypted) IPv6 endpoints

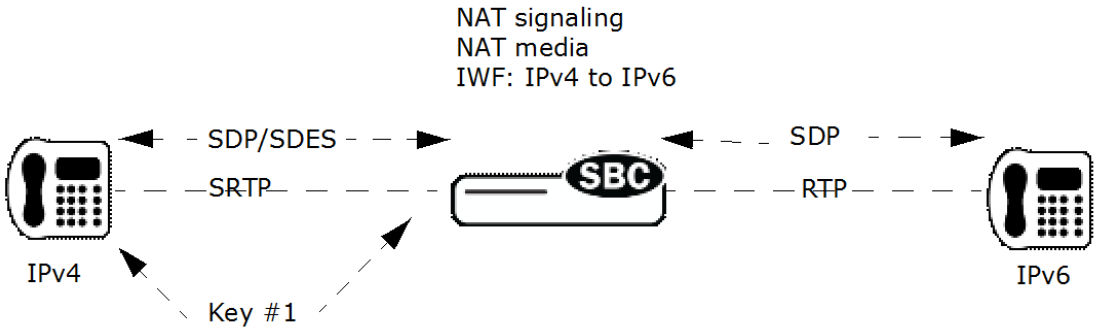
SRTP IPv6 endpoints with SRTP IPv6 endpoints

SRTP IPv6 endpoints with RTP (unencrypted) IPv4 endpoints

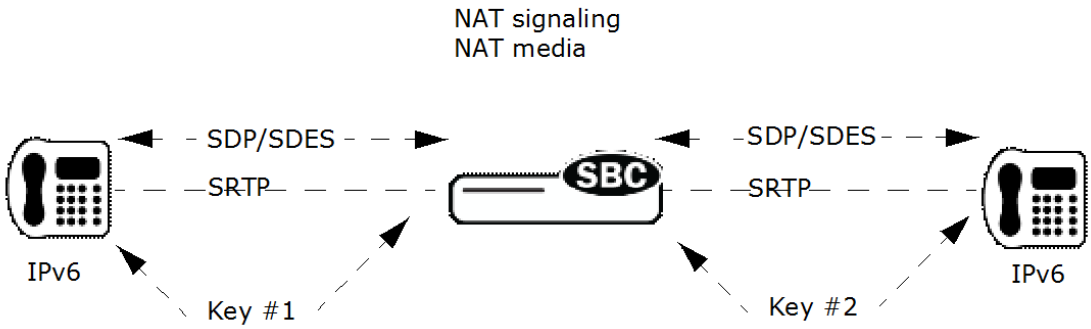
SRTP IPv6 endpoints with RTP (unencrypted) IPv6 endpoints



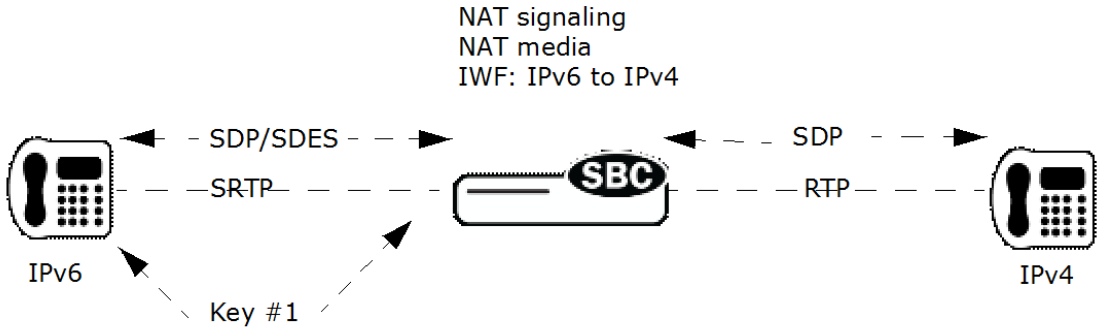
SRTP/SDES IPv4 to SRTP/SDES IPv6 Internetworking



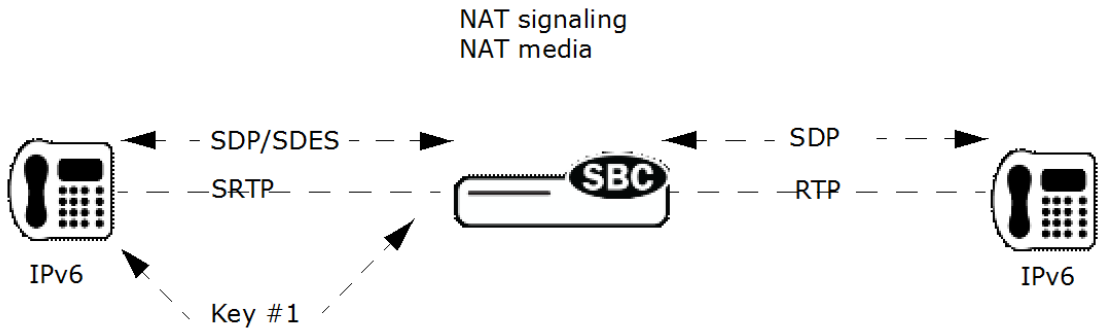
SRTP/SDES IPv4 to RTP IPv6 Internetworking



SRTP/SDES IPv6 to SRTP/SDES IPv6 Internetworking



SRTP/SDES IPv6 to RTP IPv4 Internetworking



SRTP/SDES IPv6 to RTP IPv6 Internetworking

Configuration

IPv6 support must be globally enabled to support SRTP internetworking as described above. If IPv6 is currently enabled, no additional configuration is required.

Key Exchange Protocols

Key exchange protocols enable secure communications over an untrusted network by deriving and distributing shared keys between two or more parties. The Internet Key Exchange (IKEv1) Protocol, originally defined in RFC 2409, provides a method for creating keys used by IPsec tunnels. Session Description Protocol Security Descriptions for Media Streams (SDES), defined in RFC 4568, provides alternative methods for creating keys used to encrypt Real-time Transport Protocol (RTP) and Real-time Transport Control Protocol (RTCP) transactions.

Each of these protocols is described in the following sections.

IKEv1 Protocol

IKEv1 is specified by a series of RFCs, specifically RFCs 2401 through 2412. The most relevant are:

- RFC 2407, *The Internet IP Security Domain of Interpretation for ISAKMP*
- RFC 2408, *Internet Security Association and Key Management Protocol (ISAKMP)*
- RFC 2409, *The Internet Key Exchange (IKE)*

- RFC 2412, *Oakley Key Determination Protocol*

IKEv1 combines features of the Internet Security Association and Key Management Protocol (ISAKMP) and Oakley Key Determination Protocol in order to negotiate Security Associations (SA) for two communicating peers. IKEv1 also provides for key agreement using Diffie-Hellman.

IKEv1 uses two phases. Phase 1 is used to establish an ISAKMP Security Association for IKEv1 itself. Phase 1 negotiates the authentication method and symmetric encryption algorithm to be used. Phase 1 requires either six messages (main mode) or three messages (aggressive mode).

Phase 2 negotiates the SA for two IPsec peers and is accomplished with three messages.

The initial IKEv1 implementation supports RFC 2409, *Internet Key Exchange*, and RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*.

IKEv1 Configuration

IKEv1 configuration consists of five steps.

1. Configure IKEv1 global parameters.
2. Optionally, enable and configure Dead Peer Detection (DPD) Protocol.
3. Configure IKEv1 interfaces.
4. Configure IKEv1 Security Associations (SA).
5. Assign the IKEv1 SA to an IPsec Security Policy.

IKEv1 Global Configuration

To configure global IKEv1 parameters:

> **Note:**
>
> DPD only works with IKEv2.

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you configure global IKEv1 configuration parameters.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-config
ORACLE(ike-config)#
```

2. Use the **ike-version** parameter to specify IKEv1.

   Use 1 to specify IKEv1 operations.

3. Use the **log-level** parameter to specify the contents of the IKEv1 log.

   Events are listed below in descending order of criticality.

   - emergency (most critical)
   - critical

- major
- minor
- warning
- notice
- info (least critical — the default)
- trace (test/debug, not used in production environments)
- debug (test/debug, not used in production environments)
- detail (test/debug, not used in production environments)

In the absence of an explicitly configured value, the default value of info is used.

4. Use the optional **udp-port** parameter to specify the port monitored for IKEv1 protocol traffic.

   In the absence of an explicitly configured value, the default port number of 500 is used.

5. Use the optional **negotiation-timeout** parameter to specify the maximum interval (in seconds) between Diffie-Hellman message exchanges.

   In the absence of an explicitly configured value, the default specifies a 15 second timeout value.

6. Use the optional **event-timeout** parameter to specify the maximum time (in seconds) allowed for the duration of an IKEv1 event, defined as the successful establishment of an IKE or IPsec Security Association (SA).

   In the absence of an explicitly configured value, the default specifies a 60 second time span.

7. Use the optional **phase1-mode** parameter to specify the IKE Phase 1 exchange mode.

   During Phase 1 the IKE initiator and responder establish the IKE SA, using one of two available methods.

   main mode — (the default) is more verbose, but provides greater security in that it does not reveal the identity of the IKE peers. Main mode requires six messages (3 requests and corresponding responses) to (1) negotiate the IKE SA, (2) perform a Diffie-Hellman exchange of cryptographic material, and (3) authenticate the remote peer.

   aggressive mode — is less verbose (requiring only three messages), but less secure in providing no identity protection, and less flexible in IKE SA negotiation.

   In the absence of an explicitly configured value, the default (main mode) is used.

8. Use the optional **phase1-dh-mode** parameter to specify the Diffie-Hellman Group used during IKE Phase 1 negotiation.

   - first-supported — as responder, use the first supported Diffie-Hellman group proposed by initiator

     > **Note:**
     >
     > Diffie-Hellman groups determine the lengths of the prime numbers exchanged during the symmetric key generation process.

   - dh-group5 — as initiator, propose Diffie-Hellman group 5 (1536-bit)
   - dh-group14 — as initiator, propose Diffie-Hellman group 14 (2048-bit)

- dh-group15 — as initiator, propose Diffie-Hellman group 15 (3072-bit)
- dh-group16 — as initiator, propose Diffie-Hellman group 16 (4096-bit)
- dh-group17 — as initiator, propose Diffie-Hellman group 17 (6144-bit)
- dh-group18 — as initiator, propose Diffie-Hellman group 18 (8192-bit)

9. If functioning as the IKE initiator, use the optional **phase1-life-seconds** parameter to specify the proposed lifetime (in seconds) for the IKE SA established during IKE Phase 1 negotiations.

   Allowable values are within the range 1 through 999999999 (seconds) with a default of 3600 (1 hour).

   This parameter can safely be ignored if functioning as a IKE responder.

10. If functioning as the IKE responder, use the optional **phase1-life-seconds-max** parameter to specify the maximum time (in seconds) accepted for IKE SA lifetime during IKE Phase 1 negotiations.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 86400 (1 day).

    This parameter can safely be ignored if functioning as a IKE initiator.

11. If functioning as the IKE initiator, use the optional **phase2-life-seconds** parameter to specify the proposed lifetime (in seconds) for an IPsec SA established during IKE Phase 2 negotiations.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 28800 (8 hours).

    This parameter can safely be ignored if functioning as a IKE responder.

12. If functioning as the IKE responder, use the optional **phase2-life-seconds-max** parameter to specify the maximum time (in seconds) accepted for IPsec SA lifetime during IKE Phase 2 negotiations.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 86400 (1 day).

    This parameter can safely be ignored if functioning as a IKE initiator.

13. Use the optional **phase2-exchange-mode** parameter to specify the Diffie-Hellman group used in Phase 2 negotiations.

    - phase1-group — (the default) use the same Diffie-Hellman group as used during Phase 1 negotiation
    - no-forward-secrecy — use the same key as used during Phase 1 negotiation

    > **Note:**
    >
    > Forward security indicates that compromise of a single key permits access only to data encrypted with that specific key. Failure to generate a new key for IKE Phase 2 potentially compromises additional data.

    - dh-group5 — as initiator, propose Diffie-Hellman group 5 (1536-bit)
    - dh-group14 — as initiator, propose Diffie-Hellman group 14 (2048-bit)
    - dh-group15 — as initiator, propose Diffie-Hellman group 15 (3072-bit)
    - dh-group16 — as initiator, propose Diffie-Hellman group 16 (4096-bit)

- dh-group17 — as initiator, propose Diffie-Hellman group 17 (6144-bit)
- dh-group18 — as initiator, propose Diffie-Hellman group 18 (8192-bit)

14. Use the **shared-password** parameter to specify the PSK (pre-shared key) used during authentication with the remote IKE peer.

    The PSK is a string of ACSII printable characters no longer than 255 characters (not displayed by the ACLI).

    This global PSK can be over-ridden by an interface-specific PSK.

15. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv1 global parameters instance.

IKEv1 Interface Configuration

To configure IKEv1 interface parameters:

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you configure IKEv1 interface parameters.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

2. Use the **state** parameter to enable or disable this IKE interface.
3. Use the **version** parameter to specify version 1 for this IKE interface.
4. Use the **address** parameter to specify the IPv4 address of the interface.
5. Use the **realm-id** parameter to specify the realm that contains the IP address assigned to this IKEv1 interface.
6. Use the **ike-mode** parameter to specify the operational mode, either responder (the default) or initiator.
7. If DPD has been enabled at the global level, use the **dpd-params-name** parameter to assign a DPD template, an operational set of DPD parameters, to the current IKEv1 interface.

   If DPD has not been enabled, this parameter can be safely ignored.
8. Use the optional **v1-ike-life-secs** parameter to set the IKE SA lifetime in seconds if you plan to use rekey on this interface.
   - Default: 3600
   - Range: 1 to 999999999

   Recommendations:
   - Because smaller values impact performance, the minimum recommended value is 200.
   - The maximum recommended value for rekeying IKE SA connections is 24 hours (86400 seconds).
   - The initiator and responder must not have the same rekey value. A gap of at least 100 seconds is recommended.

9. Use the optional **v1-ipsec-life-secs** parameter to set the IPsec SA lifetime in seconds if you plan to use rekey on this interface.
   - Default: 28800
   - Range: 1 to 999999999

   Recommendations:
   - Because smaller values impact performance, the minimum recommended value is 200.
   - The maximum recommended value for IPsec SA connections is 8 hours (28800 seconds).
   - The initiator and responder must not have the same rekey value. A gap of at least 100 seconds is recommended.
10. **v1-rekey**—(Optional) Enable or disable the automatic re-keying of expired IKEv1 or IPsec SAs on this IKEv1 interface.
11. Use the optional **shared-password** parameter to assign an interface PSK.

    This IKEv1-interface-specific value over-rides the global default value set at the IKE configuration level.
12. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv1 interface.
13. Repeat Steps 1 through 7 to configure additional IKEv1 interfaces.

IKEv1 Security Association Configuration

An IKEv1 SA identifies cryptographic material available for IPsec tunnel establishment.

To configure IKEv1 SA parameters:

1. Access the **ike-sainfo** configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-sainfo
ORACLE(ike-sainfo)#
```

2. Use the required **name** parameter to provide a unique identifier for this ike-sainfo instance.

   name enables the creation of multiple ike-sainfo instances.
3. Use the **security-protocol** parameter to specify the IPsec security (authentication and encryption) protocols supported by this SA.

   The following security protocols are available:
   - ah
   - esp
   - esp-auth (default)

   Refer to the following figures for additional details.

Figure 14-1 AH Transport Mode

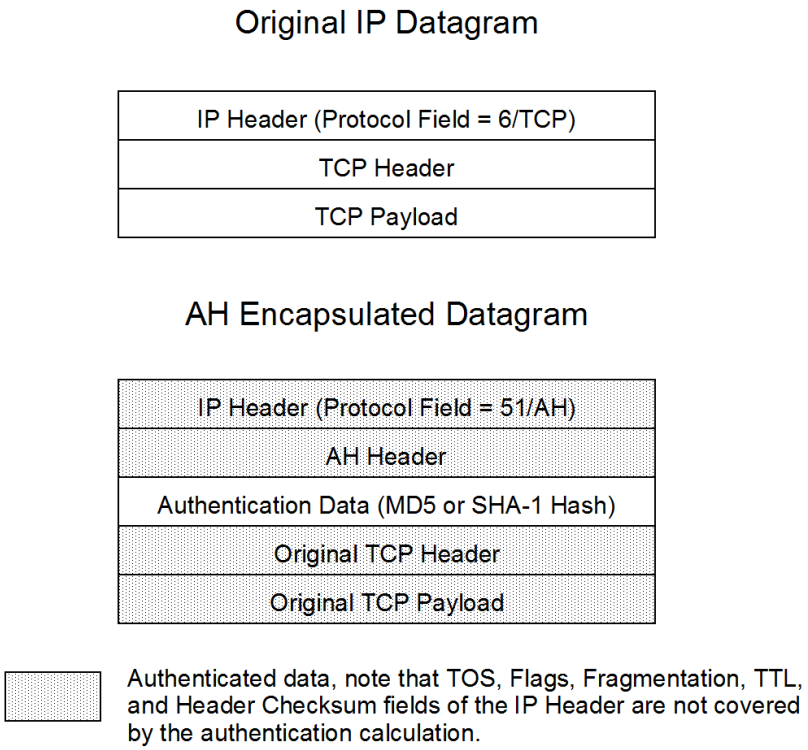


Figure 14-2 AH Tunnel Mode

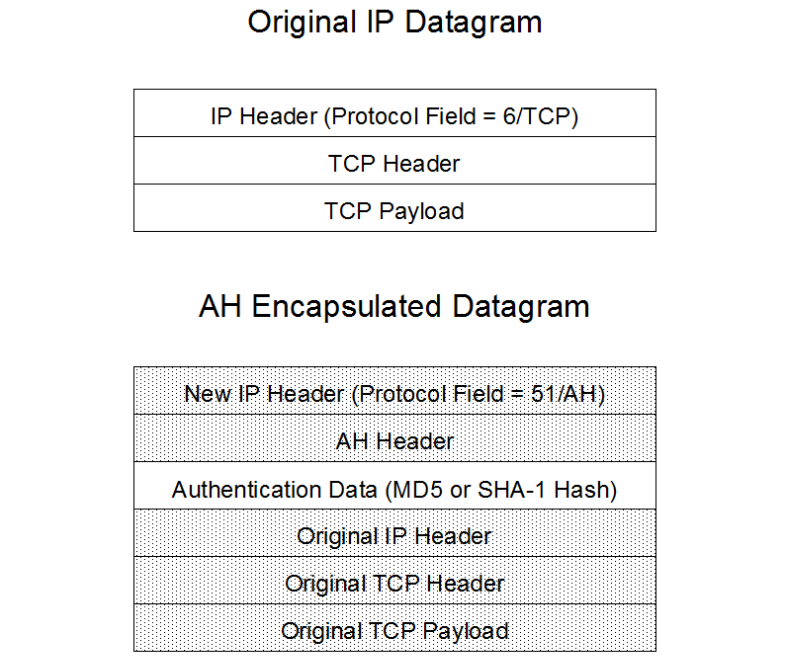


Figure 14-3 ESP Transport Mode

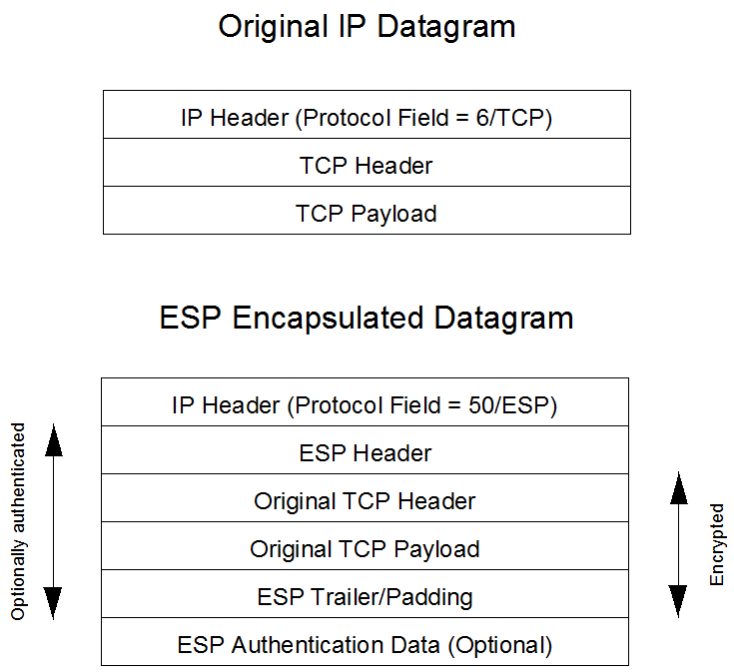
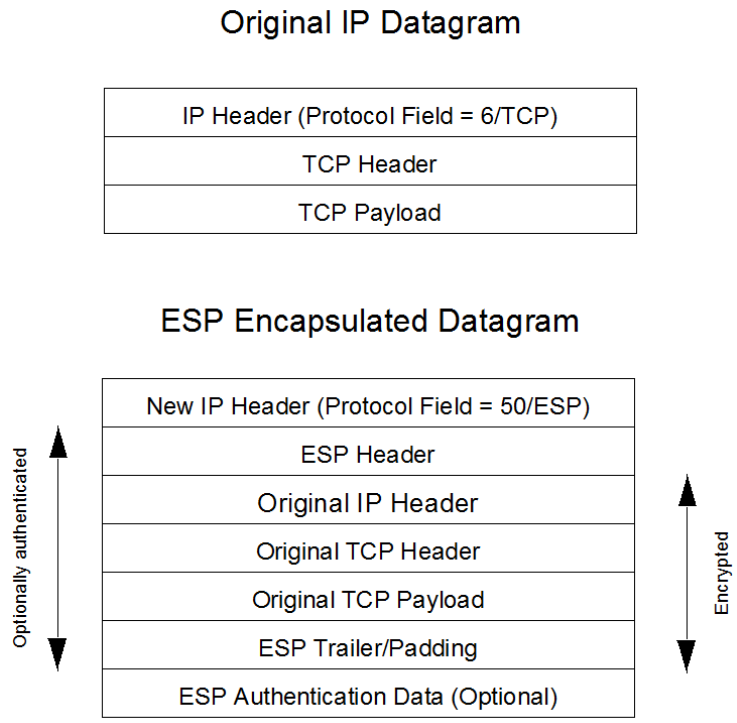


Figure 14-4 ESP Tunnel Mode




```
ORACLE(ike-sainfo)# security-protocol esp
```

4. Use the **auth-algo** parameter to specify the authentication algorithms supported by this SA.

   The following authentication protocols are available:

   - any
   - aes-xcbc
   - sha2-256
   - sha2-384
   - sha2-512 (default)

   ```
   ORACLE(ike-sainfo)# auth-algo sha2-512
   ```

5. Use the **encryption-algo** parameter to specify the encryption algorithms supported by this SA.

   The following encryption protocols are available:

   - any
   - aes-ctr
   - aes (default)

   ```
   ORACLE(ike-sainfo)# encryption-algo aes
   ```

6. Use the **ipsec-mode** parameter to specify the IPSec operational mode.

   Transport mode (the default) provides a secure end-to-end connection between two IP hosts. Transport mode encapsulates the IP payload.

   Tunnel mode provides VPN service where entire IP packets are encapsulated within an outer IP envelope and delivered from source (an IP host) to destination (generally a secure gateway) across an untrusted internet.

   Refer to the previous figures for encapsulation details.

   ```
   ORACLE(ike-sainfo)# ipsec-mode tunnel
   ```

7. If **ipsec-mode** is tunnel, use the required **tunnel-local-addr** parameter to specify the IP address of the local IKEv1 interface that terminates the IPsec tunnel.

   This parameter can safely be ignored if **ipsec-mode** is transport.

   ```
   ORACLE(ike-sainfo)# tunnel-local-addr 192.169.204.14
   ORACLE(ike-sainfo)#
   ```

8. If **ipsec-mode** is tunnel, use the **tunnel-remote-addr** parameter to specify the IP address of the remote IKEv1 peer that terminates the IPsec tunnel.

   Provide the remote IP address, or use the default wild-card value (*) to match all IP addresses.

   This parameter can safely be ignored if **ipsec-mode** is transport.

   ```
   ORACLE(ike-sainfo)# tunnel-remote-addr *
   ```

9. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv1 SA.
10. Repeat Steps 1 through 9 to configure additional IKEv1 SAs.

IPsec Security Policy Configuration

Use the following procedure to assign an IKEv1 SA to an existing IPsec Security Policy. Note that the network interface supported by the IPsec Security Policy must have been configured as an IKEv1 interface.

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you configure global IKEv1 configuration parameters.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# security-policy
ORACLE(security-policy)#
```

2. Use the required **ike-sainfo-name** parameter to assign an IKv1 SA to this IPsec Security Policy.
3. Use **done**, **exit**, and **verify-config** to complete configuration of IPsec Security Policy.

IKEv2 Protocol

IKEv2 is specified by a series of RFCs, specifically RFCs 2401 through 2412. The most relevant are:

- RFC 2412, *Oakley Key Determination Protocol*
- RFC 4301, *Security Architecture for the Internet Protocol*
- RFC 4306, *Internet Key Exchange (IKEv2) Protocol*
- RFC 4718, *IKEv2 Clarifications and Implementation Guidelines*
- RFC 5996, *Internet Key Exchange (IKEv2) Protocol*

IKEv2 combines features of the Internet Security Association and Key Management Protocol (ISAKMP) and Oakley Key Determination Protocol in order to negotiate Security Associations (SA) for two communicating peers. IKEv2 also provides for key agreement using Diffie-Hellman.

The initial IKEv2 implementation supports RFC 4306, *Internet Key Exchange (IKEv2) Protocol*, and RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*.

IKEv2 Support

The Oracle Communications Session Border Controller (SBC) supports version 2 of the Internet Key Exchange (IKE) protocol. IKEv2 provides an initial handshake in which IKE peers negotiate cryptographic algorithms, mutually authenticate, and establish session keys to create an IKEv2 Security Association (SA) and an IPsec SA.

Key elements of IKEv2 support include:

- Peering/SIP Trunking solutions and access-side use cases
- Mutual authentication between the SBC and its peers, including:
  - IKE rekey
  - Dead Peer Detection (DPD)

  - Initiator mode
  - Responder mode
- Per-interface IKEv2 configuration
- Simultaneous support of IKEv1 and IKEv2 protocols
- Either tunnel or transport mode supported per IKE interface
- Transcoding
- Separate interfaces and IP addressing for SIP and IKE for related traffic
- Certificate-based authentication during IKEv2 tunnel establishment
- Multiple endpoints beyond tunnel remote address
- Single or Multiple PSKs over a single IKE interface

With respect to IKE, if the peer does not support any of the encryption, hashing and integrity algorithms and Diffie Hellman groups supported by the SBC, it rejects the IKEv2 establishment. With respect to IPsec, if the peer does not support any of the encryption, hashing and integrity algorithms supported by the SBC, it does not create the child SA.

This can be implemented by removing these from the default list but allow manual configuration to add support.

At the IKEv2 global configuration level, users can do the following:

1. Configure IKEv2 global parameters.
2. Configure a default certificate profile.
3. Configure one or more RADIUS authentication servers (optional).
4. Configure one or more RADIUS authorization servers (optional).
5. Configure the default address pool (optional).
6. Configure pre-shared-keys if authentication is based on the contents of the IKEv2 Identification payload (optional).

To a large extent, global configuration establishes profiles that either apply to specific traffic and interfaces or you apply to elements by further configuration. To the extent that there is any overlapping configuration, the interface level takes precedence over global configuration.

IKEv2 Global Configuration

This section covers IKEv2 global configuration parameters, omitting IKEv1 parameters. A parameter within the global **ike-config** element can be overridden by the same parameter within the **ike-interface** element.

1. Access the **ike-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-config
```

2. **state**—Set to **enable**.
3. **ike-version**—Select the IKE protocol version **2**.

> **WARNING:**
>
> Enabling version 2 in the **ike-config** element disables version 1 globally.

4. **log-level**—Specify the level of the IKEv2-related logs.

   Log messages are listed below in descending order of severity.

   - emergency
   - critical
   - major
   - minor
   - warning
   - notice
   - info — (default)
   - trace — (test/debug, not used in production environments)
   - debug — (test/debug, not used in production environments)
   - detail — (test/debug, not used in production environments)

5. **udp-port**—Set to **500**.
6. **negotiation-timeout** —Use the optional parameter to specify the maximum interval (in seconds) between Diffie-Hellman message exchanges.

   In the absence of an explicitly configured value, the default specifies a 15 second timeout value.

7. **v2-ike-life-secs**—Specify the default lifetime (in seconds) of the IKEv2 SA.

   Allowable values are within the range 1800 through 999999999.

8. **v2-ipsec-life-secs**—Specify the default lifetime (in seconds) for the IPsec SA.

   Allowable values are within the range 1 through 999999999.

9. **v2-rekey** —Enable or disable the re-keying of expired IKEv2 or IPsec SAs.

   When **v2-rekey** is enabled, the SBC initiates a new negotiation to restore an expired IKEv2 or IPsec SA. The SBC makes a maximum of three retransmission attempts before abandoning the re-keying effort.

10. **anti-replay** —(Optional) Enable or disable anti-replay protection on IPsec SAs.
11. **shared-password**—If using a shared password, provide the PSK used while authenticating the remote IKEv2 peer.

    Ensure the remote peer is configured with the same PSK.

    The value of **shared-password** in the **ike-interface** configuration element takes precedence over this value.

12. **eap-bypass-identity**—(Optional) Specify whether or not to bypass the EAP identity phase.
13. **dpd-time-interval**—(Optional) Specify the maximum period of inactivity (in seconds) before the DPD protocol is initiated on an endpoint.

    Values are within the range 1 through 999999999 (seconds).

14. **overload-threshold**—Set the percentage of CPU usage that triggers an overload state.

Values are within the range 1 through 100, and less than the value of overload-critical-threshold.

15. **overload-interval**—Set the interval (in seconds) between CPU load measurements while in the overload state.

    Values are within the range 0 through 60 (seconds).

16. **overload-action**—Select the action to take when the Oracle Communications Session Border Controller (as a SG) CPU enters an overload state. The overload state is reached when CPU usage exceeds the percentage threshold specified by the overload-threshold.

    Available values are:

    - none—(the default)
    - drop-new-connection—use to implement call rejection

17. **overload-critical-threshold**—Set the percentage of CPU usage that triggers a critical overload state. This value must be greater than the value of overload-threshold.

    Values are within the range 1 through 100.

18. **overload-critical-interval**—Set the interval (in seconds) between CPU load measurements while in the critical overload state.

    Values are within the range 0 through 60 (seconds).

19. **sd-authentication-method**—Select the method used for local authentication of the IKEv2 peer.

    Two authentication methods are supported:

    - shared-password — (the default) uses a pre-shared key (PSK) to authenticate the IKEv2 peer.
    - certificate — uses an X.509 certificate to authenticate the IKEv2 peer.

    > **Note:**
    >
    > If using a certificate for authentication, see the "Certificate Configuration Process" section in the Security chapter of the *ACLI Configuration Guide*.

    The **sd-authentication-method** value can be overridden at the **ike-interface** level.

20. **certificate-profile-id**—If using a certificate, specify the **ike-certificate-profile** configuration element that contains identification and verification credentials required for PKI certificate-based IKEv2 authentication.

    The **ike-certificate-profile** value can be over-ridden at the **ike-interface** level.

21. **id-auth-type** —(Optional) Specify that the PSK used while authenticating the remote IKEv2 peer is associated with the asserted identity contained within an IKEv2 Identification payload.

    > **Note:**
    >
    > This attribute can be safely ignored if the PSK is defined globally or at the IKEv2 Interface level.

    Available values are:

- idi—use IDi KEY_ID for authentication
- idr—use IDr KEY_ID for authentication

22. **account-group-list**—(Optional) Designate one or two existing IPsec accounting groups as available to support IPsec accounting transactions.
23. Type **done**.

Multiple PSKs per IKE Interface

You can configure the SBC to support multiple pre-shared keys (PSKs) on a single IKE interface. By allowing these multiple PSK authentications, you can support multiple IKE sessions on that interface using unique PSKs. Both symmetric and asymmetric PSK deployments benefit from this capability. You accomplish this by attaching authorization configuration directly to SAs instead of the IKE interface.

A single PSK applies to an entire interface when you configure it within the **shared-password** parameter in the **ike-interface** element. The SBC associates this PSK with the **ike-interface** IP address. The SBC uses the IP associated with the **ike-interface** as its identity for the ID payload, and the PSK present in the **shared-password** field as the secret for authentication.

> **Note:**
>
> The **ike-config** also has a **shared-password** parameter, which the system uses globally if there is no **shared-password** in the **ike-interface**.

To configure this feature, you perform steps that are equivalent to configuring for a single PSK. In addition to these steps, you also define additional **ike-key-id** objects and assign them to the **local-id-profile** and **remote-id-profile** parameters in the applicable **ike-sainfo** elements.

- The **local-id-profile** refers to a unique **ike-key-id**, which contains the identity and PSK for the SBC.
- The **remote-id-profile** refers to a unique **ike-key-id**, which contains the identity and PSK for the remote device.

This allows you to establish multiple PSKs by associating the security associations with:

- IP addressing—From the **ike-sainfo** tunnel addresses
- Identity for the ID payload—From the **key-id** in the **ike-key-id**
- The secret for authentication—From the **presharedkey** in the **ike-key-id**

For this feature to operate properly, the **local-id-profile** and **remote-id-profile** must refer to different **ike-key-id** elements. This provides distinct support for asymmetric-psk deployments. Separate profiles are also required for symmetric-psk deployments because the SBC key-id must be different from the remote user's key-id, even if the PSKs (passwords) are the same.

If you do not configure either a **local-id-profile**, **remote-id-profile** or both, the system uses the password from the **ike-interface** or **ike-config** for authentication in the applicable direction (or both directions).

This feature also requires that you configure the **id-type** parameter in the **ike-key-id** to the correct key-ID type. Potential values include:

- IPv4—Version 4 address
- IPv6—Version 6 address

- KEY-ID—If you configure the **id-type** with an **ID_KEY_ID** value, that value must be 1-128 alphanumeric characters, and can include the '_','.' or '-' characters. This string may not begin with the '.' or '-' characters.

The SBC checks your configuration when you run **verify-config** and informs you of the following potential multi-PSK errors:

- The **local-id-profile** and **remote-id-profile** are the same.
- The **presharedkey** parameter in the **ike-key-id** is left empty.
- The **key-id** value does not match the IPv4 or IPv6 **id-type**.

Reporting

You can see connections during troubleshooting procedures using the **show security ike sad ike-interface <IP addr> all** command during troubleshooting procedures.

```
ORACLE# show security ike sad ike-interface 12.16.23.24 all
Displaying the total (1) number of entries may take long and could affect
system
performance.
Continue? [y/n]?: y
Peer: 12.16.23.25:4500 (NAT: Yes) Host: 12.16.23.24 State: Up
IKEv1 Cookies: 0x7f88c01a9e1e3e76[I] 0x02f088ad845d9640[R] rekeying in 194
seconds
Child Peer IP: 12.16.23.26:0 Child SPI: 118247482[I] 3363465612[O] Protocol:
ESP
TUNNEL Mode rekeying in 139 seconds
```

RADIUS Authentication

All EAP-based authentication is performed by RADIUS servers. When such authentication is specified, the Oracle Communications Session Border Controller operates as a relay between the remote IKVv2 peer and a RADIUS authentication server.

Configuring RADIUS Authentication

RADIUS authentication support requires:

- configuration of a pool of RADIUS authentication servers, with each server configuration record providing all values required for server access
- configuration of a RADIUS Authentication Servers List designating specific pool member as being available for authentication purposes
- assignment of the RADIUS Authentication Servers List to the authentication configuration object

Configure a RADIUS Server

1. Access the **radius-servers** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# authentication
ORACLE(authentication)# radius-servers
ORACLE(radius-servers)#
```

2. **state**—Set the operational state of this RADIUS authentication server.

Retain the default value, enabled, to identify this RADIUS authentication server as operational. Use disabled to place this RADIUS authentication server in a non-operational mode.

3. **authentication-methods**—Specify the authentication methods supported by this RADIUS authentication server.

   Valid values are:

   - pap
   - chap
   - mschapv2
   - eap
   - all

4. **address**—Specify the IP address of this RADIUS authentication server.
5. **port**—Specify the remote port monitored for RADIUS authentication requests.

   Valid values are:

   - 1645
   - 1812

6. **realm-id**—Identify the realm that provides transport services to this RADIUS authentication server.
7. **secret**—Specify the shared secret between the Oracle Communications Session Border Controller and this RADIUS authentication server.
8. **nas-id**—Provide a string that uniquely identifies the SBC to this RADIUS authentication server.

   For example:

   ```
   ORACLE(radius-servers)# nas-id nas-id-170-30-0-1
   ORACLE(radius-servers)#
   ```

9. **retry-limit**—Specify the number of times the SBC retransmits an unacknowledged authentication request to this RADIUS authentication server.

   - Min: 1
   - Max: 5

10. **retry-time**—Specify the interval (in seconds) between unacknowledged authentication requests.

    - Min: 5
    - Max: 10

11. **dead-time**—Specify the length (in seconds) of the quarantine period imposed an unresponsive RADIUS authentication server.

    - Min: 10
    - Max: 10000

12. **maximum-sessions**—Specify the maximum number of outstanding sessions for this RADIUS authentication server.

    - Min: 1

- Max: 255

13. **class**—Select the RADIUS authentication server class, either primary or secondary.

    The SBC tries to initiate contact with primary RADIUS authentication servers first, and only turns to secondary RADIUS authentication servers if no primaries are available.

    If more than one RADIUS authentication server is designated as primary, the SBC uses a round-robin strategy to distribute authentication requests among available primaries.

14. Type **done** to save your configuration.

15. If necessary, configure additional RADIUS authentication servers.

Configure a RADIUS Authentication Servers List

1. Access the **auth-params** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# auth-params
ORACLE(auth-params)#
```

2. **name**—Provide a unique name for this RADIUS Authentication Servers List.

3. **servers**—Compile a RADIUS Authentication Servers List.

    Provide the IP address of a previously configured RADIUS authentication server to add that server to this list.

```
ORACLE(auth-params)# servers 172.30.0.1 172.30.0.15 168.27.3.3
ORACLE(auth-params)#
```

4. Type **done** to save your configuration.

5. If necessary, configure additional RADIUS Authentication Servers Lists.

6. Access the **authentication** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# authentication
ORACLE(authentication)#
```

7. **ike-radius-params-name**—Assign a previously configured RADIUS Authentication Servers List to the authentication configuration element.

8. Type **done** to save your configuration.

Tearing Down IPsec Tunnels

If EAP-based authentication is used in conjunction with RADIUS-based assignment of requested local addresses, the Oracle Communications Session Border Controller responds to a Disconnect-Request message (as defined in RFC 5176, Dynamic Authorization Extensions to Remote Authentication Dial-In User Service) received from a configured RADIUS server.

The SBC parses the received Disconnect-Request for User-Name and Framed-IP-address attribute values. If the User-Name value matches the authenticated EAP identity, and the Framed-IP-address value matches the inner IP address assigned to the authenticated endpoint, the SBC deletes the IPsec tunnel described by the received values. Tunnel deletion is reported to the RADIUS server with a Disconnect-ACK message, which, in conformity to

Section 3.5 of RFC 5176, contains an Error Cause of 201 indicating Residual Session Context Removed.

If the IPsec tunnel cannot be deleted because of faulty/incorrect User-Name and/or Framed-IP-address values, the SBC returns a Disconnect-NACK message, which, in conformity to Section 3.5 of RFC 5176, contains an Error Cause of 404 indicating Invalid Request.

Enable RADIUS Authorization

Complete RADIUS authorization configuration by enabling RADIUS authorization on an IKEv2 interface.

1. Access the **ike-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

2. Use the **select** command to specify the target interface.
3. **authorization**—Enable RADIUS authorization on the selected interface.
4. Type **done** to save your configuration.
5. If necessary, enable RADIUS authorization on additional IKEv2 interfaces.

Local Address Pool Configuration

If your network environment requires local address pools that serve as a source of IPv4 or IPv6 addresses temporarily leased for use by remote IKEv2 peers, use the procedures in the following two sections to configure such pools.

During the IKE_AUTH exchange, the IPsec initiator (the remote endpoint) often requests an internal IP address from an IPsec responder (the Oracle Communications Session Border Controller). Refer to Section 2.19 of RFC 7296, Internet Key Exchange (IKEv2) Protocol, for a description of the request process. Procuring such a local IP address ensures that traffic returning to the endpoint is routed to the SBC, and then tunneled back to the endpoint. Local address pools provide the source of these addresses available for temporary endpoint lease.

After address assignment from the local address pool, the endpoint retains rights to that IP address for the tunnel lifetime. Tunnels are terminated either by an INFORMATIONAL exchange, defined in Section 1.4 of RFC 7296, or by expiration of the tunnel SAs as specified by the **v2-ike-life-seconds** and **v2-ipsec-life-seconds** configuration parameters. In either case, a subsequent request for an assigned IP address may, or may not result, in the assignment of the previous IP address. However, the SBC can be configured to ensure that a prematurely terminated tunnel, resulting for example from the reset or re-boot of the remote IP peer, can be restored with that previous address. Refer to Persistent Tunnel Addressing in this chapter for operational and configuration details.

During the IKE_AUTH request phase, the IKEv2 initiator can use the Configuration payload in conjunction with either the INTERNAL_IP4_DNS or INTERNAL_IP6_DNS attribute to request the addresses of DNS providers from the SBC. In environments where authorization is performed by a RADIUS AAA server, there are two potential sources of DNS information: local SBC DNS configuration elements, and external RADIUS servers that may provide DNS information in the Access-Accept packet that concludes a successful authentication effort. The source of DNS information provided by the SBC to an IKEv2 peer is subject to user configuration.

A RADIUS source of DNS information is enabled by support for certain Microsoft vendor-specific RADIUS attributes specified in RFC2548, Microsoft Vendor-Specific RADIUS Attributes. Operationally, the SBC extracts the values of the MS-Primary-DNS-Server and MS-Secondary-DNS-Server attributes from an Access-Accept packet and returns these values to the IKEv2 initiator.

When the DNS information is from external source, the SBC installs a NAT flow (a static traffic path) that provides access to the DNS server. The NAT flow is calculated based on the location of the DNS server IP returned from RADIUS AAA server and configured realm information.

Configuration of DNS information services takes place at the local address pool and IKEv2 interface levels.

Data Flow Configuration

If you need to configure address pools, first configure data flows and then assign them to a specific local address pool. A data flow establishes a static route between a remote IKEv2 peer and a core gateway or router which provides routing services after the associated traffic exits the Oracle Communications Session Border Controller.

1. Access the **data-flow** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# data-flow
ORACLE(data-flow)#
```

2. **name**—Provide a unique identifier for this data-flow instance.
3. **realm-id**—Identify the realm that supports data-flow upstream traffic, that is traffic toward the network core.
4. **group-size**—(Optional) Specify the maximum number of user elements grouped together by this **data-flow** instance.

   The size of the associated local-address-pool is divided by this value to segment the address pool into smaller groups. After determining the start address for each of the smaller address groups, the SBC uses the **data-flow** configuration to establish two static flows for each of the address groups — a downstream data-flow, in the access direction, and an upstream data-flow (via the realm specified by the **realm-id** parameter) toward a core gateway/router which provides forwarding service for the pass-thru data-flow.

   Allowable values are the powers of 2 between 1 through 256.

```
ORACLE(data-flow)# group-size 32
```

5. **upstream-rate**—Specify the allocated upstream bandwidth.
   - Min: 0 (allocate all available bandwidth)
   - Max: 122070
6. **downstream-rate**—Specify the allocated downstream bandwidth.
   - Min: 0 (allocate all available bandwidth)
   - Max: 122070
7. Type **done** to save your configuration.

Local Address Pool Configuration

You configure an address pool by associating a contiguous range or ranges of IPv4 or IPv6 addresses with an existing data-flow.

> **Note:**
>
> An address pool can contain multiple contiguous ranges of IP addresses. However, all defined ranges must specify the same type of IP address: You cannot include IPv4 and IPv6 addresses in the same address pool.

1. Access the **local-address-pool** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# local-address-pool
ORACLE(local-address-pool)#
```

2. **name**—Provide a unique identifier for this local-address-pool instance.
3. **dns-assignment**—Identify the DNS source used to respond to incoming IKE_AUTH requests for DNS information.
   - **local**—Use locally configured configuration data as the source of DNS information
   - **radius**—Use a remote RADIUS AAA server as the source of DNS information.
   - **radius-local**—Use a remote RADIUS AAA server as the preferred source of DNS information. If no DNS data is available from the RADIUS server, use locally configured DNS information.
4. **dns-realm-id**—Provide the name of the realm that supports transit to that RADIUS server.

   The **dns-realm-id** parameter can be safely ignored if **local** is specified as the DNS source.
5. **data-flow**—Identify the data-flow configuration element assigned to this local-address-pool instance.
6. **address-range**—Access the **address-range** configuration mode.
   - If building an address pool of contiguous IPv4 addresses, use **network-address** with **subnet-mask** to define a contiguous range of IPv4 addresses.

```
ORACLE(address-range)# network-address 192.168.0.0
ORACLE(address-range)# subnet-mask 255.255.255.96
```

> **Note:**
>
> The range of IPv4 addresses support only Class-B and Class-C subnet masks.

- If building an address pool of contiguous IPv6 addresses, use **network-address** parameter to provide both the IPv6 address and the bit length of the network prefix (an integer within the range 1 through 128). Leave the **subnet-mask** blank.

```
ORACLE(address-range)# network-address 1080::ac10:202/96
```

7. Type **done** to save your configuration. and **exit** to complete configuration of the address-range instance.
8. If required, add additional address ranges to this address-range instance
9. Type **done** to complete configuration of the local-address-pool instance.

Persistent Tunnel Addressing

After address assignment from the local address pool, the endpoint retains rights to that IP address for the tunnel lifetime. Tunnels can be terminated either by an INFORMATIONAL exchange, defined in Section 1.4 of RFC 7296, or by expiration of the tunnel SAs as specified by the **v2-ike-life-seconds** and **v2-ipsec-life-seconds** parameters. In either case, a subsequent request for an assigned IP address may, or may not result, in the assignment of the previous IP address. However, the Oracle Communications Session Border Controller can be configured to ensure that a prematurely terminated tunnel can be restored with that previous address.

Tunnels are usually prematurely terminated because of re-boot or reset of the remote endpoint. In either case, the endpoint's IKEv2 and IPsec SAs are lost and the tunnel no longer exists. From the point of view of the SBC, however, the tunnel remains live. The local IKEv2 and IPsec SAs still exist, and the tunnel remains available in an active state until the expiration of the lifetime timers. Similarly, the IP address assignment from the local address poll remains in effect until timer expiration.

When a crashed endpoint attempts to re-establish a tunnel, it can insert a Notify payload in the initial IKE_AUTH request. The Notify payload contains an INITIAL_CONTACT message that asserts a prior connection between the endpoint and the SBC. When receiving an INITIAL_CONTACT message, the SBC checks for the existence of a live tunnel with the requesting endpoint. If such a tunnel is found, the SBC stores the assigned IP address, tears down the tunnel by removing the supporting IKEv2 and IPsec SAs, and authenticates the endpoint. Assuming authentication succeeds, the SBC, retrieves the previously assigned IP address and returns it to the endpoint.

If a live tunnel is not found (meaning that the tunnel has timed out during the interval between the endpoint reset/re-boot and the new IKE_AUTH), the assertion of a prior connection is ignored, and address assignment is made from the local address pool.

You can use a global configuration option (**assume-initial-contact**) to enable persistent address processing with or without the reception of an INITIAL_CONTACT message. With this option enabled, all IKE_AUTH requests are processed as if they contained an INITIAL_CONTACT message.

Persistent Tunnel Addressing Configuration

Use the following command sequence to enable persistent tunnel addressing.

1. Access the **ike-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
```

```
ORACLE(security)# ike
ORACLE(ike)# ike-config
```

2. **options**—Enable address persistence.

```
ORACLE(ike-config)# options +assume-initial-contact
ORACLE(local-address-pool)#
```

3. Type **done** to save your configuration.

ike-key-id Configuration

If authentication between IKEv2 peers is based on a PSK associated with an identity asserted in the IKE Identification Payload, associate received asserted identities with a specified PSK.

1. Access the **ike-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-key-id
ORACLE(ike-key-id)#
```

2. **name**—Provide a unique identifier for this ike-key-id instance.
3. **keyid**—Specify the **keyid** to associate an asserted identity with a PSK.

```
ORACLE(ike-key-id)# keyid 172.16.20.20
```

4. **presharedkey**—Specify the string to associate an asserted identity with a PSK.

```
ORACLE(ike-key-id)# presharedkey ************************
```

5. **id-type**—Specify the data type of the **keyid** parameter you configured above.
   - ipv4 (default)—Specifies that this keyid is IPv4 (ID_IPV4_ADDR)
   - ipv6—Specifies that this keyid is IPv6 (ID_IPV6_ADDR)
   - key—Specifies that this keyid is a string (ID_KEY_ID)

```
ORACLE(ike-key-id)# id-type ipv4
```

6. Type **done**to save your configuration.
7. Repeat to configure additional ike-keyid instances.

IKEv2 Interface Configuration

After you configure global Internet Key Exchange (IKE) parameters, use the procedures described in the Security chapter to configure and monitor IKEv2 interfaces.

IKEv2 interface configuration includes the following steps.

1. Configure IKE interface attributes.
2. Configure Security Associations.

3. Configure Security Policies.
4. (Optional) Configure the Dead Peer Detection Protocol.
5. (Optional) Configure the Online Certificate Status Protocol or Certificate Revocation List Support.
6. (Optional) Configure Threshold Crossing Alerts.
7. (Optional) Configure access control allow and block lists.

EAP-based Authentication

RFC 3748, Extensible Authentication Protocol (EAP) describes a flexible and extensible framework that enables authentication services. While the RFC itself describes only a single authentication method, MD5-Challenge, the provided framework support numerous authentication methods.

The current release supports the seven EAP-based authentication methods described in the following sections. Note that for all currently supported EAP authentication methods that the actual authentication is provided by an adjacent RADIUS server. During the EAP-based authentication exchange the SBC functions as a packet relay between the authenticating client(s) and the RADIUS server.

EAP Authentication Methods

EAP supports several authentication methods.

EAP-MD5

EAP-MD5 is based on RFC 1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*. This RFC describes an authentication method that uses an agreed-upon hashing algorithm, a random challenge value, and a shared secret known only to the authenticator and the EAP peer. In the case of EAP-MD5 the hashing algorithm, which produces a 128-bit message-digest or fingerprint, is described in RFC 1321, *The MD5 Message-Digest Algorithm*.

Using EAP-MD5, authentication of the EAP peer is accomplished as follows.

1. The authenticator issues a Challenge packet, which contains, among other fields, an Identifier field that serves to correlate message exchanges, and a Data field that contains an arbitrary challenge string.
2. The peer concatenates the contents of the Identifier field, the shared-secret, and the challenge string. The peer inputs the concatenated string to the MD5 hash function, computes the 128-bit fingerprint, and returns that value to the authenticator in a Response packet.
3. The authenticator performs the same calculation, and compares its results with those reported by the EAP peer.
4. If the fingerprints are identical, the authenticator issues a Success packet; otherwise the authenticator issues a Failure packet.

> **Note:**
>
> EAP-MD5 does not provide for mutual authentication; the authenticator does not authenticate to the EAP peer.

EAP-MSCHAPv2

EAP-MSCHAPv2 is based on RFC 2759, *Microsoft PPP CHAP Extensions, Version 2*. This RFC describes an authentication method that uses a user-name and password model in conjunction with Microsoft encryption routines. Using EAP-MSCHAPv2, mutual authentication of the EAP peer and authenticator is accomplished as follows:

1. The authenticator issues a Challenge packet, which contains, among other fields, an Identifier field that serves to correlate message exchanges, and a Data field that contains an arbitrary 16-octet challenge string.
2. The peer returns a Response packet that includes the user name, a newly-generated 16-octet challenge for the authenticator, and a one-way encryption of the received challenge string, the generated challenge string, the contents of the Identifier field, and the user password.
3. The authenticator performs the same calculation as was performed by the EAP peer, and compares its results with those reported by the peer. If the results are identical, the authenticator issues a Success packet which also contains a one-way encryption of the authenticator-originated challenge string, the peer-originated challenge string, the encrypted string received from the peer in the Response packet, and the user password.
4. The EAP peer performs the same calculation as was performed by the authenticator, and compares its results with those reported by the authenticator. If the results are identical, the peer uses the mutually authenticated connection; otherwise, it drops the connection.

EAP-AKA

The Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) was devised by the 3GPP (3rd Generation Partnership Project), and made available to the Internet community in RFC 4187. EAP-AKA makes use of the Universal Subscriber Identity Module (USIM), an application resident on the smart card inserted in a 3G mobile phone. The USIM has access to authentication data stored on the smart card.

EAP-SIM

The EAP-SIM Protocol specifies an authentication method for GSM (Global System for Mobile Communication) subscribers. GSM is a second generation mobile standard, and still the most widely used. The authentication method is described in RFC 4186, Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identify Modules (EAP-SIM). Originally developed by the 3GPP, the EAP-SIM protocol specifies an EAP method for authentication and session key distribution using the GSM Subscriber Identity Module (SIM), a smart card installed in the GSM phone.

EAP-TLS

EAP-TLS uses a Transport Layer Security (TLS) handshake, encapsulated within the secure tunnel, to mutually authenticate client and server (or an AAA backend) with certificates. The SBC acts in EAP pass-through mode to communicate the EAP-TLS negotiation between the device and the AAA server.

EAP-TTLS

The EAP-TTLS authentication method is useful when there is no certificate-based infrastructure present for the operator to configure a certificate for each device. EAP-TTLS consists of a Tunneled Transport Layer Security (TTLS) handshake phase (similar to EAP-TLS) and a data phase. During the data phase, the client is authenticated to the server (or the client and server are mutually authenticated) using an arbitrary authentication mechanism encapsulated within the secure tunnel. Thus, EAP-TTLS allows legacy password-based

authentication protocols to be used against existing authentication databases, while protecting the security of these legacy protocols against eavesdropping, man-in-the-middle, and other attacks.

EAP-AKA

EAP-AKA' is a small revision to the EAP-AKA (Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement) method. The change is a new key derivation function that binds the keys derived within the method to the name of the access network. The new key derivation mechanism has been defined in the 3rd Generation Partnership Project (3GPP). This feature allows its use in EAP in an interoperable manner. Additionally, EAP-AKA' employs SHA-256 instead of SHA-1 as the Secure Hash Algorithm.

Multiple Authentication

The Oracle Communications Session Border Controller supports multiple authentication exchanges during IKEv2 negotiation. These exchanges are defined in RFC 4739, Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol. Multiple authentication enables the SBC to engage in an initial certificate-based or shared-secret-based authentication with a remote IKEv2 peer (for example, a femtocell), followed by a subsequent EAP-AKA or EAP-SIM authentication of the remote mobile subscriber.

Multiple authentication exchanges require the use of two specific Notify payloads, MULTIPLE_AUTH_SUPPORTED and ANOTHER_AUTH_FOLLOWS (Notify message type s16404 and 16405) defined in Sections 3.1 and 3.2 of RFC 4739.

Message exchange is as follows.

```
Initiator (IKEv2 peer)                                          Responder
1. HDR, SAi1, KEi, Ni --->
2.           <--- HDR, SAr1, KEr, Nr, CERTREQ, N (MULTIPLE_AUTH_SUPPORTED)
3. HDR, { IDi, CERT, CERTREQ, {IDr], AUTH, SAi2, TSi, TS2
   (MULTIPLE_AUTH_SUPPORTED) N (ANOTHER_AUTH_FOLLOWS) } --->
4.                                          <--- HDR, { IDr, CERT, AUTH }
5. HDR, { IDi } --->
6.                                             <--- HDR, { EAP (Request)}
7. HDR, { EAP (Response) } --->
8.                                             <--- HDR, { EAP (Request)}
9. HDR, { EAP (Response) } --->
10.                                            <--- HDR, { EAP (Success)}
11. HDR, { AUTH } --->
12.                                  <--- HDR, { AUTH, SAr2, TSi, TSr }
```

In Step 2 the responder advertises support for multiple authentication via the MULTIPLE_AUTH_SUPPORTED Notification Payload.

In Step 3 the initiator advertises support for multiple authentication and, using the ANOTHER_AUTH_FOLLOWS Notification Payload, signals its readiness for such authentication.

Step 4 completes mutual certificate authentication.

In Step 5 the initiator discloses its identity.

In Step 6 the responder initiates the EAP process

In Steps 7 and 8 the initiator and responder exchange authentication information for the remote peer.

In Steps 9 and 10 the initiator and responder exchange authentication information for the mobile subscriber.

Steps 11 and 12 report successful authentication.

IPv6 Inner Tunnel Address Assignment

The Oracle Communications Session Border Controller supports the assignment of IPv6 inner tunnel addresses utilizing an external RADIUS server as the IPv6 address source. During the EAP authentication of an IPsec host, neither the SBC nor the RADIUS authentication server has any knowledge of the traffic type (IPv4 or IPv6) that the IPsec host intends to transmit through the tunnel. Consequently, the RADIUS authentication server may send both IPv4 and IPv6 attributes in the RADIUS
Access-Accept message, leaving it to the SBC to select the appropriate attribute and ignore the other.

The SBC makes its decision based on the contents of the Configuration Payload received from the IPsec host. If the payload contains an INTERNAL_IP4_ADDRESS attribute, the IPv4 address received in the Access-Accept message is forwarded to the IPsec host. In a similar fashion, if the payload contains an INTERNAL_IP6_ADDRESS attribute, the IPv6 address received in the
Access-Accept message is forwarded to the IPsec host.

Assignment of IPv6 addresses requires support for the following RADIUS attributes:

- Framed-IPv6-Prefix (Type 97) — also used in RADIUS accounting
- Framed-IPv6-Pool (Type 100)

Framed-IPv6-Pool, which can be returned by a RADIUS authentication server in an Access-Accept message, contains the name of an address pool that should be used by the SBC as a source of IPv6 addresses.Use of Framed-IPv6-Pool requires the pre-configuration of the identified address pool on the SBC.

EAP-only Authentication

IKEv2 specifies that Extensible Authentication Protocol (EAP) authentication must be used together with responder authentication based on public key signatures. This is necessary with old EAP methods that provide only unilateral authentication using, for example, one-time passwords or token cards. With EAP-SIM, EAP-AKA, EAP-AKA', EAP-TTLS, and EAP-TLS, which provide mutual authentication and key agreement, extensible responder authentication for IKEv2 based on methods other than public key signatures can be used. This feature causes the SBC to default to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise.

The Extensible Authentication Protocol, defined in RFC3748, is an authentication framework that supports multiple authentication mechanisms. One of the advantages of the EAP architecture is its flexibility. Rather than requiring the authenticator (for example, a wireless LAN access point) to be updated to support each new authentication method, EAP permits the use of a backend authentication server that may implement some or all authentication methods. The SBC uses a backend authentication server (for example, 3GPP AAA) and is in pass-through mode for EAP.

IKEv2 is a component of IPsec used for performing mutual authentication and establishing and maintaining Security Associations (SAs) for IPsec Encapsulating Security Payload (ESP) and Authentication Header (AH). In addition to supporting authentication using public key signatures and shared secrets, IKEv2 also supports EAP authentication. By using EAP, IKEv2 can leverage existing authentication infrastructure and credential databases, such as Home Subscriber Server (HSS), as EAP allows users to choose a method suitable for existing

credentials, and also makes separation of the IKEv2 responder (SBC) from the EAP authentication endpoint (back-end Authentication, Authorization, and Accounting (AAA) server) easier. IKEv2 specifies that these EAP methods must also be used together with responder authentication based on public key signatures. For the public key signature authentication of the SBC to be effective, a deployment of Public Key Infrastructure (PKI) is required, which has to include management of trust anchors on all supplicants. This may not realistic in WiFi calling environments, in which the security of the SBC public key is the same as the security of a self-signed certificate. Mutually authenticating EAP methods alone can provide a sufficient level of security.

Because of these reasons, the SBC now defaults to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise by disabling the new parameter **eap-only-support** in the **ike-interface** configuration element.

EAP-only Authentication Configuration

1. Access the **ike-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

2. Select the **ike-interface** object to edit.

```
ORACLE(ike-interface)# select
<address>:

ORACLE(ike-interface)#
```

3. **eapOnlyAuthSupport** — The default is **enabled**. Set the value to **disabled** to use EAP authentication together with responder authentication based on public key signatures.
4. Type **done** to save your configuration.

Debugging IKEv2 IPsec Tunnel Establishment

The Oracle Communications Session Border Controller provides details of all IKE endpoints that establish IKEv2/IPsec tunnels. Logging can also be enabled by IP address and userid.

In a typical deployment scenario, the IP address can be the public address of a NAT device that communicates with the Oracle Communications Session Border Controller; the user-id can be the user-id of a femtocell or an IKE endpoint residing behind the NAT. The user-id can be an EAP identity exchanged during EAP authentication, or the identity contained in the IDi payload of the initial IKE_AUTH message. Typically the identity in the IDi payload is an IP address, an FQDN, or an address as defined in RFC 822, *Standard for the Format of ARPA Internet Text Messages*.

Enabling/Disabling Targeted Debugging

Targeted debugging is enabled by the **security ike debug-logging peer-ip-userid** ACLI command which takes a single string argument in the form ipAddress:userID. For example:

```
ORACLE# security ike debug-logging peer-ip-userid 172.16.20.1:12EDE12626719
ORACLE#
```

With endpoint-specific logging enabled, the log.iked, log.authd, and log.secured files are populated with data pertinent to the target endpoint only and exclude date for all other endpoints. Logging is based on an exact match of the IP address and user-id provided by the argument string.

> **Note:**
>
> This command is expensive and should be used to debug one or two endpoints at a time. The operating system imposes a hard limit of no more than 5 simultaneous targeted debugging sessions.

Use the no form of the command to stop an existing targeted debugging session

```
ORACLE# security ike debug-logging peer-ip-userid 172.16.20.1:12EDE12626719 no
ORACLE#
```

Use the **show security ike peer-endpoint-logging** ACLI command to display a list of configured debug-logging sessions

```
ORACLE# show security ike peer-endpoint-logging
ORACLE#
IPaddress : Userid
==============
172.16.20.1:12EDE12626719
ORACLE#
```

High Availability Caveat

Since the security ike debug-logging peer-ip-userid command is expensive, this implementation intentionally does NOT synchronize log data on the active and standby HA devices. Consequently, in the event of a switchover from the active to the standby, no log data is available on the newly active device. To enable debug-logging on the new active device, the user should verify tunnel establishment, and then use security ike debug-logging peer-ip-userid command on the currently active member of the HA pair.

Configure an IKEv2 Interface

This section covers IKEv2 global configuration parameters, omitting IKEv1 parameters. You can override global values set in the **ike-config** configuration element with values set at the **ike-interface** level.

1. Access the **ike-interface** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ike
   ORACLE(ike)# ike-interface
   ORACLE(ike-interface)#
   ```

2. **state**—Enable the IKEv2 interface.
3. **ike-version**—Set this attribute to **2**.

4. **address**—Specify the IPv4 or IPv6 address of the interface.

   ```
   ORACLE(ike-interface)# address 10.0.0.10
   ```

5. **realm-id**—Specify the realm that contains the IP address assigned to this IKEv2 interface.

   ```
   ORACLE(ike-interface)# realm-id access-10
   ```

6. **ike-mode**—Specify whether the SBC will act as a responder or initiator.
7. **dpd-params-name**—Enable the Dead Peer Detection Protocol on this IKEv2 interface.

   The protocol is initially enabled by setting a non-zero value to the **dpd-time-interval** parameter during IKEv2 global configuration process. The protocol is enabled at the local level by assigning an existing dpd-params configuration element to this IKEv2 interface.

   Refer to Dead Peer Detection Protocol Configuration in this chapter for information on configuring dpd-params configuration elements.

   ```
   ORACLE(ike-interface)# dpd-params-name ikeDPD
   ```

8. **v2-ike-life-seconds**—(Optional) Specify the lifetime (in seconds) for the IKEv2 SAs supported by this IKEv2 interface.

   The default is 86400 (24 hours).

   - Min: 1
   - Max: 999999999

9. **v2-ipsec-life-seconds**—(Optional) Specify the lifetime (in seconds) for the IPsec SAs supported by this IKEv2 interface.

   The default is 28800 (8 hours).

   - Min: 1
   - Max: 999999999

10. **v2-rekey**—(Optional) Enable or disable the automatic re-keying of expired IKEv2 or IPsec SAs on this IKEv2 interface.

    The default is none.

    - disabled—Disable IKE/IPSEC rekey
    - enabled—Enable IKE/IPSEC rekey
    - none—Use the rekey value from the global ike-config element

    With automatic re-keying enabled, and with the global **dpd-time-interval** parameter set to a non-zero value, the SBC retransmits the re-keying request if it does not receive a response from the remote IPsec peer within the interval specified by the ike-config **dpd-time-interval** parameter. The SBC makes a maximum of three retransmission attempts before abandoning the re-keying effort.

11. **esnsupport**—Enable to support Extended Sequence Number (ESN) per RFC 4304.
12. **shared-password**—If using the shared-password authentication method, set the shared password.
13. **eap-protocol**—(Optional) Set the EAP protocol.

    Available values are:

    - eap-tls

- eap-leap
- eap-sim
- eap-srp
- eap-ttls
- eap-aka
- eap-peap
- eap-mschapv2
- eap-fast
- eap-psk
- eap-radius-passthru

14. **sd-authentication-method**—Select the interface-specific method used by IKEv2 peers to authenticate to each other.
    - shared-password—Use a pre-shared-secret to authenticate the remote IKEv2 peer.
    - certificate—Use an X.509 certificate to authenticate the remote IKEv2 peer.

    > **Note:**
    >
    > **sd-authentication-method** can be safely ignored, if authentication utilizes any of the methods described in EAP-based Authentication.

    ```
    ORACLE(ike-interface)# sd-authentication-method shared-password
    ```

15. **certificate-profile-id-list**—If using the certificate authentication method, identify the **ike-certificate-profile** configuration element that contains identification and validation credentials required for certificate-based IKEv2 authentication.
16. **cert-status-check**—(Optional) Enable certificate status checking using either Online Certificate Status Profile (OCSP) or a local copy of a Certificate Revocation List.

    The default is **disabled**.
17. **cert-status-profile-list**—(Optional) Assign one or more **cert-status-profile** configuration elements to this IKEv2 interface.

    Each assigned cert-status-profile provides the information needed to access either an OCSP responder or a CRL source.

    > **Note:**
    >
    > Use quotation marks to assign multiple OCSP responders.

    ```
    ORACLE(ike-interface)# cert-status-profile-list "VerisignClass3Designate
    Verisign-1 Thawte-1"
    ```

18. **access-control-name**—Specify the ike-access-control list to use on this IKE interface. The list assignment applies the IKEv2 DDOS, allowlist and blocklist protection configured within the ike-access-control object to the interface.

This parameter is meaningful only when authentication uses a RADIUS server to implement the EAP-based authentication, and can otherwise be safely ignored. Allowlists and blocklists specify IMSI prefixes or MAC addresses that are allowed through or denied access to the RADIUS authentication server.

```
ORACLE(ike-interface)# access-control-name allow_01
```

19. **tunnel-orig-name-list**—(Optional) Specify the name the tunnel-origin-params element you want to apply to this IKE interface.
20. Type **done** to save your configuration.
21. Configure additional IKEv2 interfaces if required.

IPsec Security Policy Configuration

You first define ike-sainfo elements that identify cryptographic material available for Security Association negotiation, and then define interface-specific IPsec Security Policies.

IPsec SA Configuration

During the IKE_AUTH exchange, cooperating peers use the secure channel previously established by the IKE_SA_INIT exchange to negotiate child IPsec SAs to construct secure end-to-end IPsec tunnels between the peers. IKE_SA_INIT negotiations use the values provided by the ike-sainfo configuration element.

Use the following procedure to create an ike-sainfo configuration element that specifies cryptographic material used for IPsec tunnel establishment. You will later assign this ike-sainfo configuration element to an IPsec Security Policy which defines IPsec services for a specified IKEv2 interface.

1. Access the **ike-sainfo** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-sainfo
ORACLE(ike-sainfo)#
```

2. **name**—Provide a unique identifier for this ike-sainfo configuration element.

```
ORACLE(ike-sainfo)# name SA-1
```

3. **security-protocol**—Specify the IPsec security (authentication and encryption) protocols supported by this SA.

   The default value is **ah**. Supported values are:

   - ah—Authentication Header. Provides authentication integrity to include the mutual identification of remote peers, non-repudiation of received traffic, detection of data that has been altered in transit, and detection of data that has been replayed, that is copied and then re-injected into the data stream at a later time.
   - esp—Encapsulating Security Payload provides both authentication and privacy services.
   - esp-auth—Supports ESP's optional authentication

4. **auth-algo**—Specify the authentication algorithms supported by this SA.

Available protocols are:

- any
- aes-xcbc
- sha2-256
- sha2-384
- sha2-512

5. **encryption-algo**—Specify the encryption algorithms supported by this SA.

   The default is **aes**. Available protocols are:

   - any—Choose any
   - aes—AES with CBC mode
   - aes-ctr—AES with counter mode

6. **ipsec-mode**—Specify the IPsec operational mode.

   - tunnel—Provides a secure end-to-end connection between two IP hosts.
   - transport—Provides VPN service where the entire IP packets are encapsulated within an outer IP envelope and delivered from source (an IP host) to destination (generally a secure gateway) across an untrusted internet.

7. **tunnel-local-addr**—If using tunnel mode, specify the IP address of the local IKEv2 interface that terminates the IPsec tunnel.

   ```
   ORACLE(ike-sainfo)# tunnel-local-addr 172.30.89.10
   ```

8. **tunnel-remote-addr**—If using tunnel mode, specify the IP address of the remote IKEv2 peer that terminates the IPsec tunnel.

   Provide the remote IP address or use the default wild-card value (*) to match all IP addresses.

   ```
   ORACLE(ike-sainfo)# tunnel-remote-addr *
   ```

9. **local-id-profile**—Enter the name of the applicable local **ike-key-id** element for this **ike-sainfo** element. That **ike-key-id** element contains information needed to associate the SBC asserted identity with a PSK.
10. **remote-id-profile**—Enter the name of the applicable remote **ike-key-id** element for this **ike-sainfo** element. That **ike-key-id** element contains information needed to associate the remote station's asserted identity with a PSK.
11. Type **done** to save your configuration.
12. If necessary, configure additional IPsec SAs.

Security Policy Configuration

Use the following procedure to define an IPsec Security Policy.

1. Access the **security-policy** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ipsec
   ```

```
ORACLE(ipsec)# security-policy
ORACLE(security-policy)#
```

2. **name**—Identify this IPsec Security Policy.

```
ORACLE(security-policy)# name requireIPsec
```

3. **network-interface**—Provide the network interface name of the IKEv2 interface to which this security policy is applied.

```
ORACLE(security-policy)# network-interface M00:0
```

4. **priority**—(Optional) Assign a priority to this IPsec Security Policy.
   - Highest priority: 0
   - Lowest priority: 3071
5. **action**—Specify the processing of IPsec and non-IPsec traffic streams.
   - allow—Process non-IPsec traffic
   - ipsec—Allow only IPsec traffic
   - srtp—Allow only SRTP traffic
   - srtcp—Allow only SRTCP traffic
6. **direction**—Identity the traffic streams subject to the processing specified by the **action** parameter.

   Available values are:
   - in
   - out
   - both
7. **local-ip-addr-match**—(Optional) Specify the local IP address of the network interface.

   Provide the local IP address or retain the default value, 0.0.0.0, which matches all local IP addresses.

```
ORACLE(security-policy)# local-ip-addr-match 172.30.89.10
```

8. **remote-ip-addr-match**—(Optional) Specify the IP address of the remote IKEv2 peer.

   Provide the remote IP address or retain the default value, 0.0.0.0, which matches all remote IP addresses.

```
ORACLE(security-policy)# remote-ip-addr-match 0.0.0.0
```

9. **local-port-match**—(Optional) Specify the local ports to which this IPsec Security applies.

   Use 0 to specify all local ports.
   - Min: 1
   - Max: 65535
10. **local-port-match-max**—(Optional) Specify the maximum value for the local port to which this IPsec Security applies.
    - Default: 65535

- Min: 0
- Max: 65535

11. **remote-port-match**—(Optional) Specify the remote ports to which IPsec Security Policy applies.

    Use 0 to specify all remote ports.

    - Min: 1
    - Max: 65535

12. **remote-port-match-max**—(Optional) Specify the maximum value for the remote port to which this IPsec Security applies.

    - Default: 65535
    - Min: 0
    - Max: 65535

13. **ike-sainfo-name**—Assign an IPsec data SA to this Security Policy.
14. Type **done** to save your configuration.

> **Note:**
>
> Oracle recommends you configure the local-port-match-max or remote-port-match-max value under security-policy to allow processing of IPsec or non-IPsec traffic streams, based on the action configured.

Enable Tunnel Pass-Through

Use IPsec Security Policies to enable tunnel pass-through.

Pass-through IPv4 traffic via an IPv4 tunnel

1. Configure IPv4 allow policy for IKE protocol traffic
2. Configure IPv4 ipsec policy for media traffic
3. Configure the IKEv2 IPv4 interface with an IPv4 local address pool, or
4. Configure the RADIUS server to return a Framed-IP-Address and/or Framed-IP-Netmask attribute

Pass-through IPv6 traffic via an IPv6 tunnel

1. Configure IPv6 allow policy for IKE protocol traffic
2. Configure IPv6 ipsec policy for media traffic
3. Configure the IKEv2 IPv6 interface with an IPv6local address pool, or
4. Configure the RADIUS server to return a Framed-IPv6-Prefix or Framed-IPv6-Pool attribute

Pass-through IPv4 traffic via an IPv6 tunnel

1. Configure IPv6 allow policy for IKE protocol traffic
2. Configure IPv4 ipsec policy for media traffic

3. Configure the IKEv2 IPv6 interface with an IPv4 local address pool, or
4. Configure the RADIUS server to return a Framed-IP Address and/or Framed-IP-Netmask attribute

Pass-through IPv6 traffic via an IPv4 tunnel

1. Configure IPv4 allow policy for IKE protocol traffic
2. Configure IPv6 ipsec policy for media traffic
3. Configure the IKEv2 IPv4 interface with an IPv6local address pool, or
4. Configure the RADIUS server to return a Framed-IPv6-Prefix or Framed-IPv6-Pool attribute

IPSec SA Rekey on Sequence Number Overflow

The Oracle Communications Session Border Controller establishes a new IPSec security association (SA) when the counter for the outbound 32-bit Sequence Number (SN) or the 64-bit Extended Sequence Number (ESN) overflows.

The SN or ESN counter is incremented for every outbound packet. These counters can overflow when the SBC is handling packet intensive services such as video streaming or long duration calls. In accordance with RFCs 4303 and 7296, the SBC establishes new security associations, as part of rekeying, before the SN or ESN counters can roll over. It does this through the use of two parameters in the **ipsec-global-config** configuration element: **rekey-on-sn-overflow**, the default for which is **enabled**, and **sn-rekey-threshold**, which identifies the threshold for rekeying security associations as a percentage of the counter capacity and for which the default is **95**.

There are four ACLI commands you can use to monitor SN and ESN counter overflows:

show datapath etc-stats ppms ipsec

Issuing this command shows, along with other existing IPSec PPM-related statistics, the total number of times SN overflow occurred. The four pertinent parameters are:

- **ob-sn-threshold-overflows** — This counter is incremented when the SN for an outbound SA for a tunnel exceeds the user-configured threshold value.
- **ob-sn-32bit-overflows** — This counter is incremented when the lower 32-bits of the outbound ESN (when ESN is enabled) overflows.
- **standby-ob-sn-overflows** — This counter is incremented when the SN or ESN for an outbound SA for a tunnel overflows the threshold value installed on the standby node during SA installation or update on the standby system.
- **ib-sn-32bit-overflows** — This counter is incremented when the lower 32 bits of the inbound ESN (when ESN is enabled) overflows.

show datapath netlink show

Issuing this command shows the total number of SN overflow notifications received by the netlink layer on the host processor. The four newly-added parameters are the same as those in **show datapath etc-stats ppms ipsec**.

show sa stats ike

Issuing this command shows the number of times an SN overflow triggered a request for an IPsec rekey to acquire a new SA, as well as the number of times rekey requests succeeded and failed.

show security ike statistics

Issuing this command shows, with the parameter **RekeyOnSNoverflow** the number of times an SN overflow triggered an IPsec rekey.

IPSec SA Rekey on Sequence Number Overflow Configuration

1. Access the **ipsec-global-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# ipsec-global-config
ORACLE(ipsec-global-config)#
```

2. Select the **ipsec-global-config** object to edit.

```
ORACLE(ipsec-global-config)# select
ORACLE( ipsec-global-config)#
```

3. **rekey-on-sn-overflow** — Identifies whether to enable IPSec rekey on sequence number (SN) or extended sequence number (ESN) overflow. Rekey initiation is independent of the value of the parameter **v2-rekey** in the **ike-interface** configuration element. Allowable values are **enabled** and **disabled**. The default is **enabled**.
4. **sn-rekey-threshold** — Identifies the threshold for triggering an IPSec security association (SA) rekey on SN or ESN overflow as a percentage of the SN (32-bit) or ESN (64-bit) number space. The allowable range is **80** to **100** and the default is **95**.
5. Type **done** to save your configuration.

Pre-Populated ARP Table

In certain topologies remote IPsec endpoints can require access to core network hosts reachable through a Oracle Communications Session Border Controller core interface. In these instances, the SBC receives the tunneled packet, and masks the received IP destination address against its own local addresses to determines if direct delivery is possible. If so, the SBC issues an ARP request to obtain the physical destination address.

This process can be expedited by pre-populating the interface-specific ARP table with a list of commonly accessed core network host reachable by that interface. With the ARP table pre-populated with IP addresses, the ARP process issues ARP requests at 5 second intervals until a response is received. Once the pre-populated IP address has been resolved, periodic ARP refreshes are used to maintain the currency of the resolution.

Pre-Populate An Interface-Specific ARP Table

1. Access the **network-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-interface
ORACLE(network-interface)
```

2. Select the **network-interface** object to edit.

```
ORACLE(network-interface)# select
<name>:<sub-port-id>:
1: wancom0:0 ip=10.0.0.2 gw=10.0.4.1

selection: 1
ORACLE(network-interface)#
```

3. **add-neighbor-ip**—Add the initial IP address to the core-interface-specific ARP table.

```
ORACLE(network-interface)# add-neighbor-ip 10.0.0.101
```

4. If necessary, add an additional IP address to the core-interface-specific ARP table.

   You can add a maximum of ten IP addresses to a single network interface.

5. Use the **show** command to examine the pre-populated ARP table, referred to as the neighbor list.

```
ORACLE(network-interface)# show
network-interface
        ...
        neighbor-list                   10.0.0.101
                                        10.0.0.102
                                        10.0.0.103
                                        10.0.0.104
                                        10.0.0.105
                                        10.0.0.106
                                        10.0.0.107
                                        10.0.0.108
                                        10.0.0.109
                                        10.0.0.110
        ...
```

6. Type **done** to save your configuration.

Configure Dead Peer Detection

Dead Peer Detection is enabled by setting the dpd-time-interval parameter to a non-zero value. DPD exchanges are asynchronous, consisting of a simple R-U-THERE and an ACK.

1. Access the **dpd-params** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# dpd-params
ORACLE(dpd-params)#
```

2. **name**—Provide a unique identifier for this dpd-params instance.

```
ORACLE(dpd-params)# name ikeDPD
```

3. **max-loop**—Specify the maximum number DPD peers whose liveliness is examined every **dpd-interval** period.

   Periodic liveliness is tested by the Oracle Communications Session Border Controller issuing an R-U-THERE message to each peer in the current group. If the peer acknowledges receipt of the message, it is confirmed as alive. If the peer fails to respond, its status is determined by the max-retrans and max-attempts parameter values.

   - Min: 1
   - Max: 999999999

4. **max-retrans**—Specify the maximum number of times that the SBC, acting as a DPD initiator, retransmits an unacknowledged R-U-THERE message while performing periodic liveliness tests.

   The default is 3.

   - Min: 1
   - Max: 4

5. **max-attempts**—Specify the number of failed liveliness tests required to declare a peer as dead and take down the IKE tunnel.

   The default is 1.

   - Min: 1
   - Max: 4

6. **max-endpoints**—Specify the maximum number of simultaneous DPD protocol negotiations supported when the CPU is not under load, as specified by **max-cpu-limit**.

   The default is 25.

   - Min: 1
   - Max: 15000

   If CPU workload surpasses the threshold set by max-cpu-limit, this value is over-ridden by load-max-endpoints.

7. **max-cpu-limit**—Specify a threshold value (expressed as a percentage of CPU capacity) at which DPD protocol operations are minimized to conserve CPU resources.

   The default is 60.

   - Min: 0
   - Max: 100

8. **load-max-loop**—Specify the maximum number of endpoints examined every **dpd-time-interval** when the CPU is under load, as specified by **max-cpu-limit**.

   The default is 40.

   - Min: 1
   - Max: 999999999

   Ensure that the configured value is less than the value assigned to **max-loop**.

9. **load-max-endpoints**—Specify the maximum number of simultaneous DPD Protocol negotiations supported when the CPU is under load, as specified by **max-cpu-limit**.

   The default is 5.

   - Min: 1
   - Max: 15000

Ensure that the configured value is less than the value assigned to **max-endpoints**.

10. Type **done** to save your configuration.
11. If necessary, configure additional dpd-params configuration elements.
12. Access the **ike-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

13. **dpd-params-name**—Enable Dead Peer Detection on this IKEv2 interface.

```
ORACLE(ike-interface)# dpd-params-name ikeDPD
```

14. Type **done** to save your configuration.

Certificate Revocation Lists

A Certificate Revocation List (CRL) contains a list of the serial numbers of certificates that have been revoked by the issuing Certification Authority (CA). Such issuing authorities update CRLs periodically, and make the updates lists available to subscribers. CRL updates can be deliver in either PEM (Privacy Enhanced Email) or DER (Distinguished Encoding Rules) format. PEM is base-64 encoded ASCII that provides BEGIN CERTIFICATE and END CERTIFICATE statements; DIR is a binary rendering of the PEM format. Both formats (PEM and DIR) are supported by the Oracle Communications Session Border Controller.

When authentication of remote IKEv2 peers is certificate-based, you can enable CRL usage on IKEv2 interfaces to verify certificate status.

CRL-Based Certificate Verification

This section provides instruction on using the ACLI to configure periodic retrieval of CRLs.

Configuration of CRL-based certificate verification is a three-step process.

1. Specify the information and cryptological resources required to access one or more CRL sources.
2. If not already done, enable CRL usage on an IKEv2 interface.
3. Associate one or more CRLs with an IKEv2 interface.

Configure CRL Certificate Verification

The **cert-status-profile** element is a container for the information required to access a specific CRL source.

1. Access the **cert-status-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# cert-status-profile
ORACLE(cert-status-profile)#
```

2. **name**—Provide a unique name for this profile.
3. **type**—Select the certificate revocation check method.

Available values are:

- OCSP
- CRL

4. Specify either the IP address or the hostname of the CRL source.
   - **ip-address**—Specify the IP address of the CRL source.
   - **host-name**—Specify the hostname of the CRL source

> **Note:**
>
> If values are provided for both attributes, the SBC uses the IP address and ignores the **host-name** value.

5. **crl-list**—Specify the source filepath(s) to one or more requested CRLs.

   For example:

```
ORACLE(cert-status-profile)# crl-list /crl/v2/tc_class_3_ca_II.crl
```

6. **realm-id**—Specifies the realm used to request and receive CRLs.

   In the absence of an explicitly configured value, the SBC provides a default value of wancom0, specifying CRL-related transmissions across the wancom0 management interface.

> **Note:**
>
> If the CRL source is identified by its FQDN, the realm identified by **realm-id** must be DNS-enabled.

7. **responder-cert**—Identify the certificate used to validate the received CRL (the public key of the CRL source).

   Provide the name of the certificate configuration element that contains the certificate used to validate the signed CRL.

8. **retry-count**—Specify the maximum number of times to retry an CRL source in the event of connection failure.

   The default is 1.
   - Min: 0
   - Max: 10

9. **dead-time**—Specify the quarantine period imposed on an unavailable CRL source.

   The default is 0.
   - Min: 0
   - Max: 3600

10. **crl-update-interval**—Specify the interim in seconds between CRL updates.

    The default is 86400.
    - Min: 600

- Max: 2600000

CRLs are stored in the /code/crls directory. Outdated, invalid CRLs are over-written with the each newly-obtained current CRL.

11. Type **done** to save your configuration.
12. If necessary, configure additional **cert-status-profile** configuration elements.

SNMP Traps

An SNMP trap is thrown, and a major alarm generated, if the Oracle Communications Session Border Controller is unable to retrieve a CRL from the server. This trap includes the server's FQDN, assuming that the FQDN has been identified during the configuration process, the server's IP address, the reason for the failure, and the time of the last successful CRL retrieval, with the time expressed as the number of seconds since midnight January 1, 1970.

A second SNMP trap is thrown when the SBC successfully retrieves a CRL. This trap includes the server's FQDN, assuming that the FQDN has been identified during the configuration process, and the server's IP address. The issue of this trap also clears any associated major alarm.

Configuring Manual CRL Updates

The ACLI provides the ability to perform an immediate manual refresh of one or more CRLs.

Use the following command to refresh a single CRL.

```
ORACLE# load-crl local-file <fileName>
```

where <fileName> is a remote filepath specified by the crl-list attribute.

Use the following command to refresh all CRLs.

```
ORACLE# load-crl local-file all
```

Use the following command to refresh all CRLs from a specific CRL source.

```
ORACLE# load-crl cert-status-profile <certStatusProfileName>
```

where <certStatusProfileName> references the certificate-status-profile configuration element that contains the CRL source IP address or FQDN.

Use the following command to refresh all CRLs.

```
ORACLE# load-crl cert-status-profile all
```

Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) enables users to determine the revocation state of a specific certificate. Because OCSP ensures access to the freshest CRL, it can provide a more timely source of revocation information than is possible with dynamically or manually loaded CRLs. Guaranteed access to the most recent CRL, however, comes at the expense of increased traffic: a single request/response exchange for each revocation check.

If the OCSP responder returns a status of good, the certificate is accepted and authentication succeeds. If the OCSP responder returns a status other than good, the certificate is rejected and authentication fails.

Certificate status is reported as

- good—which indicates a positive response to the status inquiry. At a minimum, a positive response indicates that the certificate is not revoked, but does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval.
- revoked—which indicates a negative response to the status inquiry. The certificate has been revoked, either permanently or temporarily.
- unknown—which indicates a negative response to the status inquiry. The responder cannot identify the certificate.

When authentication of remote IKEv2 peers is certificate-based, you can enable OCSP on IKEv2 interfaces to verify certificate status.

OCSP-Based Certificate Verification

The following sections provides instruction on using the ACLI to configure OCSP-based certificate verification.

Configuration of OCSP-based certificate verification is a three-step process.

1. Specify the information and cryptological resources required to access one or more OSCP responders.
2. Enable OCSP on an IKEv2 interface.
3. Associate one or more OCSP responders with an IKEv2 interface.

Configure OCSP Certificate Verification

1. Access the **cert-status-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# cert-status-profile
ORACLE(cert-status-profile)#
```

2. **name**—Provide a unique name for this profile.
3. Specify either the IP address or the hostname of the CRL source.
   - **ip-address**—Specify the IP address of the CRL source.
   - **host-name**—Specify the hostname of the CRL source

> **Note:**
>
> If values are provided for both attributes, the SBC uses the IP address and ignores the **host-name** value.

4. **port**—Identifies the port monitored by the HTTP server for incoming OCSP requests.

   The **port** attribute can be safely ignored if the SBC is specified as a FQDN by the **host-name** attribute, but is required if the SBC is identified by the **ip-address** attribute.
5. **type**—Select the certificate revocation check method.

   Available values are:
   - OCSP
   - CRL

6. **trans-protocol**—Retain the default value (http) for **trans-protocol** attribute, which identifies the transport method used to access the SBC.
7. **requester-cert**—Specify the certificate used to sign requests.

   Ignore this attribute if requests are not signed. If a signed request is required by the OCSP responder, provide the name of the certificate configuration element that contains the certificate used to sign OCSP requests.
8. **responder-cert**—Identifies the certificate used to validate signed OCSP response (a public key of the OCSP responder).

   > **Note:**
   >
   > RFC 2560 requires that all OCSP responders digitally sign OCSP responses, and that OCSP requesters validate incoming signatures.

9. **trusted-cas**—A list of certificate configuration objects that identifies the approved DoD-approved CAs that sign SBC certificates.

   In DISA/DoD environments that support the delegated trust model, you must provide a list of CAs used to validate the received certificate.

   If a certificate or a certificate chain is appended to the OCSP response, the OCSP client verifies that the first certificate signed the response, and that the CA is trusted by the SBC (that is, the CA certificate is contained in the **trusted-cas** list. The client then walks through each additional certificate (if any exist) ensuring that each certificate is also trusted. If all certificates are trusted, the OCSP response is accepted; otherwise, it is rejected.
10. **realm-id**—Specify the realm used to transmit OCSP requests and receive OCSP responses.

    In the absence of an explicitly configured value, the SBC provides a default value of wancom0, specifying OCSP protocol transmissions across the wancom0 management interface.
11. **retry-count**—Specify the maximum number of times to retry an CRL source in the event of connection failure.

    The default is 1.

    - Min: 0
    - Max: 10
12. **dead-time**—Specify the quarantine period imposed on an unavailable CRL source.

    The default is 0.

    - Min: 0
    - Max: 3600
13. Type **done** to save your configuration.
14. If necessary, configure additional **cert-status-profile** configuration elements.

SNMP Traps

An SNMP trap is thrown if a configured OSCP responder becomes unreachable.

A second SNMP trap is thrown when connectivity is re-established with a previously unreachable OCSP responder.

Enable Certificate Verification on an IKEv2 Interface

1. Access the **ike-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

2. Select the **ike-interface** object to edit.

```
ORACLE(ike-interface)# select
<address>:

ORACLE(ike-interface)#
```

3. **cert-status-check**—Enable certificate status checking on this IKEv2 interface.
4. **cert-status-profile-list**—Assign a CRL source or sources to the IKEv2 interface

> **Note:**
>
> Use quotation marks to assign multiple CRL sources.

```
ORACLE(ike-interface)# cert-status-profile-list "CRL1-VS CRL2-VS CRL3-VS"
ORACLE(ike-interface)#
```

5. Type **done** to save your configuration.

Threat Protection

Internet Key Exchange (IKE) v2 threat protection is composed of:

- IKEv2 DDoS control
- Allowlist and Blocklist access controls
- IP-header based firewalls

IKEv2-Based DDoS Attacks

Given its usual location at the network edge, and the two stage negotiation process required for the establishment of IPsec tunnels, the Oracle Communications Session Border Controller can be a target of IKEv2-based DDoS (distributed denial of service) attacks. Such attacks, which seek to overwhelm or monopolize system resources to the detriment of the gateway's functionality, can take several forms including:

- prolonging/failing to complete negotiation of the IKEv2 Security Association (SA)
- prolonging/failing to complete negotiation of the IPsec SA
- excessive renegotiation/rekeying of an established IKEv2 SA
- excessive renegotiation/rekeying of an established IPsec SA

- sabotaging the IKEv2 negotiation by failing to present a valid cookie when required to do so
- sabotaging the IKEv2 negotiation by failing to present valid credentials when required to do so

The SBC provides protection against DDoS attacks by monitoring IKEv2 signaling traffic from remote endpoints (defined by an IP address:UDP port pair). All endpoints start in the allowed state, meaning that IKEv2 signaling received from the endpoint is accepted as valid by the SBC. A group of policing parameters, and associated counters, provide protection against DDoS attacks by monitoring IKEv2 signaling from individual endpoints. The SBC maintains a set of counters for each endpoint. The counters record instances of suspect traffic, which may indicate malicious intent, and periodically compare endpoint-specific traffic counts to threshold values established by the policing parameters. If endpoint counts meet or exceed threshold values, the SBC places the endpoint in the deny state, and, if they exist, tears down the IKEv2 SA and IPsec SA associated with the endpoint. While in the deny state, the endpoint is quarantined and refused all access to the IKEv2 interface. The endpoint remains quarantined until the expiration of a pre-set timer. At timer expiration, the endpoint is transitioned to the allowed state, and granted IKEv2 interface access.

Configuration of IKEv2 DDOS protection consists of the following steps.

1. Configure one or more IKEv2 Access Control Templates.
   An IKEv2 Access Control Template enables protection against a DDOS attack, and provides a set of configurable timers and policing parameters used to monitor and squelch suspect IKEv2 traffic.

   Two parameters set user-configurable timers; **tolerance-window** sets the interval between periodic checks of suspect traffic counts, and **deny-period** specifies the duration of the deny state.

   Additional parameters (**pre-ipsec-invalid-threshold**, **pre-ipsec-maximum-threshold**, **invalid-cookie-threshold**, **post-ipsec-invalid-threshold**, **pre-ipsec-maximum-threshold**, and **auth-failure-threshold**) set threshold counts for suspect traffic that may be malicious in nature.
2. Assign a template to an IKEv2 interface.
   Assignment of a template to an IKEv2 interface enables protection against a DDOS attack on that specific interface.

Constructing an IKEv2 Access Control Template

Use the following procedure to construct an IKEv2 Access Control Template.

1. From superuser mode, use the following command sequence to access **ike-access-control** configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-access-control
ORACLE(ike-access-control)#
```

2. Use the required **name** parameter to assign a unique name to this IKEv2 Access Control Template instance.

You will use this name as an identifier when assigning the template to a specific IKEv2 interface.

```
ORACLE(ike-access-control)# name ikev2-ddos-1
ORACLE(ike-access-control)#
```

3. Use the **state** parameter to enable or disable this template instance.

   Supported values are enabled (the default) and disabled.

   ```
   ORACLE(ike-access-control)# state enabled
   ORACLE(ike-access-control)#
   ```

4. Use the **tolerance-window** parameter to specify the interval (in seconds) between checks of endpoint-specific traffic counters.

   At the specified interval, the Oracle Communications Session Border Controller checks the value of each of the counters associated with one of the policing parameters. If the counter value is less than the threshold value set by the policing parameter, the counter is cleared, and the endpoint remains in the allowed state. If the counter value is equal to or greater than the threshold value, the counter is cleared, and the endpoint is placed in the deny state.

   tolerance-window and **deny-period** must both be set to non-zero values to enable IKEv2 DDOS protection.

   Supported values are integers within the range 0, the default, through 999999999 (packets). The default value, 0, specifies that no IKEv2 DDOS protection is enabled.

   ```
   ORACLE(ike-access-control)# tolerance-window 100
   ORACLE(ike-access-control)#
   ```

5. Use the **deny-period** parameter to specify the quarantine period imposed on an endpoint that transitions to the deny state. During the quarantine period, the endpoint is denied all access to the IKEv2 interface.

   deny-period and **tolerance-window** must both be set to non-zero values to enable IKEv2 DDOS protection.

   Supported values are integers within the range 0 through 999999999, with a default value of 30 (seconds).

   ```
   ORACLE(ike-access-control)# deny-period 50
   ORACLE(ike-access-control)#
   ```

6. Use the **pre-ipsec-invalid-threshold** parameter to enable protection against a DDOS attack that sends malformed, or otherwise invalid, packets during the IKEv2 SA negotiation process.

   pre-ipsec-invalid-threshold specifies the maximum number of malformed IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to consume system resources in a futile effort to complete negotiation of IKEv2 SAs.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

Supported values are integers within the range 0, the default, through 999999999 (packets). The default value, 0, specifies that protection against malformed or invalid IKEv2 SA negotiation packets is not enabled.

```
ORACLE(ike-access-control)# pre-ipsec-invalid-threshold 10
ORACLE(ike-access-control)#
```

7. Use the **pre-ipsec-maximum-threshold** parameter to enable protection against an IKEv2 DDOS attack that sends excessive packets during the IKEv2 SA negotiation process.

   pre-ipsec-maximum-threshold specifies the maximum number of valid IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to prolong the IKEv2 negotiation by persistently renegotiating the IKEv2 SA.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

   Supported values are integers within the range 0, the default, through 999999999 (packets). The default value, 0, specifies that protection against valid, but excessive, IKEv2 SA negotiation packets is not enabled.

   ```
   ORACLE(ike-access-control)# pre-ipsec-maximum-threshold 100
   ORACLE(ike-access-control)#
   ```

8. Use the **invalid-cookie-threshold** parameter to enable protection against an IKEv2 DDOS attack that prolongs the IKEv2 SA negotiation process by having the endpoint deliberately fail to follow required protocol behavior, as defined in Section 2.6 of RFC 4306, Internet Key Exchange (IKEv2) Protocol.

   During the IKEv2 negotiation process, the SBC can issue an IKE_SA_INIT response that contains a cookie notification payload. The payload mandates that the endpoint retry IKEv2 SA negotiation with the cookie value as the first payload in its response to the IKE_SA_INIT.

   invalid-cookie-threshold specifies the maximum number of packets containing an erroneous cookie value tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

   Supported values are integers within the range 0, the default, through 999999999 (packets). The default value, 0, specifies that protection against erroneous cookie responses is not enabled.

   ```
   ORACLE(ike-access-control)# invalid-cookie-threshold 10
   ORACLE(ike-access-control)#
   ```

9. Use the **after-ipsec-invalid-threshold** parameter to enable protection against an IKEv2 DDOS attack that sends malformed IKEv2 SA packets after the establishment of an IPsec tunnel.

   after-ipsec-invalid-threshold specifies the maximum number of malformed, or otherwise invalid, IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to consume system resources in a futile effort to renegotiate the IKEv2 SA.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

Supported values are integers within the range 0, the default, through 999999999 (packets). The default value, 0, specifies that protection against malformed or invalid IKEv2 SA renegotiation packets is not enabled.

```
ORACLE(ike-access-control)# post-ipsec-invalid-threshold 10
ORACLE(ike-access-control)#
```

10. Use the **after-ipsec-maximum-threshold** parameter to enable protection against an IKEv2 DDOS attack that sends valid, but excessive, IKEv2 SA packets after the establishment of an IPsec tunnel.

    after-ipsec-maximum-threshold specifies the maximum number of valid IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to consume system resources by persistently renegotiating the IKEv2 SA.

    If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

    Supported values are integers within the range 0, the default, through 999999999 (packets). The default value, 0, specifies that protection against valid, but excessive, IKEv2 SA negotiation packets is not enabled.

    ```
    ORACLE(ike-access-control)# post-ipsec-maximum-threshold 1000
    ORACLE(ike-access-control)#
    ```

11. Use the **auth-failure-threshold** parameter in conjunction with **auth-failure-threshold-report** to enable protection against an IKEv2 DDOS attack that attempts to consume system resources by overwhelming the authentication function.

    auth-failure-threshold specifies the maximum number of failed authentication attempts tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks attempt to consume system resources by persistently presenting invalid credentials during the endpoint authentication process.

    If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

    Supported values are integers within the range 0, the default, through 999999999 (authentication attempts). The default value, 0, specifies that protection against invalid authentications is not enabled.

    ```
    ORACLE(ike-access-control)# auth-failure-threshold 10
    ORACLE(ike-access-control)#
    ```

12. Use the **auth-failure-threshold-report** parameter in conjunction with **auth-failure-threshold** to enable protection against an IKEv2 DDOS attack that attempts to consume system resources by overwhelming the authentication function.

    auth-failure-threshold-report specifies how failed authentications are reported. Supported values are:

    - no-reporting—(the default), authentication failures are not reported
    - snmp-trap-only—authentication failures are reported by generating an SNMP trap (refer to SNMP Trap for information of trap structure)
    - syslog-only—authentication failures are reported by sending a syslog message

- snmp-trap-and-syslog—authentication failures are reported with both an SNMP trap and a syslog message

```
ORACLE(ike-access-control)# auth-failure-threshold-report snmp-trap-only
ORACLE(ike-access-control)#
```

13. Use **done**, **exit**, and **verify-config** to complete configuration of the IKEv2 Access Control Template.
14. Repeat Steps 1 through 13 to complete additional IKEv2 Access Control templates if required.

Assigning an Access Control Template to an IKEv2 Interface

Use the following procedure to assign an IKEv2 Access Control Template to an IKEv2 interface. The template assignment enables IKEv2 DDOS protection on the interface.

1. From superuser mode, use the following command sequence to access ike-interface configuration mode

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

2. Use the **select** command to specify the interface to which the IKEv2 Access Control Template will be assigned.

```
ORACLE(ike-interface)# select
<address>:
172.30.1.150
172.30.1.151
172.30.55.127
selection: 1
ORACLE(ike-interface)#
```

3. Use the **access-control-name** parameter to assign an IKEv2 Access Control Template to the interface.

```
ORACLE(ike-interface)# access-control-name ikev2-ddos-1
ORACLE(ike-interface)#
```

4. Use **done**, **exit**, and **verify-config** to complete IKEv2 Access Control Template assignment.

SNMP Trap

Violation of the authenticate failure threshold can generate an SNMP trap that includes the endpoint's ID or MSISDN (Mobile Station International Subscriber Directory Number), its IP address, and port number.

High Availability

IKE counters that track suspected IKEv2 DDOS attacks are not synchronized with the standby Oracle Communications Session Border Controller. Endpoint deny status, however, is synchronized with the standby.

Support for Allowlists and Blocklists with Access Control

The SBC supports Internet Key Exchange (IKE) v2 access-control allow lists that permit authentication only for a provisioned list of IMSI prefixes or MAC addresses. The SBC also supports block lists that deny authentication to a provisioned list of IMSI prefixes or MAC addresses.

Allowlist Configuration

Use the procedures described in "Threat Protection" only when the EAP-SIM protocol performs authentication. You can disregard this section when the Oracle Communications Session Border Controller uses any other authentication method.

EAP-SIM Protocol Overview

The EAP-SIM Protocol is described in RFC 4186, Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identify Modules (EAP-SIM). Originally developed by the 3GPP (3rd Generation Partnership Project), the EAP-SIM protocol provides for mutual authentication between the authenticator (a RADIUS server) and a GSM subscriber.

Within the EAP-SIM framework the GSM subscriber identifies itself with its International Mobile Subscriber Identity (IMSI), a digit string providing a globally unique identity for the subscriber's device. The IMSI is stored on a Subscriber Identity Module (SIM) installed in the GSM phone.

The IMSI is usually a 15-digit string that takes the following form:

```
<MCC><MNC><MSIN>
```

- MCC (Mobile Country Code) prefix — 3 digits that uniquely identify the carrier's residence, not the subscriber's current location
- MNC (Mobile Network Code) prefix — 2 or 3 digits that identify the carrier (the concatenation of the MCC and MNC prefixes provide unambiguous identification of the carrier network)
- MSIN (Mobile Station Identification Number) — the remaining digits identify the specific device within the carrier's network

IMSI and MAC Filtering

With EAP-SIM protocol in use, authentication is accomplished by a RADIUS server. Using the Wm interface, the Oracle Communications Session Border Controller passes the received IMSI identity to the RADIUS server. In order to minimize server processing, the SBC provides users with the optional ability to compile IMSI prefix allowlists that filter identities presented for RADIUS authentication. Allowlists are inclusive in that only those identities matching list contents are granted RADIUS access; non-matching identities are summarily rejected by the SBC. The allowlists contain numeric strings or simple regular expressions that identify blocks of subscribers eligible for access to the RADIUS server.

These strings are interpreted as either an IMSI prefix or as a MAC address. Allowists now contain either IMSI or MAC identifiers. Identifiers are constructed using the digits 0 through 9 , any hexadecimal digit, and the ^ wild-card character, which specifies any single base-10 or base-16 digit. Each identifier supports one or more subscribers eligible for authentication.

Sample identifiers are as follows:

- 744 matches the country of Paraguay
- 74401 matches a specific Paraguayan carrier (Hola Paraguay S.A.)
- 7440^ matches all current Paraguayan carriers (74401, 74402, 74404, and 74405)

Configure IMSI and MAC Allowlists

The ike-access-control configuration element defines an allowlist that filters Intenrational Mobile Subscriber Identity (IMSI) or Media Access Control (MAC) identities presented by remote endpoints during the authentication process. Only those identities matching the literal or regular expressions contained in the allowlist are forwarded through the Wm interface to a RADIUS server for authentication.

1. Access the **ike-access-control** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-access-control
ORACLE(ike-access-control)#
```

2. **name**—Provide a unique identifier.

```
ORACLE(ike-access-control)# name allow_01
```

3. **state**—Enable access control.
4. **identifier**—Provide one or more MCC or MCC or MNC match patterns for IMSI-based allowlisting.

   This identifier, a literal string, matches the Russian Federation.

```
ORACLE(ike-access-control)# identifier 250
```

   This identifier, which uses the wildcard symbol (^) signifying any single digit within the range 0 through 9, matches the continental United States.

```
ORACLE(ike-access-control)# identifier 31^
```

   This identifier, a double-quote delimited list of prefixes separated by spaces, matches T-Mobile United States networks.

```
ORACLE(ike-access-control)# identifier "26201 26206"
```

   This identifier, a double-quote delimited list of prefixes separated by spaces, matches Verizon Wireless United States networks.

```
ORACLE(ike-access-control)# identifier "310004 310012"
```

   For MAC-based allowlisting, the following double-quote delimited list identifies three specific MAC addresses.

```
ORACLE(ike-access-control)# identifier "0123456789AB 6789912345BF
DA2345918290"
```

> **Note:**
>
> Do not configure an empty allowlist. Assigning an empty allowlist to an IKEv2 interface results in authentication failure for all presented identities.

5. Type **done** to save your configuration.
6. If necessary, configure additional **ike-access-control** configuration elements.

Configure a Blocklist

A blocklist is provisioned with a femtocell's EAP identity, taking the form `<MAC ID>@cellID.serviceProvider.com` and denying authentication for such femtocells trying to establish IKE/IPsec tunnels. Blocklists are only applicable for femtocell clients performing EAP authentication to the SBC and are not applicable for clients doing password-based or certificate-based authentication.

1. Access the **ike-access-control** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-access-control
ORACLE(ike-access-control)#
```

2. **name**—Provide a unique identifier.

```
ORACLE(ike-access-control)# name block_01
```

3. **state**—Enable access control.
4. **blocklisted-dentifiers**—Provide one or more MAC-based match patterns for MAC-address-based blocklists.

   The following double-quote delimited list identifies three specific MAC addresses whose authentication is summarily rejected.

```
ORACLE(ike-access-control)# blocklisted-dentifiers "0123456789AB
6789912345BF DA2345918290"
```

   This identifier, which uses the wildcard symbol (^) signifying any single hexadecimal digit, specifies two ranges of contiguous MAC addresses.

```
ORACLE(ike-access-control)# blocklisted-dentifiers "0123456789A^,
^123456789AB"
```

   For IMSI-based blocklists, this example uses a double-quote delimited list of prefixes separated by spaces, to match Verizon Wireless United States networks.

```
ORACLE(ike-access-control)# blocklisted-dentifiers "310004 310012"
```

> **Note:**
>
> Do not configure an empty blocklist. Assigning an empty blocklist to an IKEv2 interface results in authentication eligibility for all presented identities.

5. Type **done** to save the configuration.
6. If necessary, configure additional **ike-access-control** configuration elements.

Assign an Allowlist or Blocklist to an IKEv2 Interface

1. Access the **ike-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

2. Select the **ike-interface** object to edit.

```
ORACLE(ike-interface)# select
<address>:

ORACLE(ike-interface)#
```

3. **access-control-name**—Identify the allowlist or block list assigned to the current interface.

```
ORACLE(ike-interface)# access-control-name allow_01
```

4. Type **done** to save your configuration.

Allowlist and Blocklist Interaction

Allowlists and blocklists may or may not be assigned to the IKEv2 interfaces. The following rules are used to support implementation of both list types.

1. If neither an allowlist nor a blocklist are assigned to an IKEv2 interface, all EAP authentication requests are forwarded to a RADIUS authentication server for final determination.
2. When only an allowlist is assigned to an IKEv2 interface, the incoming EAP identity is to be checked against that allowlist. When the EAP identity is contained in the allowlist, the authentication request is forwarded to a RADIUS authentication server for final determination. If the EAP identity is absent, authentication is denied.
3. When only a blocklist is assigned to an IKEv2 interface, the incoming EAP identity is checked against that blocklist list. If the EAP identity is contained in the blocklist, authentication is denied. When the EAP identity is absent, the authentication request is forwarded to a RADIUS authentication server for final determination..
4. If both a allowlist and a blocklist are assigned to an IKEv2 interface, the SBC checks both lists for incoming EAP identity.

   When the EAP identity is contained in the allowlist, and is absent from the blocklist, the authentication request is forwarded to a RADIUS authentication server for final determination.

   When the EAP identity is contained in the blocklist and is absent from the allowlist, authentication is rejected.

When the EAP identity is present in both lists, the blocklist takes priority and authentication is rejected. This situation will have been previously reported by the **verify-config** ACLI command.

When the EAP identity is absent from both lists, the allowlist takes priority. Because the allowlist does not contain the EAP identity, authentication is denied.

View Security IKE Statistics

Use the **show security ike statistics** ACLI command to view statistics derived from the IKEAuthIDError and BlocklistIKEAuthIDError counters, containing the number of authentication denials due to both allowlist and blocklist filtering.

For detailed information about the **show security ike statistics** ACLI command, see the *ACLI Reference Guide*.

Stateless Firewall

The SBC provides enhanced security-policy-based filters that can be applied to data packets coming through IPSec tunnels on the protected interfaces, and to non-IPSec packets on the trusted interfaces. These filters evaluate only the IP header layer, and treat each individual packet as a discrete event. As such, the functionality they provide can be compared to that provided by a stateless firewall.

Available filters are discussed in the following sections.

ICMP Filtering

Internet Control Message Protocol (ICMP) messages can be filtered based on message Type and associated message Codes.

ICMP Policy Configuration

Use the following procedure to define security-policy-based filtering of ICMP packets. This sample policy discards ICMP message type 0, Echo Reply, code 0, Net Unreachable.

1. From superuser mode, use the following command sequence to access security-policy configuration mode. While in this mode, you configure new security policies or modify existing policies.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# security-policy
ORACLE(security-policy)#
```

2. Use the required **name** parameter to identify this policy.

```
ORACLE(security-policy)# name deny_icmp-type0_code0
ORACLE(security-policy)#
```

3. Use the required **network-interface** parameter to provide the network interface name of the IKEv2 interface to which this security policy is applied.

```
ORACLE(security-policy)# network-interface M00:433
ORACLE(security-policy)#
```

4. Use the optional **local-ip-addr-match** parameter to specify the local IP address of the network interface.

Provide the local IP address or retain the default value, 0.0.0.0, which matches all local IP addresses.

```
ORACLE(security-policy)# local-ip-addr-match 192.168.89.10
ORACLE(security-policy)#
```

5. Use the optional **local-ip-mask** parameter to specify the local address mask.

```
ORACLE(security-policy)# local-ip-mask 255.255.0.0
ORACLE(security-policy)#
```

6. Use the optional **remote-ip-addr-match** parameter to specify the local IP address of the network interface.

   Provide the remote IP address or retain the default value, 0.0.0.0, which matches all remote IP addresses.

```
ORACLE(security-policy)# remote-ip-addr-match 15.0.0.0
ORACLE(security-policy)#
```

7. Use the optional **remote-ip-mask** parameter to specify the remote address mask.

```
ORACLE(security-policy)# local-ip-mask 255.0.0.0
ORACLE(security-policy)#
```

8. Use the optional **local-port-match** parameter in conjunction with the **local-port-match-max** parameter to specify a contiguous range of local ports to which this security policy applies.

   To specify a single local port:

```
ORACLE(security-policy)# local-port-match 64000
ORACLE(security-policy)# local-port-match-max 64000
ORACLE(security-policy)#
```

   To specify a local port range:

```
ORACLE(security-policy)# local-port-match 64000
ORACLE(security-policy)# local-port-match-max 64500
ORACLE(security-policy)#
```

   To specify all local ports:

```
ORACLE(security-policy)# local-port-match 0
ORACLE(security-policy)# local-port-match-max 65535
ORACLE(security-policy)#
```

9. Use the optional **remote-port-match** parameter in conjunction with the **remote-port-match-max** parameter to specify a contiguous range of remote ports to which this security policy applies.

To specify a single remote port:

```
ORACLE(security-policy)# remote-port-match 32000
ORACLE(security-policy)# remote-port-match-max 32000
ORACLE(security-policy)#
```

To specify a local port range:

```
ORACLE(security-policy)# remote-port-match 64000
ORACLE(security-policy)# remote-port-match-max 64500
ORACLE(security-policy)#
```

To specify all local ports:

```
ORACLE(security-policy)# remote-port-match 0
ORACLE(security-policy)# remote-port-match-max 65535
ORACLE(security-policy)#
```

10. Use the optional **priority** parameter to assign a priority to this security policy.

    Supported values are integers within the range 0 (the highest priority) through 3071 (the lowest priority).

    You can assign more than one security policy to a specific interface. With multiple security policy assignments, each policy is applied in order of its priority (highest to lowest).

    ```
    ORACLE(security-policy)# priority 0
    ORACLE(security-policy)#
    ```

11. Use the optional **trans-protocol-match** parameter to identify the filtered protocol, in this example, ICMP.

    ```
    ORACLE(security-policy)# trans-protocol-match icmp
    ORACLE(security-policy)#
    ```

12. Use the optional **trans-sub-protocol-match** parameter to identify a specific ICMP message type.

    The ICMP 8-bit Type field specifies the message type. Contents of the Type field are administered by the Internet Assigned Numbers Authority (IANA). Current Type numbers can be viewed at http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xml#icmp-parameters-types.

    The following command sequence designates the ICMP Echo Reply message.

    ```
    ORACLE(security-policy)# trans-sub-protocol-match 0
    ORACLE(security-policy)#
    ```

13. Use the optional **trans-sub-protocol-code-match** parameter to identify a specific ICMP code.

    Many ICMP message types have associated numeric codes which provide additional information pertinent to that message types. ICMP code values are administered by the Internet Assigned Numbers Authority (IANA). Up to date codes can be viewed at http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xml#icmp-parameters-codes.

In the absence of a specifically identified code, all codes are filtered.

The following command sequence designates the Net Unreachable code.

```
ORACLE(security-policy)# trans-sub-protocol-code-match 0
ORACLE(security-policy)#
```

14. Use the optional **action** parameter to specify how incoming ICMP messages that match filtering criteria are processed.

    Use **discard** to drop all ICMP messages that match filtering criteria.

    Use **allow** to pass-thru all ICMP messages that match filtering criteria.

    ```
    ORACLE(security-policy)# action discard
    ORACLE(security-policy)#
    ```

15. Use the optional **direction** parameter to identity the traffic streams subject to the processing specified by the **action** parameter.

    Use **both** to apply the specified processing to both inbound and outbound traffic.

    ```
    ORACLE(security-policy)# direction both
    ORACLE(security-policy)#
    ```

16. Ignore the **ike-sainfo-name** parameter.
17. Use **done**, **exit**, and **verify-config** to complete configuration of the security policy.
18. Repeat Steps 1 through 17 to configure additional security policies.

SCTP Filters

Internet Control Message Protocol (ICMP) messages can be filtered based on Payload Protocol Identifiers.

SCTP Policy Configuration

Use the following procedure to define security-policy-based filtering of SCTP packets. This sample policy allows SCTP, Payload Protocol Identifier 34, Diameter in SCTP DATA chunk.

1. From superuser mode, use the following command sequence to access security-policy configuration mode. While in this mode, you configure new security policies or modify existing policies.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ipsec
   ORACLE(ipsec)# security-policy
   ORACLE(security-policy)#
   ```

2. \Use the required **name** parameter to identify this policy.

   ```
   ORACLE(security-policy)# name allow_sctp-ppid_46
   ORACLE(security-policy)#
   ```

3. Use the required **network-interface** parameter to provide the network interface name of the IKEv2 interface to which this security policy is applied.

```
ORACLE(security-policy)# network-interface M00:433
ORACLE(security-policy)#
```

4. Use the optional **local-ip-addr-match parameter** to specify the local IP address of the network interface.

   Provide the local IP address or retain the default value, 0.0.0.0, which matches all local IP addresses.

```
ORACLE(security-policy)# local-ip-addr-match 192.168.89.10
ORACLE(security-policy)#
```

5. Use the optional **local-ip-mask** parameter to specify the local address mask.

```
ORACLE(security-policy)# local-ip-mask 255.255.0.0
ORACLE(security-policy)#
```

6. Use the optional **remote-ip-addr-match parameter** to specify the remote IP address of the network interface.

   Provide the remote IP address or retain the default value, 0.0.0.0, which matches all local IP addresses.

```
ORACLE(security-policy)# remote-ip-addr-match 192.168.89.10
ORACLE(security-policy)#
```

7. Use the optional **remote-ip-mask** parameter to specify the remote address mask.

```
ORACLE(security-policy)# remote-ip-mask 255.255.0.0
ORACLE(security-policy)#
```

8. Use the optional **local-port-match** parameter in conjunction with the **local-port-match-max** parameter to specify a contiguous range of local ports to which this security policy applies.

   To specify a single local port:

```
ORACLE(security-policy)# local-port-match 64000
ORACLE(security-policy)# local-port-match-max 64000
ORACLE(security-policy)#
```

   To specify a local port range:

```
ORACLE(security-policy)# local-port-match 64000
ORACLE(security-policy)# local-port-match-max 64500
ORACLE(security-policy)#
```

   To specify all local ports:

```
ORACLE(security-policy)# local-port-match 0
ORACLE(security-policy)# local-port-match-max 65535
ORACLE(security-policy)#
```

9. Use the optional **remote-port-match** parameter in conjunction with the **remote-port-match-max** parameter to specify a contiguous range of remote ports to which this security policy applies.

   To specify a single remote port:

   ```
   ORACLE(security-policy)# remote-port-match 32000
   ORACLE(security-policy)# remote-port-match-max 32000
   ORACLE(security-policy)#
   ```

   To specify a local port range:

   ```
   ORACLE(security-policy)# remote-port-match 64000
   ORACLE(security-policy)# remote-port-match-max 64500
   ORACLE(security-policy)#
   ```

   To specify all local ports:

   ```
   ORACLE(security-policy)# remote-port-match 0
   ORACLE(security-policy)# remote-port-match-max 65535
   ORACLE(security-policy)#
   ```

10. Use the optional **priority** parameter to assign a priority to this security policy.

    Supported values are integers within the range 0 (the highest priority) through 3071 (the lowest priority).

    You can assign more than one security policy to a specific interface. With multiple security policy assignments, each policy is applied in order of its priority (highest to lowest).

    ```
    ORACLE(security-policy)# priority 0
    ORACLE(security-policy)#
    ```

11. Use the optional **trans-protocol-match** parameter to identify the filtered protocol, in this example, SCTP.

    ```
    ORACLE(security-policy)# trans-protocol-match sctp
    ORACLE(security-policy)#
    ```

12. Use the optional **trans-sub-protocol-match** parameter to identify a specific SCTP Protocol Payload Identifier.

    SCTP DATA chunks contain a 32-bit Payload Protocol Identifier field specifying the protocol that originated the data contained in the SCTP chunk. SCTP Payload Protocol Identifiers are administered by the IANA. Current identifiers can be found at http://www.iana.org/assignments/sctp-parameters/sctp-parameters.xml#sctp-parameters-25.

    The following command sequence designates DIAMETER data.

    ```
    ORACLE(security-policy)# trans-sub-protocol-match 46
    ORACLE(security-policy)#
    ```

13. Ignore the optional **trans-sub-protocol-code-match** parameter, which is not currently used for SCTP filter configuration.
14. Use the optional **action** parameter to specify how incoming SCTP messages that match filtering criteria are processed.

Use **discard** to drop all SCTP messages that match filtering criteria.

Use **allow** to pass-thru all SCTP messages that match filtering criteria.

```
ORACLE(security-policy)# action allow
ORACLE(security-policy)#
```

15. Use the optional **direction** parameter to identity the traffic streams subject to the processing specified by the **action** parameter.

    Use **both** to apply the specified processing to both inbound and outbound traffic.

```
ORACLE(security-policy)# direction both
ORACLE(security-policy)#
```

16. Ignore the **ike-sainfo-name** parameter.
17. Use **done**, **exit**, and **verify-config** to complete configuration of the security policy.
18. Repeat Steps 1 through 17 to configure additional security policies.

Source Routing Packets

The SBC can unconditionally discard all source routed packets at the global level. Source routed packets are identified by the presence of either a Loose Source Route/Record (LSRR) or a Strict Source Route/Record (SSRR) option within the IP header Options header.

Both options have the potential to mask malicious intent. An attacker can use the specified routes to hide the true source of a packet, or to gain access to a protected network. Consequently, such packets are often dropped upon network entry.

Use the following procedure to unconditionally discard all source routed packets.

1. From superuser mode, use the following command sequence to access ipsec-global-config configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# ipsec-global-config
ORACLE(ipsec-global-config)#
```

2. Use the **options** command in conjunction with the **source-routing-drop** argument to unconditionally discard all source routing packets — identified by the presence of either an SSRR or LSRR option in the IP header Options header.

```
ORACLE(ipsec-global-config)# options +source-routing-drop
ORACLE(ipsec-global-config)#
```

3. Use the **done** and **exit** to complete configuration.

Fragmented Packets

The SBC can unconditionally discard all inbound fragmented Encapsulating Security Protocol (ESP) packets using a global option. Refer to Figure 3, ESP Transport Mode, and Figure 5, ESP Tunnel Mode, for ESP details.

Upon reception, the SG re-assembles such packets and then decrypts the re-assembled packet. After decryption, if the decrypted packet is still a fragment, the new option mandates that the packet fragment be discarded in light of the SG's inability to do a proper policy check on an incomplete message.

Use the following procedure to unconditionally discard all fragmented ESP packets.

1. From superuser mode, use the following command sequence to access ipsec-global-config configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# ipsec-global-config
ORACLE(ipsec-global-config)#
```

2. Use the **options** command in conjunction with the **fragmented-packet-drop** argument to unconditionally discard all inbound fragmented ESP packets.

```
ORACLE(ipsec-global-config)# options +fragmented-packet-drop
ORACLE(ipsec-global-config)#
```

3. Use the **done** and **exit** to complete configuration.

ACLI show Commands

Two new ACLI commands display filtering statistics.

- **show security security-policy statistics all**, which displays statistics for all filtering policies
- **show security security-policy [policyName]**, which displays statistics for a specific filtering policy

Threshold Crossing Alert Configuration

Threshold Crossing Alerts (TCAs) monitor specific MIB variables or counters, and generate SNMP traps when object values cross defined thresholds. Three types of TCAs are supported:

- IKE Failed Authentication (monitors IKE negotiation counters)
- IPsec Tunnel Removal (monitors IPsec tunnel counters)
- Dead Peer Detections (monitors DPD protocol counters)

Threshold levels, listed in order of increasing importance are clear, minor, major, and critical. Each threshold level is user-configurable and is accompanied by a associated reset-counter, also user-configurable, which prevents the issue of extraneous SNMP traps when a counter is bouncing across threshold values.

A threshold crossing event occurs when the associated counter value rises above the next-highest threshold value, or when the associated counter value falls below the next-lowest reset-threshold value. An SNMP trap, raising the alert level, is generated as soon as the counter value exceeds the next-highest threshold. An SNMP trap, lowering the alert level, occurs only during a check period when the TCA examines all counter values. Such check periods occur at 100 second intervals.

The following scenario illustrates TCA operations. The sample TCA, ike-tca-group, monitors the count of dead IKEv2 peers. Threshold and reset values are shown. A minor alarm threshold and its associated reset threshold have not been configured.

```
nameike-tca-group
tca-typeike-dpd
critical100
reset-critical90
major80
```

```
reset-major50
minor0
reset-minor0
```

t=time

t=0 ike-dpd counter= 30 ike-dpd alert level=clear

t=1 ike-dpd counter= 60 ike-dpd alert level=clear

t=2 ike-dpd counter= 80 ike-dpd alert level=major trap sent

t=3 ike-dpd counter= 95 ike-dpd alert level=major

t=4 ike-dpd counter=100 ike-dpd alert level=critical trap sent

t=5 ike-dpd counter=120 ike-dpd alert level=critical

t=6 ike-dpd counter= 99 ike-dpd alert level=critical

t=7 ike-dpd counter= 90 ike-dpd alert level=major trap sent

t=8 ike-dpd counter= 60 ike-dpd alert level=major

t=9 ike-dpd counter= 0 ike-dpd alert level=clear trap sent

Use the following procedure to configure TCAs.

1. From superuser mode, use the following command sequence to access threshold-crossing-alert-group configuration mode. While in this mode, you configure threshold-crossing-alert-group configuration elements.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# threshold-crossing-alert-group
   ORACLE(threshold-crossing-alert-group)#
   ```

2. Use the **name** parameter to provide a unique identifier for this threshold-crossing-alert-group instance.

   name enables the creation of multiple threshold-crossing-alert-group instances.

   ```
   ORACLE(threshold-crossing-alert-group)# name ikeTCA
   ORACLE(threshold-crossing-alert-group)#
   ```

3. Use the **threshold-crossing-alert** parameter to enter threshold-crossing-alert configuration mode. While in this mode, you create specific TCA types and associated values.

   ```
   ORACLE(threshold-crossing-alert-group)# threshold-crossing-alert
   ORACLE(threshold-crossing-alert)#
   ```

4. Use the **type** parameter to specify the TCA type.

   Supported values are:

   - ike-failed-auth — (the default) tracks authentication failures
   - ipsec-tunnel-removal — tracks the destruction of IPsec tunnels

- ike-dpd — tracks the detection of dead DPD peers

```
ORACLE(threshold-crossing-alert)# type ike-dpd
ORACLE(threshold-crossing-alert)#
```

5. Use the **critical** parameter to specify the critical threshold level.

   The default value (0) indicates that the threshold is not configured.

```
ORACLE(threshold-crossing-alert)# critical 100
ORACLE(threshold-crossing-alert)#
```

6. Use the **reset-critical** parameter to specify the value at which the critical level is replaced with the next lowest configured threshold level (major, minor, or clear, depending on configuration values).

   The default value (0) indicates that the threshold is not configured.

```
ORACLE(threshold-crossing-alert)# reset-critical 90
ORACLE(threshold-crossing-alert)#
```

7. Use the **major** parameter to specify the major threshold level.

   The default value (0) indicates that the threshold is not configured.

```
ORACLE(threshold-crossing-alert)# major 80
ORACLE(threshold-crossing-alert)#
```

8. Use the **reset-major** parameter to specify the value at which the major level is replaced with the next lowest configured threshold level (minor or clear, depending on configuration values).

   The default value (0) indicates that the threshold is not configured.

```
ORACLE(threshold-crossing-alert)# reset-major 50
ORACLE(threshold-crossing-alert)#
```

9. Use the **minor** parameter to specify the minor threshold level.

   The default value (0) indicates that the threshold is not configured.

```
ORACLE(threshold-crossing-alert)# minor 0
ORACLE(threshold-crossing-alert)#
```

10. Use the **reset-minor** parameter to specify the value at which the minor level is replaced with the next lowest configured threshold level (clear).

    The default value (0) indicates that the threshold is not configured.

```
ORACLE(threshold-crossing-alert)# reset-minor 0
ORACLE(threshold-crossing-alert)#
```

11. If required, repeat Steps 4 through 10 to add other TCA types to the current threshold-crossing-alert-group configuration element.

    The threshold-crossing-alert-group configuration element can contain a maximum of three individual threshold-crossing-alerts, one of each supported type.

12. Use **done**, **exit**, and **verify-config** to complete configuration of the threshold-crossing-alert-group configuration element.
13. If necessary, repeat Steps 1 through 12 to configure additional threshold-crossing-alert-group configuration elements.
14. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you configure IKEv2 interface parameters.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# ike-interface
ORACLE(ike-interface)#
```

15. Use the optional **threshold-crossing-alert-group-name** parameter to assign an existing threshold-crossing-alert-group configuration element to this IKEv2 interface.

```
ORACLE(ike-interface)# threshold-crossing-alert-group-name ikeTCA
ORACLE(ike-interface)#
```

16. Use **done**, **exit**, and **verify-config** to complete configuration of the TCA.

IKEv2 Interface Management

The following two sections provide details on available counters that gather usage and error data related to IKEv2/IPsec operations on the Oracle Communications Session Border Controller.

The first section, IKEv2 Protocol Operations, describes a series of 32-bit counters that report interface-specific data on various protocol transactions. Protocol operations counter values are available with SNMP, through the ACLI **show security ike statistics** command, and can also be obtained by subscription to the ike_stats HDR group.

The second section, IKEv2 Negotiation Errors, describes a series of 32-bit counters that report interface-specific errors encountered during IKEv2 negotiations. Negotiation errors counter values are also available with SNMP, through the ACLI **show security ike statistics** command, and can also be obtained by subscription to the ike-stats HDR group.

The third section, RADIUS Protocol Operations, describes a series of 32-bit counters that report RADIUS-server-specific data. RADIUS protocol operations counter values are also available with SNMP, through the ACLI **show radius** command, and can also be obtained by subscription to the radius-stats HDR group.

The final section, Diameter Protocol Operations, describes a series of 32-bit counters that report Diameter-server-specific data. Diameter protocol operations counter values are also available with SNMP, and can also be obtained by subscription to the diameter-stats HDR group.

IKEv2 Protocol Operations

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.9.1.9.X (apSecurityIkeInterfaceInfoTable), where X specifies the interface index. For example, the SNMP MIB for the Current Child SA Pairs is 1.3.6.1.4.1.9148.3.9.1.9.X.33, where X specifies the interface index.

> **Note:**
>
> The range for all 32-bit counters is 0 to 4294967295.

Name	Description	Type	SNMP MIB Ending
Current Child SA Pairs	The number of current child IPsec SA pairs on the interface. As each IPsec tunnel requires two unidirectional SAs, this number equals the current number of tunnels on the interface. Note that this count is available through both an ACLI show command and an SNMP GET operation.	gauge	.33
Maximum Child SA Pairs	The largest number of child IPsec SA pairs on the interface since this counter was last reset. As each IPsec tunnel requires a single SA pair, this value equates to the largest number of tunnels on the interface.	gauge	
Last Reset Timestamp	The time that this interface was last reset -- expressed as a UNIX timestamp containing the number of seconds since January 1, 1970.	UNIX timestamp	
Child SA Request	The number of requests to add a child SA pair that were received on the interface. These requests include IPsec SA rekey requests.	counter	.1
Child SA Success	The number of requests to add a child SA pair that were successfully completed on the interface. These successes include new children created by IPsec SA rekeys.	counter	.2
Child SA Failure	The number of requests to add a child SA pair that were not successfully completed on the interface. These failures include unsuccessful IPsec SA rekeys.	counter	.3

Name	Description	Type	SNMP MIB Ending
Child SA Delete Requests	The number of requests to delete a child SA pair that were received on the interface. These requests include deletion requests associated with IPsec SA rekeys.	counter	.4
Child SA Delete Success	The number of requests to delete a child SA pair that were successfully completed on the interface. These successes include children deleted by IPsec SA rekeys.	counter	.5
Child SA Delete Failure	The number of requests to delete a child SA pair that were not successfully completed on the interface. These failures include unsuccessful deletions associated IPsec SA rekeys.	counter	.6
Child SA Rekey	The number of child IPsec rekey exchanges transacted on the interface.	counter	.7
Initial Child SA Establishment	The number of initial child SA pair establishments, in other words, the number of successful IKE_AUTH exchanges transacted on the interface.	counter	.8
DPD Received Port Change	The number of DPD messages received on the interface that contained a port change from the previously received message. The port change indicates that the IKEv2 has moved to another port, or that an intervening NAT device has changed port mapping. These actions do not impact SA functions.	counter	.9
DPD Received IP Change	The number of DPD messages received on the interface that contained an IP address change from the previously received message.	counter	.10

Name	Description	Type	SNMP MIB Ending
DPD Response Received	The number of DPD ACK responses received on the interface. An ACK is sent by an IKEv2 peer in response to an R-U-THERE issued by the Oracle Communications Session Border Controller. A successful R-U-THERE/ACK exchange establishes availability on the remote IKEv2 peer.	counter	.11
DPD Response Not Received	The number of R-U-THERE messages transmitted on the interface that were not acknowledged within the DPD allowed interval.	counter	.12
DPD Received	The number of all DPD protocol messages received on the interface.	counter	.13
DPD Retransmitted	The number of R-U-THERE messages that were re-transmitted because the original R-U-THERE message was not acknowledged.	counter	.14
DPD Sent	The number of R-U-THERE messages that were sent across the interface, to include retransmitals.	counter	.15
IKE SA Packets Sent	The number of IKEv2 SA packets sent across the interface.	counter	.16
IKE SA Packets Received	The number of IKEv2 SA packets received across the interface.	counter	.17
IKE SA Packets Dropped	The number of IKEv2 SA packets dropped by the interface.	counter	.18
Authentication Failures	The number of authentication failures that occurred after the purported identity of the remote IKEv2 peer was ascertained.	counter	.19
IKE Message Errors	The number of otherwise uncharacterized IKEv2 message errors.	counter	.20
Authentication ID Errors	The number of errors that occurred during the identification stage of the authentication process.	counter	.21

Name	Description	Type	SNMP MIB Ending
Certificate Status Requests	The number of certificate status requests sent across the interface to an OCSP responder.	counter	.22
Certificate Status Success	The total number of OCSP successes, that is the number of OCSP requests that generated a good status from an OCSP responder.	counter	.23
Certificate Status Fail	The total number of OCSP failures, to include unacknowledged OCSP requests and those requests that generated a revoked or unknown response from an OCSP responder.	counter	.24
DDoS Sent	The number of suspicious, and possibly malicious, endpoints reported by the interface-specific DDoS process (if configured as described in the IKEv2 DDoS Protection section of the Oracle Communications Session Border Controller Essentials guide).	counter	.25
DDoS Received	The number of suspicious, and possibly malicious, endpoints reported by statically provisioned deny lists (as described in SIP Signaling Services and Security chapters of the ACLI Configuration Guide).	counter	.26
IKE Message Retransmissions	The total number of IKEv2 message re-transmissions.	counter	.27
SA Init Messages Received	The total number of IKEv2 message re-transmissions.	counter	.28
SA Init Message Sent	The total number of IKEv2 message re-transmissions.	counter	.29
SA Establishment Attempts	The total number of IKEv2 message re-transmissions.	counter	.30

Name	Description	Type	SNMP MIB Ending
SA Establishment Success	The total number of IKEv2 SA successfully established on the IKEv2 interface.	counter	.31
Tunnel Rate	Specifies the tunnel establishment rate, in terms of tunnels created per second. Note that this count is available through both an ACLI show command and an SNMP GET operation.	gauge	.32

IKEv2 Negotiation Errors

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.9.1.3.X (apSecurityIkeInterfaceStatsEntry), where X specifies the interface index. For example, the SNMP MIB for the CPU Overload Errors is 1.3.6.1.4.1.9148.3.9.1.3.X.3, where X specifies the interface index.

Name	Description	SNMP MIB Ending
CPU Overload Errors	The number of IKEv2 requests that were rejected because of CPU load constraints.	.3
Init Cookie Errors	The number of all IKEv2 exchanges that failed because of faulty Security Parameter Index (SPI) values. SPIs provide a local SA identifier and are exchanged between IKEv2 peers in the common IKEv2 header and in Notify Payloads.	.4
Auth Errors	The number of failed IKE_AUTH exchanges, regardless of the specific reason for failure.	.5
EAP Access Request Errors	The number of authentication failures that occur ed during the EAP access phase.	.6
EAP Access Challenge Errors	The number of authentication failures that occur ed during the EAP challenge phase.	.7
TS Errors	The number of CREATE_CHILD_SA exchanges that failed because of faulty TS payload contents, or failure on the part of the remote peers to negotiate the offered traffic selectors.	.8
CP Errors	The number of IKE_AUTH and/or CREATE_CHILD_SA exchanges that failed because of faulty, unsupported, or unknown Configuration Payload contents.	.9

Name	Description	SNMP MIB Ending
IKE Errors	The number of IKE_SA_INIT and/or CREATE_CHILD_SA exchanges that failed because of faulty, unsupported, or unknown Key Exchange Payload contents.	.10
Proposal Errors	The number of failed negotiations that resulted from the inability to reconcile crytographic proposals contained in the Security Association Payloads exchanged by IKEv2 peers. Security Association Payloads are exchanged during the IKE_SA_INIT, IKE_AUTH, and CREATE_CHILD_SA stages.	.11
Syntax Errors	The number of failed negotiations, of any type, resulting from otherwise uncharacterized errors.	.12
Critical Payload Errors	The number of failed negotiations that resulted from the presence of a Critical flag in a payload that could not be parsed, or was not supported. IKEv2 adds a critical flag to each payload header for further flexibility for forward compatibility. If the critical flag is set and the payload type is unrecognized, the message must be rejected and the response to the IKE request containing that payload MUST include a Notify payload UNSUPPORTED_CRITICAL_PAYLOAD, indicating an unsupported critical payload was included. If the critical flag is not set and the payload type is unsupported, that payload must be ignored.	.13

RADIUS Protocol Operations

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.18.1.1.1 (aapRadiusServerStatsEntry). For example, the SNMP MIB for the Server Roundtrip Time is 1.3.6.1.4.1.9148.3.18.1.1.1.3.

Name	Description	SNMP MIB Ending
Server Roundtrip Time	Contains the average round trip time for a response from this RADIUS server.	.3
Server Malformed Access Response	Contains the number of malformed access responses received on this RADIUS server.	.4
Server Access Requests	Contains the number of access requests received on this RADIUS server.	.5

Name	Description	SNMP MIB Ending
Server Disconnect Requests	Contains the number of disconnect requests received on this RADIUS server.	.6
Server Disconnect ACKS	Contains the number of acknowledged disconnects on this RADIUS server.	.7
Server Disconnect NACKS	Contains the number of unacknowledged disconnects on this RADIUS server.	.8
Server Bad Authenticators	Contains the number of authentication rejections on this RADIUS server.	.9
Server Access Retransmissions	Contains the number of access retransmitals on this RADIUS server.	.10
Server Access Accepts	Contains the number of successful authentications on this RADIUS server.	.11
Server Timeouts	Contains the number of Response timeouts on this RADIUS server.	.12
Server Access Rejects	Contains the number of unsuccessful authentications on this RADIUS server.	.13
Server Unknown PDUTypes	Contains the number or unknown/unreadable PDUs received by this RADIUS server.	.14
Server Access Challenges	Contains the number of Access Challenges on this RADIUS server.	.15

Diameter Protocol Operations

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.13.1.1.2.2.X (apDiamInterfaceStatsTable), where X specifies the diameter server index. For example, the SNMP MIB for the Diameter Messages Sent is 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.3, where X specifies the diameter server index.

Name	Description	SNMP MIB Ending
Diameter Messages Sent	Contains the number of messages sent by this Diameter server.	.3
Diameter Messages Sent Failed	Contains the number of unacknowledged messages sent by this Diameter server.	.4
Diameter Messages Resent	Contains the number of messages re-transmitted to this Diameter server.	.5
Diameter Messages Received	Contains the number of messages received by this Diameter server.	.6
Diameter Messages Processed	Contains the number of messages processed by this Diameter server.	.7
Diameter Connection Timeouts	Contains the number of connection timeouts on the Diameter server.	.8
Diameter BadState Drops	Contains the number of packets dropped because of faulty state on the Diameter server.	.9

Name	Description	SNMP MIB Ending
Diameter BadType Drops	Contains the number of packets dropped because of faulty type on the Diameter server.	.10
Diameter BadID Drops	Contains the number of packets dropped because of faulty ID on the Diameter server.	.11
Diameter AuthFail Drops	Contains the number of failed authentications on the Diameter server.	.12
Diameter Invalid Peer Messages	Contains the number of client messages that could not be parsed on the Diameter server.	.13

ACLI Show Commands

ACLI **show** commands

- display and reset IKEv2 performance and error counters
- display IKEv2 SA data
- display IKEv2 TCA data

Performance and Error Counters

Three ACLI commands display and reset IKEv2 performance and error counters.

Use the **show security** command to display performance and error counters for a specified IKEv2 interface, or for all IKEv2 interfaces.

```
ORACLE# show security 192.169.204.15
```

with a specified interface, displays performance and error counters for the target interface

```
ORACLE# show security all
```

with all, displays performance and error counters for all IKEv2 interfaces

Use the **reset ike-stats** command to reset (set to 0) performance and error counters for a specified IKEv2 interface, or for all IKEv2 interfaces.

```
ORACLE# reset ike-stats 192.169.204.15
```

with a specified interface, resets performance and error counters for the target interface

```
ORACLE# reset ike-stats all
```

with all, resets performance and error counters for all IKEv2 interfaces

Use the **reset ike-mib** command to reset (set to 0) MIB-based error counters for all IKEv2 interfaces.

```
ORACLE# reset ike-mib
```

re-sets the MIB-based error counters for all IKEv2 interfaces

IKEv2 and Child SAs

Use the **show security** command with optional arguments to display IKEv2 and child SA information to include:

- IP address and port of remote end-point
- intervening NAT device (yes | no)
- local IP address
- tunnel state (up | down)
- initiator cookie
- responder cookie
- remote inner (tunnel) IP address
- incoming/outgoing Security Parameter Indexes (SPI) of the child SA

```
ORACLE# show security ike sad ike-interface 192.169.204.15
```

with a specified interface address, displays SA information for a single IKEv2 interface

```
ORACLE# show security ike sad ike-interface all
```

with all, displays SA information for all IKEv2 interfaces

```
ORACLE# show security ike sad ike-interface all
Displaying the total (4321) number of entries may take long and could affect
system performance.
Continue? [y/n]?: y
Peer: 6.0.0.36:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0x23e71b73d5a10c58[I] 0xd2017a6fb84a4fa6[R]
    Child Peer IP: 101.0.0.36:0 Child SPI: 4236760138[I] 1721373661[O]
Peer: 6.0.0.28:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0xf64d031d32525730[I] 0xcea2d5ae3c91050f[R]
    Child Peer IP: 101.0.0.28:0 Child SPI: 3632387333[I] 1421117246[O]
Peer: 6.0.0.9:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0x84ec95a1cd0a4c5d[I] 0x1b61b385c4e627b4[R]
    Child Peer IP: 101.0.0.9:0 Child SPI: 2432742837[I] 3872387177[O]
Peer: 6.0.0.25:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0x541b2651e88c9368[I] 0xdc393a61af6dc909[R]
    Child Peer IP: 101.0.0.25:0 Child SPI: 785656546[I] 148357787[O]
Peer: 6.0.0.27:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0x3ba43c5c685e37e6[I] 0x7bfa6f0781dce1a8[R]
    Child Peer IP: 101.0.0.27:0 Child SPI: 767765646[I] 3797275291[O]
Peer: 6.0.0.22:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0x925e540ecbd58dbb[I] 0x7e1101371a5a5823[R]
    Child Peer IP: 101.0.0.22:0 Child SPI: 787745714[I] 876969665[O]
Peer: 6.0.0.2:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0xda0f568684ba5e2c[I] 0x74c533da2fd29901[R]
    Child Peer IP: 101.0.0.2:0 Child SPI: 3884481109[I] 1862217459[O]
Peer: 6.0.0.7:500 (NAT: No) Host: 172.16.101.2 State: Up
    IKE Cookies: 0x6166bac4438f3ca7[I] 0x71d1049a0f8520f4[R]
    Child Peer IP: 101.0.0.7:0 Child SPI: 2798332266[I] 2789214337[O]
Peer: 6.0.0.15:500 (NAT: No) Host: 172.16.101.2 State: Up
```

```
    IKE Cookies: 0x0e060701115069bf[I] 0x2e69adbf15438000[R]
    Child Peer IP: 101.0.0.15:0 Child SPI: 713005957[I] 1985608540[O]
Continue? [y/n]?: y
...
...
```

Use **show security** with the peer address obtained by the previous command to display more detailed information regarding a specific tunnel to include:

- IKE version
- Diffie Hellman group
- the IKE SA hash algorithm
- the IKE SA message authentication code algorithm
- the IKE SA encryption algorithm
- seconds since SA creation
- SA lifetime in seconds
- remaining lifetime in seconds
- IPsec operational mode (tunnel | transport)
- IPsec security protocol (AH | ESP)
- IPsec authentication protocol
- IPsec encryption protocol

```
ORACLE# show security ike sad ike-interface <ipAddress> peer <ipAddress>
ORACLE# show security ike sad ike-interface 172.16.101.2 peer 6.0.0.36:500

IKE SA:

    IKE Version : 2
    Tunnel State : Up
    Last Response [Seconds] : 212
    AAA Identity :
    NAT : No

    IP Addresses [IP:Port]
        Peer : 6.0.0.36:500
        Server Instance : 172.16.101.2:500

    Cookies
        Initiator : 0x23e71b73d5a10c58
        Responder : 0xd2017a6fb84a4fa6

    Algorithms
        DH Group : 17
        Hash : HMAC-SHA-256
        MAC : SHA-256
        Cipher : AES

    SA Times [Seconds]
        Creation : 141
        Expiry : 86400
```

```
        Remaining : 86188

IPSec SA:

    IP Addresses [IP:Port]
        Destination : 101.0.0.36:0
        Source : 172.16.101.2:0

    SPI
        Outbound : 1721373661
        Inbound : 4236760138

    Algorithms
        Mode : TUNNEL
        Protocol : ESP
        Authentication : SHA1
        Encryption : AES

    Traffic Selectors [Start IP - End IP]
        Destination : 101.0.0.36 - 101.0.0.36
        Source : 172.16.101.2 - 172.16.101.2
```

TCA Counters

An ACLI command is provided to display TCA information.

```
ORACLE# show security ike threshold-crossing-alert <ipAddress> || all
```

with a specified IPv4/IPv6 interface address, displays TCA information for the specified IKEv2 interface, otherwise displays TCA information for all IKEv2 interfaces

```
ORACLE# show security ike threshold-crossing-alert all
ORACLE# show security ike threshold-crossing-alert all
IKE Threshold Crossing Alerts
tca-type: ike-auth-failure
           reset            reset                           reset
critical     critical    major     major       minor      minor
---------- ---------- ---------- ---------- ---------- ----------
       40         30         25         24         12          1
current value:
   Window Total Maximum
        0    0        0
current level: clear

tca-type: ipsec-tunnel-removal
             reset               reset                     reset
critical     critical    major     major       minor      minor
---------- ---------- ---------- ---------- ---------- ----------
        0          0         10          5          0          0
current value:
     Window Total Maximum
          0     0       0
current level: clear
```

TCA Traps

TCAs generate SNMP traps to report crossing of threshold levels, or to clear threshold levels.

Historical Data Records

Various statistical counts are available as comma separated values (CSV) Historical Data Record (HDR) files. HDR files are specified and pushed to an accounting server as described in the Overview chapter of the 4000 C-Series Historical Data Recording (HDR) Resource Guide.

IKEv2 Interface HDR

CSV header fields for IKEv2 Interface HDRs are listed below.

IKEv2 Interface HDR	Type
TimeStamp	Integer
Interface	IP Address
Current Child SA Pairs	Counter
Maximum Child SA Pairs	Counter
Last Reset TimeStamp	Integer
Child SA Requests	Counter
Child SA Success	Counter
Child SA Failure	Counter
Child SA Delete Request	Counter
Child SA Delete Success	Counter
Child SA Delete Failure	Counter
Child SA Rekey	Counter
Initial Child SA Establishment	Counter
DPD Received Port Change	Counter
DPD Received IP Change	Counter
DPD Response Received	Counter
DPD Response Not Received	Counter
DPD Received	Counter
DPD Retransmitted	Counter
DPD Sent	Counter
IKE SA Packets Sent	Counter
IKE SA Packets Received	Counter
IKE SA Packets Dropped	Counter
Authentication Failures	Counter
IKE Message Errors	Counter
Authentication ID Errors	Counter
Certificate Status Requests	Counter
Certificate Status Success	Counter
Certificate Status Fail	Counter
DDoS Sent	Counter
DDoS Received	Counter
IKE Message Retransmissions	Counter
Tunnel Rate	Counter
Child SA Pair	Guage
IKE SA INIT Messages Received	Counter
IKE SA INIT Messages Sent	Counter
IKE SA Establishment Attempts	Counter

IKEv2 Interface HDR	Type
IKE SA Establishment Success	Counter
IKE CPU Overload Error	Counter
IKE init Cookie Error	Counter
IKE EapAccessRequestError	Counter
IKE EapAccessChallengeError	Counter
IKE TS Error	Counter
IKE CP Error	Counter
IKE KE Error	Counter
IKE Proposal Error	Counter
IKE Syntax Error	Counter
IKE Critica; Payload Error	Counter

RADIUS HDR

CSV header fields for RADIUS HDRs are listed below.

IKEv2 Interface HDR	Type
Time Stamp	Integer
RADIUS Sever IP Address	IP Address
RADIUS Server Port	Port Address
Round Trip Time	Counter
Malformed Access Response	Counter
Access Requests	Counter
Disconnect Requests	Counter
Disconnect ACKs	Counter
Bad Authenticators	Counter
Access Retransmissions	Counter
Access Accepts	Counter
Timeouts	Counter
Access Rejects	Counter
Unknown PDU Types	Counter
Access Challenges	Counter

Diameter HDR

CSV header fields for Diameter HDRs are listed below.

IKEv2 Interface HDR	Type
Time Stamp	Integer
Diameter Sever IP Address	IP Address
Diameter Server Port	Port Address
Messages Sent	Counter
Messages Sent Failed	Counter
Messages Resent	Counter
Messages Received	Counter
Messages Processed	Counter
Connection Timeouts	Counter
Bad State Drops	Counter

IKEv2 Interface HDR	Type
Bad Type Drops	Counter
Bad ID Drops	Counter
Auth Failed Drops	Counter
Invalid Peer Messages	Counter

Secure Real-Time Transport Protocol

The Secure Real-Time Transport Protocol, as described in RFC 3711, *The Secure Real-time Transport Protocol (SRTP)*, provides a framework for the encryption and authentication of Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) streams. Both RTP and RTCP are defined by RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*.

Encryption ensures that the call content and associated signaling remains private during transmission. Authentication ensures that (1) received packets are from the purported source, (2) packets are not been tampered with during transmission, and (3) a packet has not been replayed by a malicious server.

Protocol Overview

While the RFC 3711 framework provides encryption and authentication procedures and defines a set of default cryptographic transforms required for RFC compliance, it does not specify a key management protocol to securely derive and exchange cryptographic keys. RFC4568, *Session Description Protocol (SDP) Security Description for Media Streams*, defines such a protocol specifically designed to exchange cryptographic material using a newly defined SDP crypto attribute. Cryptographic parameters are established with only a single message or in single round-trip exchange using the offer/answer model defined in RFC 3264, *An Offer/Answer Model with the Session Description Protocol*.

Release S-C6.2.0 provides support for an initial SDP Security Descriptions (SDES) implementation that generates keys used to encrypt SRTP/SRTCP packets. Authentication of packets will be added to a subsequent release.

A sample SDP exchange is shown below:

The SDP offerer sends:

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
```

The SDP answerer replies:

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
```

```
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
```

The media-level SDP attribute, crypto, describes the cryptographic suite, key parameters, and session parameters for the preceding unicast media line. The crypto attribute takes the form:

a=crypto: tag crypto-suite key-parameter [session-parameters]

tag

The tag field contains a decimal number that identifies a specific attribute instance. When an offer contains multiple crypto attributes, the answer uses the tag value to identify the accepted offer.

In the sample offer the tag value is 1.

crypto-suite

The crypto-suite field contains the encryption and authentication algorithms.

- AES_CM_128_HMAC_SHA1_80
- AES_CM_128_HMAC_SHA1_32
- AES_256_CM_HMAC_SHA1_80
- AEAD_AES_256_GCM

The key-parameter field contains one or more sets of keying material for the selected crypto-suite and it has following format.

```
"inline:" <key||salt> ["|" lifetime] ["|" MKI ":" length]
```

inline is a method and specifies that the crypto material to be used by the offerer is transmitted via the SDP.

The key||salt field contains a base64-encoded concatenated master key and salt.

Assuming the offer is accepted, the key || salt provides the crypto material used by the offerer to encrypt SRTP/SRTCP packets, and used by the answerer to decrypt SRTP/SRTCP packets.

Conversely the key || salt contained in the answer to the offer provides the crypto material used by the answerer to encrypt SRTP/SRTCP packets, and used by the offerer to decrypt SRTP/SRTCP packets.

The lifetime field optionally contains the master key lifetime (maximum number of SRTP or SRTCP packets encoded using this master key).

In the sample offer the lifetime value is 1,048, 576 (220) packets.

The MKI:length field optionally contains the Master Key Index (MKI) value and the MKI length.

The MKI is used only when the offer contains multiple keys; it provides a means to differentiate one key from another. The MKI takes the form of an integer, followed by its byte length when included in SRTP/SRTCP packets.

In the sample offer the MKI value is 1 with a length of 4 bytes.

The session-parameters field contains a set of optional parameters that may override SRTP session defaults for the SRTP and SRTCP streams.

UNENCRYPTED_SRTP — SRTP messages are not encrypted

UNENCRYPTED_SRTCP — SRTCP messages are not encrypted

UNAUTHENTICATED_SRTP — SRTP messages are not authenticated

When generating an initial offer, the offerer must ensure that there is at least one crypto attribute for each media stream for which security is desired. Each crypto attribute for a given media stream must contain a unique tag. The ordering of multiple crypto attributes is significant — the most preferred crypto suite is listed first.

Upon receiving the initial offer, the answerer must either accept one of the offered crypto attributes, or reject the offer in its entirety.

When an offered crypto attribute is accepted, the crypto attribute in the answer MUST contain the tag and crypto-suite from the accepted crypto attribute in the offer, and the key(s) the answerer will be using for media sent to the offerer.

The crypto-suite is bidirectional and specifies encryption and authentication algorithms for both ends of the connection. The keys are unidirectional in that one key or key set encrypts and decrypts traffic originated by the offerer, while the other key or key set encrypts and decrypts traffic originated by the answerer. The use of symmetric keying, where the same key is used for both encryption and decryption, mandates the key exchange between the offerer and the answerer.

Key exchange via text-based SDP is unacceptable in that malicious network elements could easily eavesdrop and obtain the plaintext keys, thus compromising the privacy and integrity of the encrypted media stream. Consequently, the SDP exchange must be protected by a security protocol such as IPsec or TLS.

Licensing and Hardware Requirements

SRTP/SRTCP support requires the presence of an IPsec NIU and an SSM (Security Service Module).

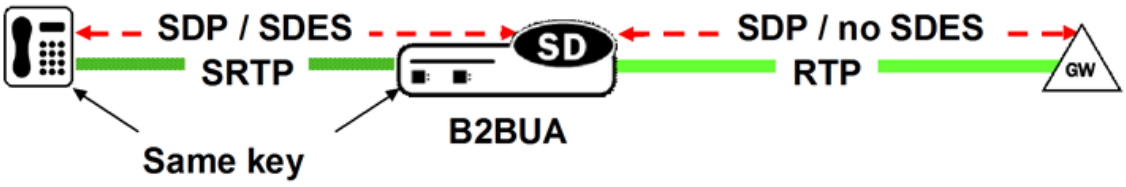
No additional licences are required.

Operational Modes

SRTP topologies can be reduced to three basic topologies which are described in the following sections.

Single-Ended SRTP Termination

Single-ended SRTP termination is illustrated in the following figure.



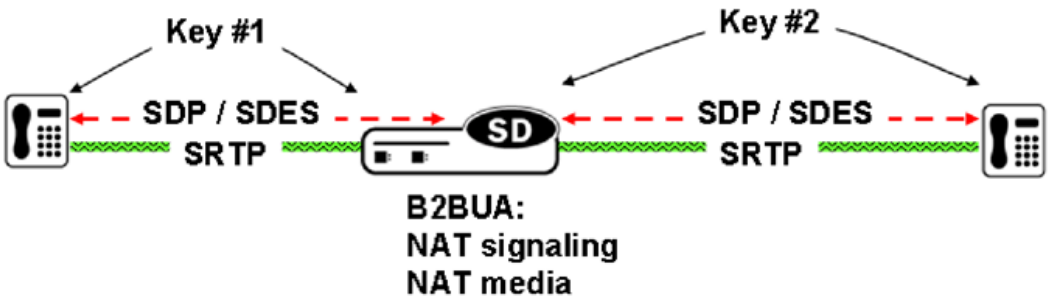
Single-Ended SRTP Termination

If SRTP is enabled for the inbound realm/interface, the Oracle Communications Session Border Controller handles the incoming call as specified by the Media Security Policy assigned to the inbound realm. If there is crypto attribute contained in the offer, the Oracle Communications Session Border Controller parses the crypto attributes and optional parameters, if any. If the offer contains a crypto attribute or attributes compatible with the requirements specified by the SDES profile assigned to the Media Security policy, it selects the most preferred compatible attribute. Otherwise, the Oracle Communications Session Border Controller rejects the offer. Before the SDP is forwarded to the called party, the Oracle Communications Session Border Controller allocates resources, established SRTP and SRTCP Security Associations and updates the SDP by removing the crypto attribute and inserting possibly NAT'ed media addresses and ports. At the same time, the original crypto attribute is also removed from the SDP.

Once the reply from the called party is received, the Oracle Communications Session Border Controller inserts appropriate crypto attribute(s) to form a new SDP, and forward the response back to the calling party.

Back-to-Back SRTP Termination

Back-to-back SRTP termination is illustrated in the following figure.

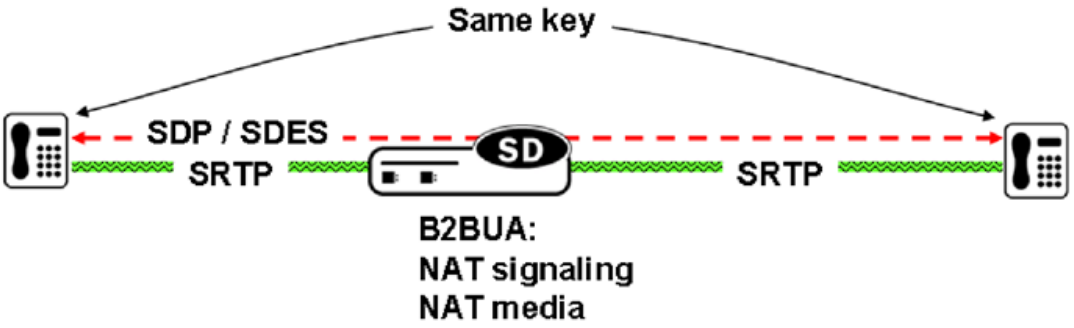


Back-to-Back SRTP Termination

Initial processing is similar to the single-ended termination described above. Before forwarding the request to the called party, the Oracle Communications Session Border Controller replaces the original crypto attribute with a new one whose crypto attribute conforms to the media security policy for the outbound realm/interface. Upon receiving the answer from the called party, the Oracle Communications Session Border Controller accepts or rejects it, again based upon conformity to the media security policy. If accepted, the Oracle Communications Session Border Controller replaces the original crypto attribute from the called party with its own to form a new SDP, which it forwards back to the calling party. At this point, SRTP media sessions are established on both sides for both calling and called parties.

SRTP Pass-Thru

SRTP pass-thru is illustrated in the following figure.



SRTP Pass-Thru

If the media security policy specifies pass-through mode, the Oracle Communications Session Border Controller does not alter the crypto attribute exchange between the calling and the called party; the attribute is transparently passed.

RFC 5939 Support

You can configure the SBC to support RFC 5939-based SDP capability negotiation. This support overrides the supported RFC 3264-based mechanism for generating mixed RTP/SRTP offers to better support secure and non-secure flows in the same realm. Within the RFC 3264 model, both the offer and answer contain actual configurations, but separate capabilities and potential configurations are not supported. The RFC 5939 implementation on the SBC is backward compatible and uses the RFC 3264-based model by default.

RFC 5939 addresses both attribute and transport capability negotiation by offering potential configurations. For transport, this contrasts with the RFC 3264 method, which presents separate RTP and SRTP offers and allows the end-station to set one to port number 0. One of the primary uses of RFC 5939 by the SBC is generating and supporting mixed RTP/SRTP offers.

When configured, the SBC generates RFC 5939-compliant offers when it receives initial INVITE, UPDATE or Re-INVITE messages. These offers contain multiple potential configurations for a media profile. Each potential configuration has a set of capabilities associated with it. The receiver chooses one of the potential configurations and sends it back in answer as the configuration for that media profile. On receiving the answer from the outbound peer, the SBC generates an RFC 5939-compliant answer using the highest priority configuration received in the incoming offer from the inbound peer. If the answer to an RFC 5939-compliant offer is not RFC 5939 compliant, the SBC reverts to the RFC 3264 method.

To enable RFC 5939 compliant offer generation, set the **egress-offer-format** parameter in the applicable **sdes-profile** to **rfc5939-compliant**. In addition, you must set either the applicable inbound or outbound **media-security-policy**, **mode** parameter to **any**.

The SBC does not support RFC3264 mixed mode offers and RFC5939 compliant offer in the same realm. Create a separate realm to support RFC5939 and assign a dedicated **media-security-policy** that has an associated **sdes-profile** with the **egress-offer-format** set to **rfc5939-compliant**, to that realm.

RFC 5939 Operation

The SBC complies with RFC 5939, but its behavior is dependent upon your configuration. This section describes the SBC behaviors relative to specific points in the call flow.

The SBC considers an offer RFC 5939 compliant if it includes a potential configuration attribute (a=pcfg).

Incoming Offer

The SBC processes the incoming initial INVITE, re-INVITE or UPDATE message containing RFC 5939 compliant offer based on the **media-security-policy**, **inbound**, **mode** parameter associated with the inbound realm as follows:

- **rtp**—The SBC does not process any RFC 5939 compliant offer and rejects the session with “488 Not Acceptable Here”.
- **srtp**—The SBC does not process any RFC 5939 compliant offer and rejects the session with “488 Not Acceptable Here”.
- **any**:
  - The SBC only accepts an SDP offer containing an actual or at least one valid potential configuration having RTP/AVP or RTP/SAVP protocol. Otherwise, it rejects the session with a 488 Not Acceptable Here.
  - If all potential configurations present in the received offer contain unsupported transport capability or an unsupported crypto attribute., the SBC processes the received offer as a normal offer/answer.
  - If all potential configurations present in the received offer contain either delete-attributes (a=-m / a=-s / a=-ms) or extension capabilities, the SBC processes the received offer as normal offer/answer
  - The SBC parses and stores one valid, highest priority configuration for both the RTP and SRTP protocols, if available, from the list of potential configurations.

When selecting a potential configuration for SRTP, the priority of the potential configuration takes precedence over the priority of the configured cryptography suites.

Outgoing Offer

The SBC generates the outgoing RFC 5939 compliant offer based on the setting of:

- The **media-security-policy**, **outbound**, **mode** parameter associated with the outbound realm and:
- The **egress-offer-format** in the **sdes-profile** associated with the **media-security-policy**.

SBC behavior, based on the outbound media security policy mode, includes:

- **rtp**:
  - If the incoming offer is in non-RFC 5939 format, the SBC follows RFC 3265 behavior.
  - If the incoming offer is in RFC 5939 format:
    * And the actual or a valid potential configuration contains only RTP/AVP for a media line, the SBC generates a non RFC 5939 format offer with the RTP/AVP media line.
    * And the actual or a valid potential configuration contains only RTP/SAVP, the SBC generates a non RFC 5939 format offer after converting the RTP/SAVP to RTP/AVP media line.
    * And the actual and valid potential configuration contains both RTP/AVP and RTP/SAVP protocol for a media line, the SBC generates a non RFC 5939 format offer with the RTP/AVP media line.
- **srtp**:

  - If the incoming offer is in non-RFC 5939 format, the SBC follows RFC 3264 behavior.
  - If the incoming offer is in RFC 5939 format, the SBC behaves as follows:
    * If the actual or valid potential configuration contains only RTP/SAVP for a media line, the SBC generates a non RFC5939 format offer with an RTP/SAVP media line.
    * If the actual or a valid potential configuration contains only RTP/AVP, the SBC generates a non RFC5939 format offer after converting the RTP/AVP to RTP/SAVP media line.
    * If the actual and valid potential configuration contains both RTP/AVP and RTP/SAVP protocol for a media line, the SBC generates a non RFC 5939 format offer with an RTP/SAVP media line.

- **any**—The SBC creates offer SDP based on the value configured in the **egress-offer-format** set in the **sdes-profile** configuration:
  - If the incoming offer is not in RFC 5939 format, the SBC behaves as follows:
    * If the value is same-as-ingress, the SBC leaves the profile of the media lines unchanged.
    * If the value is simultaneous-best-effort, the SBC behaves as follows:
      * Adds an RTP/SAVP media line for any media profile that has only the RTP/AVP media profile.
      * Adds an RTP/AVP media line for any media profile that has only the RTP/SAVP media profile.
      * Should the media profile in the incoming offer SDP already have two media lines (one for RTP/AVP and on for RTP/SAVP), the SBC does not have to make these additions. Instead, it maps the media lines in the answer it receives with the media lines from the incoming offer SDP. It also ensures that the media lines in the answer SDP it sends match the media lines from the incoming offer SDP.
    * If the value is RFC 5939-compliant, the SBC generates an RFC 5939 compliant offer containing actual configuration with RTP/AVP protocol and potential configuration with RTP/SAVP protocol.
  - If the incoming offer is in RFC 5939 format, the SBC behaves as follows:
    * If the value is **simultaneous-best-effort**, the SBC generates a non RFC5939 compliant offer with two m-lines, one with RTP/AVP and other with RTP/SAVP protocol.
    * If the value is **rfc5939-compliant**, the SBC generates an RFC5939 compliant offer containing the actual configuration with RTP/AVP protocol and a potential configuration with RTP/SAVP protocol.

While generating an RFC 5939 compliant offer, the SBC populates the RFC 5939 specific attributes as follows:

- Adds a transport capability attribute at the session level, appearing as a=tcap:1 RTP/SAVP
- Adds a new potential configuration for each configured crypto attribute at the media level. Examples include:
  - a=acap:1 crypto:1 AES_CM_128_HMAC_SHA1_32 inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj|2^20|1:32
    a=pcfg:1 t=1 a=1

  - a=acap:2 crypto:2 AES_CM_128_HMAC_SHA1_80 inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4 a=pcfg:2 t=1 a=2

> **Note:**
>
> While adding potential configurations, the SBC assigns priority to “a=pcfg” based on the priority of the configured crypto suites in the sdes-profile associated with outgoing media-security-policy.

Incoming Answer

The SBC processes the incoming answer based on the format of the outgoing offer generated by the SBC as follows:

- If the incoming answer is not RFC5939 compliant, then process answer as per normal offer/answer rules as is defined in RFC 3264.
- If the incoming answer is RFC 5939 compliant, then:
  - For each media description in incoming answer, for which a potential configuration was included in outgoing offer, the SBC ensures that the config-number in the actual configuration attribute (“a=acfg”) matches the config-number of the potential configuration attribute (“a=pcfg”) of the outgoing offer. Also the att-cap-num and trpr-cap-num in the actual configuration must match the attribute and transport capability present in the potential configuration of the outgoing offer:
    * If the actual configuration satisfies these conditions, use the capability attributes and transport capability attribute as per actual configuration.
    * If the actual configuration for a media description doesn't satisfy the conditions mentioned in the above point, process answer as per normal offer/answer rules.

> **Note:**
>
> If “a=acfg” attribute in incoming answer contains “t=” with more than one value (say a=acfg:1 t=1,2 or a=acfg:1 t=1|2), the SBC treats the actual configuration as invalid and the answer with normal offer/answer rules.

> **Note:**
>
> If “a=acfg” attribute in incoming answer contains “a=” with more than one value (say a=acfg:1 t=1 a=1,2 or a=acfg:1 t=1 a=1|2), the SBC treats the actual configuration as invalid and treats the answer with normal offer/answer rules.

Outgoing Answer

The SBC generates the outgoing answer based on the format of the incoming offer from the ingress peer and the mode configured in the media-security-policy associated with the inbound realm as follows:

- If the incoming offer was not RFC 5939 compliant, having a single m-line with RTP or SRTP protocol (**media-security-policy** associated with inbound realm has **mode** set to **rtp** or **srtp** or **any**), then the SBC generates a non RFC 939 compliant answer having a

single m-line with the RTP or SRTP protocol based on the configured mode using the answer received from the egress peer.
If the answer received from egress peer is in RFC 5939 format, then the SBC uses the m-line present in the RFC 5939 compliant answer to generate the outgoing answer.

- The following assume you have configured the SBC with the **media-security-policy** associated with inbound realm has **mode** set to **any**.
  - If the incoming offer was not RFC 5939 compliant, having two m-lines with both RTP and SRTP protocols, then the SBC generates a non RFC 5939 compliant answer having two m-lines with the RTP and SRTP protocols based on the configured mode using the answer received from the egress peer.
    If the answer received from the egress peer is in RFC 5939 format, then the SBC uses the m-line present in the RFC 5939 compliant answer to generate the outgoing answer.
  - If the incoming offer was RFC 5939 compliant, then the SBC generates an RFC 5939 compliant answer using the highest priority configuration present in the offer received from the inbound peer and the configuration associated with the inbound realm.
    If outgoing answer is generated based on the actual configuration rather than a potential configuration received in the incoming offer, then the SBC generates an answer that is not RFC 5939 compliant.
  - If the incoming offer was RFC 5939 compliant, but with all unsupported attributes in the potential configuration, then the SBC generates an answer that is not RFC 5939 compliant, using the actual configuration present in the offer received from the inbound peer and the configuration associated with the inbound realm.

RFC 5939 Capability Negotiation Attributes

The SBC can use the RFC 5939 based SDP capability negotiation mechanism in which the SBC generates offers containing multiple potential configurations for a media profile. Each potential configuration has a set of capabilities associated with it. On receiving an answer containing an actual configuration, selected from potential configurations by outbound peer, the SBC generates an RFC 5939 compliant answer using the highest priority configuration received in the incoming offer from the inbound peer.

A potential SDP configuration may include attribute capabilities and transport capabilities, transport capabilities only, or some other combination of capabilities. If transport capabilities are not included in a potential configuration, the SBC uses the default transport for that media stream. The actual SDP configuration is the potential configuration that was selected. The selected configuration number and all selected capability numbers used in the actual configuration attribute refer to those from the offer, not the answer.

Key RFC 5939 attribute values used for this support include:

- Capability negotiation—a=creq:<option-tag-list>
- Capability negotiation—a=csup:<option-tag-list>
- Potential config—a=pcfg potential <config-number> [<pot-cfg-list>]
- Actual config—a=acfg: <config-number> [<sel-cfg-list>]

For all attributes, white space is not allowed before config-number. For a=pcfg, a=acfg, a=tcap and a= acap, the attribute numbering should be in the range of 1 to 2^31-1 (both included).

Operational rules on how the SBC uses these attributes include the following:

- Supported Capability Negotiation Extensions Attribute (a=csup)

  - If the incoming offer/answer contains a=csup attribute either at session or media level (one per media description), the SBC ignores it and continues processing the offer/answer.
  - If the incoming offer/answer contains multiple instances of a=csup attribute at either the session or media level, the SBC ignores them and continues processing the offer/answer. The RFC states that you can have only one instance of this attribute at the session or the media level (one per media description).
- Required Capability Negotiation Extensions Attribute (a=creq)
  - If the incoming offer contains a=creq attribute with value other than “cap-v0” either at session or media level (one per media description), the SBC “488 Not Acceptable Here” as the answer.
  - If the creq contains multiple comma separated values with any of the value other than “cap-v0”, the SBC generates a “488 Not Acceptable Here” as the answer.
  - If the offer contains more than one instance of a=creq attribute at either session or media level, the SBC generates a “488 Invalid Session Description” as the answer.
- Transport Capability Attribute (a=tcap)
  - If there is more than one transport capability attribute at the session level or more than one transport capability attribute in any media description, the SBC processes the offer based on normal offer/answer rules.
  - If the tcap attribute at session and media level has same trpr-cap-num, the SBC ignores both the tcap attributes and processes the offer based on normal offer/answer rules.
  - The SBC ignores any transport capability attribute indicating protocol other than RTP/AVP or RTP/SAVP.
- Attribute Capability Attribute (a=acap)
  - The SBC ignores the Attribute capability attribute present at session level.
  - The SBC only supports the crypto parameter for SRTP as the “acap” attribute at media level. In order for crypto parameter to be considered valid, crypto suite should match one of the currently supported suites. The SBC ignores any other parameter at media level.
- Potential Configuration Attribute (a=pcfg)
  - The SBC ignores potential configuration at the session level, if present.
  - If the potential configuration in the incoming offer contains either delete-attributes (a=-m / a=-s / a=-ms) or extension capabilities, the SBC ignores that potential configuration and processes the rest of the potential configurations.
  - If the transport protocol configuration list in the potential configuration contains transport protocol other than RTP/AVP or RTP/SAVP, the SBC ignores that potential configuration and processes the rest of the potential configurations.
  - If the transport capability is not present in potential configuration, the SBC uses the transport capability specified in the m-line for that potential configuration.
  - If the potential configuration in incoming offer contains comma separated values in “a=” or “t=”, the SBC ignores that potential configuration and processes the rest of the potential configurations.
- Actual Configuration Attribute (a=acfg)
  - The SBC ignores actual configuration attributes if present at the session level.

- If multiple instances of the actual configuration attribute are present for a media description, the SBC ignores all except the first instance.
- If the actual configuration in the received answer contains either delete-attributes (a=-m / a=-s / a=-ms) or extension capabilities, the SBC ignores the actual configuration and uses normal offer/answer rules.
- If the actual configuration in the received answer contains multiple transport protocols (“t=”) in pipe separated or comma separated form, the SBC ignores the actual configuration and uses normal offer/answer rules.
- If the actual configuration in the received answer contains multiple capabilities (“a=”) in pipe separated or comma separated form, the SBC ignores the actual configuration and uses normal offer/answer rules.

SDES Configuration

SDES configuration consists of the following steps.

1. Create one or more SDES profiles which specify parameter values negotiated during the offer/answer exchange.
2. Create one or more Media Security Policies that specify key exchange protocols and protocol-specific profiles.
3. Assign a Media Security Policy to a realm.
4. Create an interface-specific Security Policy.

SDES Profile Configuration

An SDES profile specifies the parameter values offered or accepted during SDES negotiation.

To configure SDES profile parameters:

1. From superuser mode, use the following command sequence to access sdes-profile configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

2. Use the required **name** parameter to provide a unique identifier for this sdes-profile instance.

   name enables the creation of multiple sdes-profile instances.

3. Use the **crypto-suite** parameter to select the encryption and authentication algorithms accepted or offered by this sdes-profile.

   Allowable values are:

   AES_CM_128_HMAC_SHA1_80 (the default value)

   supports AES/128 bit key for encryption and HMAC/SHA-1 80-bit digest for authentication

   AES_CM_128_HMAC_SHA1_32

   supports AES/128 bit key for encryption and HMAC/SHA-1 32-bit digest for authentication

4. Use the **srtp-auth** parameter to enable or disable the authentication of SRTP packets.

5. Use the **srtp-encrypt** parameter to enable or disable the encryption of RTP packets.

   With encryption enabled, the default condition, the Oracle Communications Session Border Controller offers RTP encryption, and rejects an answer that contains an UNENCRYPTED_SRTP session parameter in the crypto attribute.

   With encryption disabled, the Oracle Communications Session Border Controller does not offer RTP encryption and includes an UNENCRYPTED_SRTP session parameter in the SDP crypto attribute; it accepts an answer that contains an UNENCRYPTED_SRTP session parameter.

6. Use the **srtcp-encrypt** parameter to enable or disable the encryption of RTCP packets.

   With encryption enabled, the default condition, the Oracle Communications Session Border Controller offers RTCP encryption, and rejects an answer that contains an UNENCRYPTED_SRTCP session parameter in the crypto attribute.

   With encryption disabled, the Oracle Communications Session Border Controller does not offer RTCP encryption and includes an UNENCRYPTED_SRTCP session parameter in the SDP crypto attribute; it accepts an answer that contains an UNENCRYPTED_SRTCP session parameter.

7. Use the **mki** parameter to enable or disable the inclusion of the MKI:length field in the SDP crypto attribute.

   The master key identifier (MKI) is an optional field within the SDP crypto attribute that differentiates one key from another. MKI is expressed as a pair of decimal numbers in the form: |mki:mki_length| where mki is the MKI integer value and mki_length is the length of the MKI field in bytes. For hardware-based platforms, the length value can be up to 32 bytes. For software-based platforms, the length value is 4 bytes.

   The MKI field is necessary only in topologies that may offer multiple keys within the crypto attribute.

   Allowable values are enabled and disabled (the default).

   enabled – an MKI field is sent within the crypto attribute (16 bytes maximum)

   disabled – no MKI field is sent

8. Use **done**, **exit**, and **verify-config** to complete configuration of this SDES profile instance.
9. Repeat Steps 1 through 8 to configure additional SDES profiles.

Media Security Policy Configuration

Use the following procedure to create a Media Security Policy that specifies the role of the Oracle Communications Session Border Controller in the security negotiation. If the Oracle Communications Session Border Controller takes part in the negotiation, the policy specifies a key exchange protocol and SDES profile for both incoming and outgoing calls.

> **Note:**
>
> The media security policy configuration does not apply to hairpin calls.

To configure media-security-policy parameters:

1. From superuser mode, use the following command sequence to access media-sec-policy configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# media-sec-policy
ORACLE(media-sec-policy)#
```

2. Use the required **name** parameter to provide a unique identifier for this media-sec-policy instance.

   name enables the creation of multiple media-sec-policy instances.

3. Use optional **pass-thru** parameter to enable or disable pass-thru mode.

   With pass-thru mode enabled, the User Agent (UA) endpoints negotiate security parameters between each other; consequently, the Oracle Communications Session Border Controller simply passes SRTP traffic between the two endpoints.

   With pass-thru mode disabled (the default state), the Oracle Communications Session Border Controller disallows end-to-end negotiation — rather the Oracle Communications Session Border Controller initiates and terminates SRTP tunnels with both endpoints.

4. Use the **outbound** navigation command to move to media-sec-outbound configuration mode. While in this configuration mode you specify security parameters applied to the outbound call leg, that is calls sent by the Oracle Communications Session Border Controller.

5. Use the **protocol** parameter to select the key exchange protocol.

   Select sdes for SDES key exchange.

6. Use the **profile** parameter to specify the name of the SDES profile applied to calls sent by the Oracle Communications Session Border Controller.

7. Use the **mode** parameter to select the real time transport protocol.

   Allowable values are rtp and srtp (the default).

   mode identifies the transport protocol (RTP or SRTP) included in an SDP offer when this media-security-policy is in effect.

8. Use the **done** and **exit** parameters to return to media-sec-policy configuration mode.

9. Use the **inbound** navigation command to move to media-sec-inbound configuration mode. While in this configuration mode you specify security parameters applied to the inbound call leg, that is calls received by the Oracle Communications Session Border Controller.

10. Use the **protocol** parameter to select the key exchange protocol.

    Select sdes for SDES.

11. Use the **profile** parameter to specify the name of the SDES profile applied to calls received by the Oracle Communications Session Border Controller.

12. Use the **mode** parameter to select the real time transport protocol.

    Allowable values are rtp and srtp (the default).

    mode identifies the transport protocol (RTP or SRTP) accepted in an SDP offer when this media-security-policy is in effect.

13. Use **done**, **exit**, and **verify-config** to complete configuration of this media security policy instance.

14. Repeat Steps 1 through 13 to configure additional media-security policies.

Assign the Media Security Policy to a Realm

To assign a media-security-policy to a realm:

1. From superuser mode, use the following command sequence to access realm-config configuration mode. While in this mode, you assign an existing media-security-policy to an existing realm.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
identifier:
1. access-12
...
...
selection: 1
ORACLE(realm-config)#
```

2. Use the **media-sec-policy** parameter to assign the policy to the target realm.
3. Use **done**, **exit**, and **verify-config** to complete assignment of the media-security-policy to the realm.

RFC 5939 Configuration

This setting provides support for RFC 5939.

To configure SDES profile parameters:

1. From superuser mode, use the following command sequence to access sdes-profile configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

2. Set the **egress-offer-format** parameter to **rfc5939-compliant**.

```
ORACLE(sdes-profile)#egress-offer-format rfc5939-compliant
```

3. Use **done**, **exit**, and **verify-config** to complete configuration of this compliance.

ACLI Example Configurations

The following section contain relevant sections of system configurations for basic operational modes.

Single-Ended SRTP Termination Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
    name                        sdes1
    crypto-list                 AES_CM_128_HMAC_SHA1_80
    srtp-auth                   enabled
    srtp-encrypt                enabled
    srtcp-encrypt               enabled
    mki                         disabled
    key
    salt
    last-modified-by            admin@console
    last-modified-date          2009-11-16 15:37:13
media-sec-policy
    name                        msp2
    pass-through                disabled
    inbound
        profile                 sdes1
        mode                    srtp
        protocol                sdes
    outbound
        profile                 sdes1
        mode                    srtp
        protocol                sdes
    last-modified-by            admin@console
    last-modified-date          2009-11-16 15:37:51
...
...
...
realm-config
    identifier                  peer
    description
    addr-prefix                 192.168.0.0/16
    network-interfaces          M00:0
    mm-in-realm                 enabled
    mm-in-network               enabled
    mm-same-ip                  enabled
    mm-in-system                enabled
    bw-cac-non-mm               disabled
    msm-release                 disabled
    qos-enable                  disabled
    generate-UDP-checksum       disabled
    max-bandwidth               0
    fallback-bandwidth          0
    max-priority-bandwidth      0
    max-latency                 0
    max-jitter                  0
    max-packet-loss             0
    observ-window-size          0
    parent-realm
```

```
    dns-realm
    media-policy
    media-sec-policy           msp2
    in-translationid
...
...
...
    last-modified-by           admin@console
    last-modified-date         2009-11-10 15:38:19
```

Back-to-Back SRTP Termination Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
   name                        sdes1
   crypto-list                 AES_CM_128_HMAC_SHA1_80
   srtp-auth                   enabled
   srtp-encrypt                enabled
   srtcp-encrypt               enabled
   mki                         disabled
   key
   salt
   last-modified-by            admin@console
   last-modified-date          2009-11-16 15:37:13
media-sec-policy
   name                        msp2
   pass-through                disabled
   inbound
      profile                  sdes1
      mode                     srtp
      protocol                 sdes
   outbound
      profile                  sdes1
      mode                     srtp
      protocol                    sdes
   last-modified-by            admin@console
   last-modified-date          2009-11-16 15:37:51
...
...
...
realm-config
   identifier                  peer
   description
   addr-prefix                 192.168.0.0/16
   network-interfaces          M00:0
   mm-in-realm                 enabled
   mm-in-network               enabled
mm-same-ip                     enabled
   mm-in-system                enabled
   bw-cac-non-mm               disabled
   msm-release                 disabled
   qos-enable                  disabled
```

```
   generate-UDP-checksum      disabled
   max-bandwidth              0
   fallback-bandwidth         0
   max-priority-bandwidth     0
   max-latency                0
   max-jitter                 0
   max-packet-loss            0
   observ-window-size         0
   parent-realm
   dns-realm
   media-policy
   media-sec-policy    msp2
...
...
...
realm-config
   identifier                 core
   description
   addr-prefix                172.16.0.0/16
   network-interfaces         M10:0
   mm-in-realm                enabled
   mm-in-network              enabled
   mm-same-ip                 enabled
   mm-in-system               enabled
   bw-cac-non-mm              disabled
   msm-release                disabled
   qos-enable                 disabled
   generate-UDP-checksum      disabled
   max-bandwidth              0
   fallback-bandwidth         0
   max-priority-bandwidth     0
   max-latency                0
   max-jitter                 0
   max-packet-loss            0
   observ-window-size         0
   parent-realm
   dns-realm
   media-policy
   media-sec-policy           msp2
   in-translationid
...
...
...
   last-modified-by           admin@console
   last-modified-date         2009-11-10 15:38:19
```

SRTP Pass-Thru Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
   name                            sdes1
   crypto-list                     AES_CM_128_HMAC_SHA1_80
```

```
      srtp-auth                       enabled
      srtp-encrypt                    enabled
      srtcp-encrypt                   enabled
      mki                             disabled
      key
      salt
      last-modified-by                admin@console
      last-modified-date              2009-11-16 15:37:13
media-sec-policy
      name                            msp2
      pass-through                    enabled
      inbound
         profile                      sdes1
         mode                         srtp
         protocol                     sdes
      outbound
         profile                      sdes1
         mode                         srtp
         protocol                     sdes
      last-modified-by                admin@console
      last-modified-date              2009-11-16 15:37:51
...
...
...
realm-config
      identifier                        peer
      description
      addr-prefix                       192.168.0.0/16
      network-interfaces              M00:0
      mm-in-realm                       enabled
      mm-in-network                     enabled
      mm-same-ip                        enabled
      mm-in-system                      enabled
      bw-cac-non-mm                     disabled
      msm-release                       disabled
      qos-enable                        disabled
      generate-UDP-checksum             disabled
      max-bandwidth                     0
      fallback-bandwidth                0
      max-priority-bandwidth            0
      max-latency                       0
      max-jitter                        0
      max-packet-loss                   0
      observ-window-size                0
      parent-realm
      dns-realm
      media-policy
      media-sec-policy                   msp2
...
...
...
realm-config
      identifier                        core
      description
      addr-prefix                       172.16.0.0/16
      network-interfaces              M10:0
```

```
mm-in-realm                         enabled
mm-in-network                       enabled
mm-same-ip                          enabled
mm-in-system                        enabled
bw-cac-non-mm                       disabled
msm-release                         disabled
qos-enable                          disabled
generate-UDP-checksum               disabled
max-bandwidth                       0
fallback-bandwidth                  0
max-priority-bandwidth              0
max-latency                         0
max-jitter                          0
max-packet-loss                     0
observ-window-size                  0
parent-realm
dns-realm
media-policy
media-sec-policy                  msp2
in-translationid
...
...
...
last-modified-by                  admin@console
last-modified-date                2009-11-10 15:38:19
```

Security Policy

A Security Policy enables the Oracle Communications Session Border Controller to identify inbound and outbound media streams that are treated as SRTP/SRTCP. The high-priority Security Policy, p1, (shown below) allows signaling traffic from source 172.16.1.3 to destination 172.16.1.10:5060. The lower-priority Security Policy, p2, (also shown below) matches media traffic with the same source and destination, but without any specific ports. Consequently, SIP signaling traffic (from local port 5060) go through, but the media stream will be handled by appropriate SRTP SA.

```
security-policy
     name                              p1
     network-interface                 private:0
     priority                          0
     local-ip-addr-match               172.16.1.3
     remote-ip-addr-match              172.16.1.10
     local-port-match                  5060
     remote-port-match                 0
     trans-protocol-match              UDP
     direction                         both
     local-ip-mask                     255.255.255.255
     remote-ip-mask                    255.255.255.255
     action                            allow
     ike-sainfo-name
     outbound-sa-fine-grained-mask
          local-ip-mask                255.255.255.255
          remote-ip-mask               255.255.255.255
          local-port-mask              0
          remote-port-mask             0
```

```
            trans-protocol-mask          0
            valid                        enabled
            vlan-mask                    0xFFF
      last-modified-by                   admin@console
      last-modified-date                 2009-11-09 15:01:55

security-policy
      name                               p2
      network-interface                  private:0
      priority                           10
      local-ip-addr-match                172.16.1.3
      remote-ip-addr-match               172.16.1.10
      local-port-match                   0
      remote-port-match                  0
      trans-protocol-match               UDP
      direction                          both
      local-ip-mask                      255.255.255.255
      remote-ip-mask                     255.255.255.255
      action                             ipsec
      ike-sainfo-name
      outbound-sa-fine-grained-mask
            local-ip-mask                0.0.0.0
            remote-ip-mask               255.255.255.255
            local-port-mask              0
            remote-port-mask             65535
            trans-protocol-mask          255
            valid                        enabled
            vlan-mask                    0xFFF
      last-modified-by                   admin@console
      last-modified-date                 2009-11-09 15:38:19
```

Modified ALCI Configuration Elements

The action parameter in security-policy configuration mode has been modified to accept additional values, srtp and srtcp.

- From superuser mode, use the following command sequence to access media-sec-policy configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# security-policy
ORACLE(security-policy)# action ?
<enumeration> action (default: ipsec)
ipsec, allow, discard, srtp, srtcp
ORACLE(security-policy)#
```

The **show security** command has been updated with an **srtp** option.

```
ORACLE# show security srtp
sad
spd
statistics
```

```
SRTP Statistics
status
```

The **srtp** option is similar to the **ipsec** option save for the **sad** sub-option that provides data for only SRTP SAs.

The **show sa stats** command has been updated with an **srtp** option.

```
ORACLE# show sa stats
<ENTER>    Show statistics summary of all Security Associations
<ike>    Show statistics for IKE Security Associations
<ims-aka>    Show statistics for IMS-AKA Security Associations
<srtp>    Show statistics for SRTP Security Associations
sd# show sa stats srtp
20:06:24-114
SA Statistics                    ---- Lifetime ----
                           Recent    Total    PerMax
SRTP Statistics
ADD-SA Req Rcvd               0        0        0
ADD-SA Success Resp Sent      0        0        0
ADD-SA Fail Resp Sent         0        0        0
DEL-SA Req Rcvd               0        0        0
DEL-SA Success Resp Sent      0        0        0
DEL-SA Fail Resp Sent         0        0        0
SA Added                      0        0        0
SA Add Failed                 0        0        0
SA Deleted                    0        0        0
SA Delete Failed              0        0        0
```

Increase SSRC changes allowed in a SRTP stream

By default, SBC allows only seven SSRC changes and blocks SRTP streams with new SSRC on the same port. This happens for both audio and video streams. If you revert to plain RTP there are no limitations on the number of SSRC streams on the same RTP port. To increase the limit of SSRC changes allowed by SBC, configure the **allowed-ssrc-change-limit** under **realm-config**.

- Default: 7
- Values: Min : 7 / Max : 15

Adding allowed-ssrc-change-limit option

To change the default value of allowed-ssrc-change-limit:

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# options +allowed-ssrc-change-limit=<value>
```

If you type the option without the plus sign, you overwrite any previously configured options. To append the new option to the options list, prepend the new option with a plus sign as shown in the previous example.

Secure Real-Time Protocol (SRTP) for Software

The Secure Real-Time Transport Protocol, as described in RFC 3711, *The Secure Real-time Transport Protocol (SRTP)*, provides a framework for the encryption and authentication of Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) streams. Both RTP and RTCP are defined by RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*.

Encryption ensures that the call content and associated signalling remains private during transmission.

Authentication ensures the following.

- Received packets are from the purported source
- Packets have not been tampered with during transmission
- A packet has not been replayed by a malicious server

Protocol Overview

While the RFC 3711 framework provides encryption and authentication procedures and defines a set of default cryptographic transforms required for RFC compliance, it does not specify a key management protocol to securely derive and exchange cryptographic keys. RFC 4568, *Session Description Protocol (SDP) Security Description for Media Streams*, defines such a protocol specifically designed to exchange cryptographic material using a newly defined SDP crypto attribute. Cryptographic parameters are established with only a single message or in single round-trip exchange using the offer/answer model defined in RFC 3264, *An Offer/Answer Model with the Session Description Protocol*.

The current release provides support for an initial SDP Security Descriptions (SDES) implementation that generates keys used to encrypt SRTP/SRTCP packets.

Authentication of packets will be added to a subsequent release.

A sample SDP exchange is shown below:

The SDP offerer sends:

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
```

The SDP answerer replies:

```
v=0
o=jill 25690844 8070842634 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
```

```
e=homer@example.com (Homer Simpson)
c=IN IP4 168.2.17.11
t=2873397526 2873405696
m=audio 32640 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
```

The media-level SDP attribute, crypto, describes the cryptographic suite, key parameters, and session parameters for the preceding unicast media line. The crypto attribute takes the form:

a=crypto: tag crypto-suite key-parameter [session-parameters]

tag

The tag field contains a decimal number that identifies a specific attribute instance. When an offer contains multiple crypto attributes, the answer uses the tag value to identify the accepted offer.

In the sample offer the tag value is 1.

crypto-suite

The crypto-suite field contains the encryption and authentication algorithms, either AES_CM_128_HMAC_SHA1_80 or AES_CM_128_HMAC_SHA1_32.

key-parameter

The key-parameter field contains one or more sets of keying material for the selected crypto-suite and it has following format.

```
"inline:" <key||salt> ["|" lifetime] ["|" MKI ":" length]
```

inline is a method and specifies that the crypto material to be used by the offerer is transmitted via the SDP.

The key||salt field contains a base64-encoded concatenated master key and salt.

Assuming the offer is accepted, the key || salt provides the crypto material used by the offerer to encrypt SRTP/SRTCP packets, and used by the answerer to decrypt SRTP/SRTCP packets.

Conversely the key || salt contained in the answer to the offer provides the crypto material used by the answerer to encrypt SRTP/SRTCP packets, and used by the offerer to decrypt SRTP/SRTCP packets.

The lifetime field optionally contains the master key lifetime (maximum number of SRTP or SRTCP packets encoded using this master key).

In the sample offer the lifetime value is 1,048, 576 (220) packets.

The MKI:length field optionally contains the Master Key Index (MKI) value and the MKI length.

The MKI is used only when the offer contains multiple keys; it provides a means to differentiate one key from another. The MKI takes the form of an integer, followed by its byte length when included in SRTP/SRTCP packets. For hardware-based platforms, the length value can be up to 32 bytes. For software-based platforms, the length value is 4 bytes.

In the sample offer the MKI value is 1 with a length of 4 bytes.

The session-parameters field contains a set of optional parameters that may override SRTP session defaults for the SRTP and SRTCP streams.

UNENCRYPTED_SRTP — SRTP messages are not encrypted

UNENCRYPTED_SRTCP — SRTCP messages are not encrypted

UNAUTHENTICATED_SRTP — SRTP messages are not authenticated

When generating an initial offer, the offerer ensures that there is at least one crypto attribute for each media stream for which security is desired. Each crypto attribute for a given media stream must contain a unique tag. The ordering of multiple crypto attributes is significant — the most preferred crypto suite is listed first.

Upon receiving the initial offer, the answerer must either accept one of the offered crypto attributes, or reject the offer in its entirety.

When an offered crypto attribute is accepted, the crypto attribute contained in the answer MUST contain the tag and crypto-suite from the accepted crypto attribute in the offer, and the key(s) the answerer will use to encrypt media sent to the offerer.

The crypto-suite is bidirectional and specifies encryption and authentication algorithms for both ends of the connection. The keys are unidirectional in that one key or key set encrypts and decrypts traffic originated by the offerer, while the other key or key set encrypts and decrypts traffic originated by the answerer.

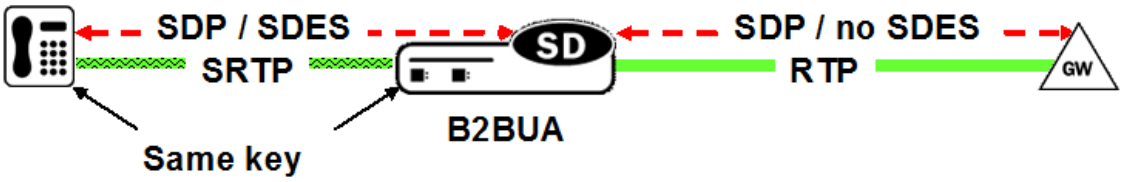
Key exchange via text-based SDP is unacceptable in that malicious network elements could easily eavesdrop and obtain the plaintext keys, thus compromising the privacy and integrity of the encrypted media stream. Consequently, the SDP exchange must be protected by a security protocol such as TLS.

Operational Modes

SRTP topologies can be reduced to three basic topologies which are described in the following sections.

Single-Ended SRTP Termination

Single-ended SRTP termination is illustrated in the following figure.



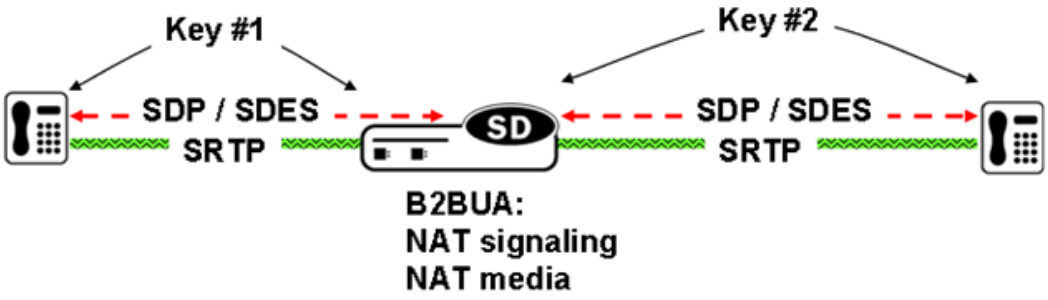
If SRTP is enabled for the inbound realm/interface, the SBC handles the incoming call as specified by the Media Security Policy assigned to the inbound realm. If there is crypto attribute contained in the offer, the SBC parses the crypto attributes and optional parameters, if any. If the offer contains a crypto attribute or attributes compatible with the requirements specified by the SDES profile assigned to the Media Security policy, it selects the most preferred compatible attribute. Otherwise, the SBC rejects the offer. Before the SDP is forwarded to the called party, the SBC allocates resources, established SRTP and SRTCP Security Associations and updates the SDP by removing the crypto attribute and inserting possibly NAT'ed media addresses and ports. At the same time, the original crypto attribute is also removed from the SDP.

Once the reply from the called party is received, the SBC inserts appropriate crypto attribute(s) to form a new SDP, and forward the response back to the calling party.

Refer to ACLI Example Configurations for a sample ACLI configuration.

Back-to-Back SRTP Termination

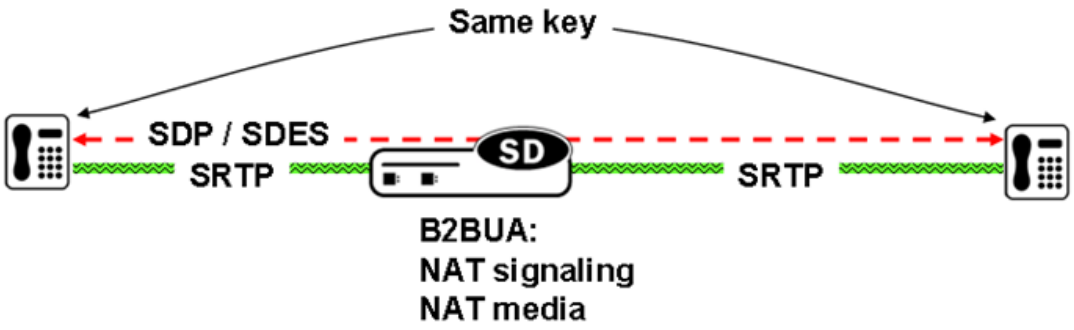
The following figure illustrates Back-to-back SRTP termination.



Initial processing is similar to the single-ended termination described above. Before forwarding the request to the called party, the Oracle Communications Session Border Controller (SBC) replaces the original crypto attribute with a new one whose crypto attribute conforms to the media security policy for the outbound realm/interface. Upon receiving the answer from the called party, the SBC accepts or rejects it, again based upon conformity to the media security policy. If accepted, the SBC replaces the original crypto attribute from the called party with its own to form a new SDP, which it forwards back to the calling party. At this point, SRTP media sessions are established on both sides for both calling and called parties.

SRTP Pass-Thru

The following figure illustrates SRTP pass-thru.



SRTP Pass-Thru

If the media security policy specifies pass-through mode, the SBC does not alter the crypto attribute exchange between the calling and the called party; the attribute is transparently passed.

ACLI Instructions

SDES configuration consists of the following steps.

1. Create one or more SDES profiles which specify parameter values negotiated during the offer/answer exchange.

2. Create one or more Media Security Policies that specify key exchange protocols and protocol-specific profiles.
3. Assign a Media Security Policy to a realm.
4. Create an interface-specific Security Policy (refer to Security Policy for a sample ACLI configuration)

Configure an SDES Profile

A Session Description Protocol Security Descriptions (SDES) profile specifies the parameter values offered or accepted during SDES negotiation.

In the following procedure, use the **Key** and **Salt** parameters to generate the synchronous key used to encrypt and decrypt SRTP/SRTCP traffic originated by theOracle Communications Session Border Controller (SBC). The SBC passes these concatenated values to the remote SRTP peer. Upon reception, the remote peer inputs the key and salt values to the negotiated encryption algorithm (AES in the current implementation), and derives the key required to decrypt SRTP/SRTCP traffic received from the SBC. The **key** parameter provides the basic keying material, while the salt (a bit string) provides the randomness/entropy required by the encryption algorithm.

1. Access the **sdes-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

2. In sdes-profile, do the following:

name	Type the unique name of this profile.
crypto-list	Add one or more cryptography suites to this profile. Default: AES_CM_128_HMAC_SHA1_80. Valid values: • AES_CM_128_HMAC_SHA1_80 • AES_CM_128_HMAC_SHA1_32 • AES_256_CM_HMAC_SHA1_80 • AEAD_AES_256_GCM
srtp-auth	Enable authentication of RTP packets. Default: Enable. Valid values: Enable | Disable.
srtp-encrypt	• Enable to reject an answer that contains an UNENCRYPTED_SRTP session parameter in the crypto attribute. • Disable to not to offer RTP encryption and include an UNENCRYPTED_SRTP session parameter in the SDP crypto attribute and accept an answer that contains an UNENCRYPTED_SRTP session parameter. Default: Enable. Valid values: Enable | Disable.

srtcp-encrypt	• Enable to offer RTCP encryption, and reject an answer that contains an UNENCRYPTED_SRTCP session parameter in the crypto attribute. • Disable to not offer RTCP encryption and include an UNENCRYPTED_SRTCP session parameter in the SDP crypto attribute; accepting an answer that contains an UNENCRYPTED_SRTCP session parameter. Default: Enable. Valid values: Enable | Disable.
mki	Enable or disable the use of the master key identifier within the SDP crypto attribute that differentiates one key from another. • Enable—The SBC sends an MKI field within the crypto attribute (16 bytes maximum). Express MKI as a pair of decimal numbers in the form: | mki:mki_length| where MKI is the MKI integer value and MKI length is the length of the MKI field in bytes. • Disable—The SBC sends no MKI field. Default: disable. Valid values: enable | disable.
egress-offer-format	Set the egress offer format for this profile to use when you also set the outbound mode in the associated media security policy to "any." If the media security policy requires either RTP or SRTP, ignore this parameter. • same-as-ingress—The SBC does not change the profile of the media lines. • simultaneous-best- effort—The SBC inspects the incoming offer SDP, and adds one of the following: – an RTP/SAVP media line for any media profile that has only the RTP/AVP media profile – an RTP/AVP media line for any media profile that has only the RTP/SAVP media profile Default: same as ingress. Valid values: same as ingress | simultaneous best effort.
use-ingress-session-params	Add one or more allowable ingress session parameters. Default: None. Valid values: srtcp-encrypt | srtcp-auth | srtcp-encrypt.
lifetime	Add the lifetime parameter value to a=crypto in the SDP offer. Default: 0 (Do not add lifetime to a=crypto.) Valid values: 20-48. (Express as 2^<value>. For example, using the value 2^20: inline:zYALksQps3ntUw/KsbDdNuxChEQ81Z3BqvTJH|2^20)
options	Add one or more optional features and parameters.
key	Type the master key. (for testing purposes)
salt	Type the master salt. (for testing purposes)
srtp-rekey-on -reinvite	Enable to generate new outbound SRTP keys on every re-invite. Default: Disable. Valid values: Enable | Disable.

3. Type **done** to save the configuration.

Media Security Policy Configuration

Use the following procedure to create a Media Security Policy that specifies the role of the Oracle Communications Session Border Controller (SBC) in the security negotiation. When the SBC takes part in the negotiation, the policy specifies a key exchange protocol and SDES profile for both incoming and outgoing calls.

1. Access the media-sec-policy configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# media-sec-policy
ORACLE(media-sec-policy)#
```

2. Use the required **name** parameter to provide a unique identifier for this media-sec-policy instance.

   name enables the creation of multiple media-sec-policy instances.

3. Use optional **pass-through** parameter to enable or disable pass-thru mode.

   With pass-through mode disabled (the default state), the SBC disallows end-to-end negotiation — rather the Oracle Communications Session Border Controller initiates and terminates SRTP connections with both endpoints.

   With pass-through mode enabled, the SRTP endpoints negotiate security parameters between each other; consequently, the SBC simply relays SRTP traffic between the two endpoints.

4. Use the **outbound** navigation command to move to media-sec-outbound configuration mode. While in this configuration mode you specify security parameters applied to the outbound call leg, that is calls sent by the SBC.

5. Use the **profile** parameter to specify the name of the SDES profile applied to calls sent by the SBC.

6. Use the **mode** parameter to select the real time transport protocol.

   Allowable values are rtp (the default) | srtp | any (either rtp | srtp)

   mode identifies the transport protocol (RTP or SRTP) included in an SDP offer when this media-security-policy is in effect.

7. Use the **protocol** parameter to select the key exchange protocol.

   Select sdes for SDES key exchange.

8. Use the **done** and **exit** parameters to return to media-sec-policy configuration mode.

9. Use the **inbound** navigation command to move to media-sec-inbound configuration mode. While in this configuration mode you specify security parameters applied to the inbound call leg, that is calls received by the SBC.

10. Use the **profile** parameter to specify the name of the SDES profile applied to calls received by the SBC.

11. Use the **mode** parameter to select the real time transport protocol.

    Allowable values are rtp (the default) | srtp | any (either rtp | srtp)

    mode identifies the transport protocol (RTP or SRTP) included in an SDP offer when this media-security-policy is in effect.

12. Use the **protocol** parameter to select the key exchange protocol.

    Select sdes for SDES key exchange.

13. Use **done**, **exit**, and **verify-config** to complete configuration of this media security policy instance.
14. Repeat Steps 1 through 13 to configure additional media-security policies.

Assign the Media Security Policy to a Realm

To assign a media-security-policy to a realm:

1. From superuser mode, use the following command sequence to access realm-config configuration mode. While in this mode, you assign an existing media-security-policy to an existing realm.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
identifier:
1. access-12
...
...
selection: 1
ORACLE(realm-config)#
```

2. Use the **media-sec-policy** parameter to assign the policy to the target realm.
3. Use **done**, **exit**, and **verify-config** to complete assignment of the media-security-policy to the realm.

ACLI Example Configurations

The following section contain XML representations of system configurations for basic operational modes.

Single-Ended SRTP Termination Configuration

```
<sdesProfile name='sdes'
    srtpAuth='enabled'
    srtpEncrypt='enabled'
    srtcpEncrypt='enabled'
    mki='disabled'
    egressOfferFormat='same-as-ingress'
    useIngressSessionParams=''
    options=''
    key=''
    salt=''
    lastModifiedBy='admin@172.30.11.55'
    lastModifiedDate='2013-03-04 19:29:40' objectId='21'>
    <cipherSuites name='AES_CM_128_HMAC_SHA1_80'/>
    key>sdes</key>
</sdesProfile>

<mediaSecPolicy name='sdes'
```

```
    passThrough='disabled'
    options=''
    lastModifiedBy='admin@172.30.11.55'
    lastModifiedDate='2013-03-04 19:30:23' objectId='22'>

    <inbound profile='sdes'
        mode='srtp'
        protocol='sdes'/>

    <outbound profile='sdes'
        mode='sdes'
        protocol='sdes'/>

    key>sdes</key>
        </mediaSecPolicy>

...
...
...
realm-config
    identifier                peer
    description
    addr-prefix               192.168.0.0/16
    network-interfaces        M00:0
    mm-in-realm               enabled
    mm-in-network             enabled
    mm-same-ip                enabled
    mm-in-system              enabled
    bw-cac-non-mm             disabled
    msm-release               disabled
    qos-enable                disabled
    generate-UDP-checksum     disabled
    max-bandwidth             0
    fallback-bandwidth        0
    max-priority-bandwidth    0
    max-latency               0
    max-jitter                0
    max-packet-loss           0
    observ-window-size        0
    parent-realm
    dns-realm
    media-policy
    media-sec-policy          msp2
    in-translationid
    ...
    ...
    ...
    last-modified-by          admin@console
    last-modified-date        2009-11-10 15:38:19
```

> **Note:**
>
> Whenever there is a RTP for ingress leg and SRTP for egress leg, add an RTP media-sec-policy for single-ended SRTP termination configuration that adds a=crypto line to the Oracle Communications Session Border Controller.

Back-to-Back SRTP Termination Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
     name                            sdes1
     crypto-list                     AES_CM_128_HMAC_SHA1_80
     srtp-auth                       enabled
     srtp-encrypt                    enabled
     srtcp-encrypt                   enabled
     mki                             disabled
     key
     salt
     last-modified-by                admin@console
     last-modified-date              2009-11-16 15:37:13
media-sec-policy
     name                            msp2
     pass-through                    disabled
     inbound
          profile                    sdes1
     mode                            srtp
     protocol                        sdes
     outbound
          profile                    sdes1
     mode                            srtp
     protocol                        sdes
     last-modified-by                admin@console
     last-modified-date              2009-11-16 15:37:51
...
...
...
realm-config
     identifier                      peer
     description
     addr-prefix                     192.168.0.0/16
     network-interfaces    M00:0
     mm-in-realm    enabled
     mm-in-network                   enabled
     mm-same-ip                      enabled
     mm-in-system                    enabled
     bw-cac-non-mm                   disabled
     msm-release                     disabled
     qos-enable                      disabled
     generate-UDP-checksum           disabled
     max-bandwidth                   0
```

```
        fallback-bandwidth              0
        max-priority-bandwidth          0
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        media-sec-policy                msp2
        in-translationid
        ...
        ...
        ...
realm-config
        identifier                      backOffice
        description
        addr-prefix                        172.16.0.0/16
        network-interfaces    M10:0
        mm-in-realm                     enabled
        mm-in-network                   enabled
        mm-same-ip                      enabled
        mm-in-system                    enabled
        bw-cac-non-mm                   disabled
        msm-release                     disabled
        qos-enable                      disabled
        generate-UDP-checksum           disabled
        max-bandwidth                   0
        fallback-bandwidth              0
        max-priority-bandwidth          0
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        media-sec-policy                msp2
        in-translationid
        ...
        ...
        ...
        last-modified-by               admin@console
        last-modified-date             2009-11-10 15:38:19
```

SRTP Pass-Thru Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
        name                            sdes1
        crypto-list                     AES_CM_128_HMAC_SHA1_80
        srtp-auth                       enabled
```

```
        srtp-encrypt                enabled
        srtcp-encrypt               enabled
        mki                         disabled
        key
        salt
        last-modified-by            admin@console
        last-modified-date          2009-11-16 15:37:13
        media-sec-policy
        name                        msp2
        pass-through                enabled
        inbound
        profile                     sdes1
        mode                        srtp
        protocol                    sdes
        outbound
        profile                     sdes1
        mode                        srtp
        protocol                    sdes
        last-modified-by            admin@console
        last-modified-date          2009-11-16 15:37:51
        ...
        ...
        ...
    realm-config
        identifier                  peer
        description
        addr-prefix                 192.168.0.0/16
        network-interfaces          M00:0
        mm-in-realm                 enabled
        mm-in-network               enabled
        mm-same-ip                  enabled
        mm-in-system                enabled
        bw-cac-non-mm               disabled
        msm-release                 disabled
        qos-enable                  disabled
        generate-UDP-checksum       disabled
        max-bandwidth               0
        fallback-bandwidth          0
        max-priority-bandwidth      0
        max-latency                 0
        max-jitter                  0
        max-packet-loss             0
        observ-window-size          0
        parent-realm
        dns-realm
        media-policy
        media-sec-policy            msp2
        ...
        ...
        ...
    realm-config
        identifier                  core
        description
        addr-prefix                 172.16.0.0/16
        network-interfaces          M10:0
        mm-in-realm                 enabled
```

```
        mm-in-network                   enabled
        mm-same-ip                      enabled
        mm-in-system                    enabled
        bw-cac-non-mm                   disabled
        msm-release                     disabled
        qos-enable                      disabled
        generate-UDP-checksum           disabled
        max-bandwidth                   0
        fallback-bandwidth              0
        max-priority-bandwidth          0
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        media-sec-policy                msp2
        in-translationid
        ...
        ...
        ...
        last-modified-by                admin@console
        last-modified-date              2009-11-10 15:38:19
```

Security Policy

A Security Policy enables the Oracle Communications Session Border Controller to identify inbound and outbound media streams that are treated as SRTP/SRTCP. The high-priority Security Policy, p1, (shown below) allows signaling traffic from source 172.16.1.3 to destination 172.16.1.10:5060. The lower-priority Security Policy, p2, (also shown below) matches media traffic with the same source and destination, but without any specific ports. Consequently, SIP signaling traffic (from local port 5060) go through, but the media stream will be handled by appropriate SRTP SA.

```
security-policy
     name                            p1
     network-interface               private:0
     priority                        0
     local-ip-addr-match             172.16.1.3
     remote-ip-addr-match            172.16.1.10
     local-port-match                5060
     remote-port-match               0
     trans-protocol-match            UDP
     direction                       both
     local-ip-mask                   255.255.255.255
     remote-ip-mask                  255.255.255.255
     action                          allow
     ike-sainfo-name
     outbound-sa-fine-grained-mask
          local-ip-mask              255.255.255.255
          remote-ip-mask             255.255.255.255
          local-port-mask            0
          remote-port-mask           0
          trans-protocol-mask        0
```

```
        valid                     enabled
        vlan-mask                 0xFFF
    last-modified-by              admin@console
    last-modified-date            2009-11-09 15:01:55

    security-policy
    name                          p2
    network-interface             private:0
    priority                      10
    local-ip-addr-match           172.16.1.3
    remote-ip-addr-match          172.16.1.10
    local-port-match              0
    remote-port-match             0
    trans-protocol-match          UDP
    direction                     both
    local-ip-mask                 255.255.255.255
    remote-ip-mask                255.255.255.255
    action                        ipsec
    ike-sainfo-name
    outbound-sa-fine-grained-mask
    local-ip-mask                 0.0.0.0
    remote-ip-mask                255.255.255.255
    local-port-mask               0
    remote-port-mask              65535
    trans-protocol-mask           255
    valid                         enabled
    vlan-mask                     0xFFF
    last-modified-by              admin@console
    last-modified-date            2009-11-09 15:38:19
```

SRTP Re-keying

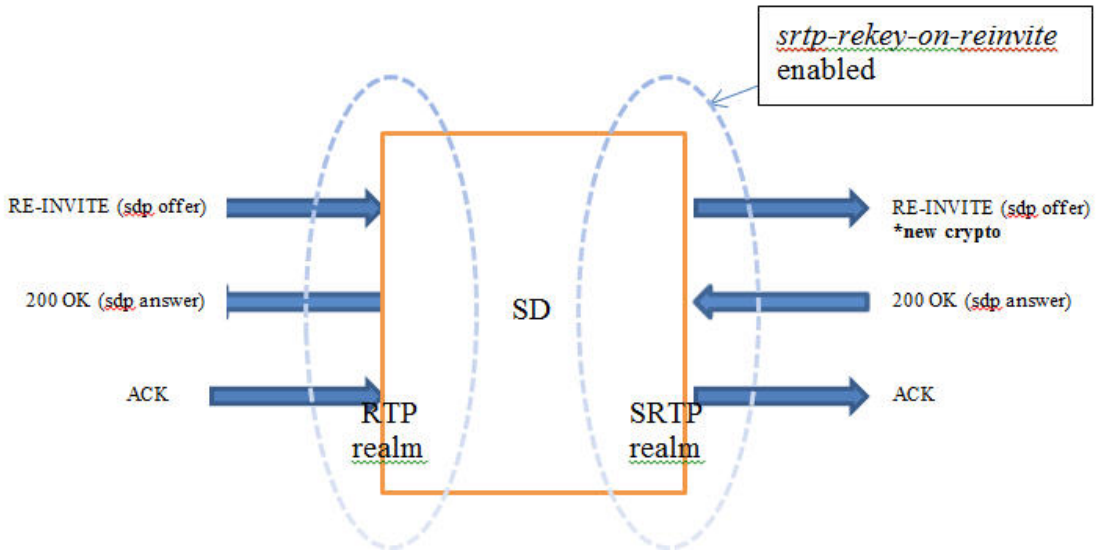
Initialization of SRTP re-keying is supported by the Oracle Communications Session Border Controller.

The Oracle Communications Session Border Controller can generate a new outbound crypto attribute in the SDP offer in a SIP re-INVITE when the **srtp-rekey-on-reinvite** parameter is set to **enabled**. The system generates the attribute regardless of the state of the flow, active or not.

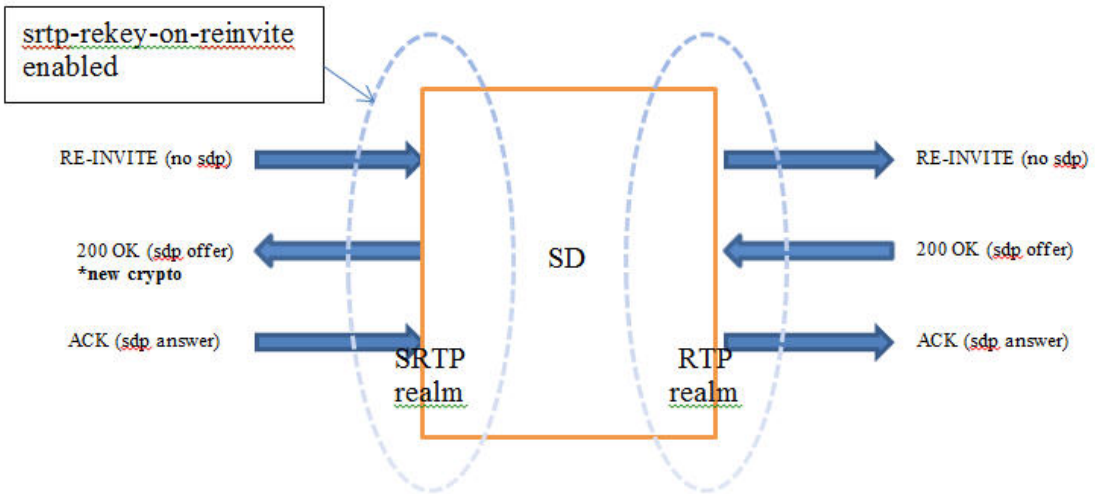
This capability is important for some clients that reside on the SRTP side in a single SRTP termination mode configuration. Any media changes that happen in the RTP side are hidden by the Oracle Communications Session Border Controller. This concealment may cause issues in some configurations, where media servers are involved. When the media changes from media server to called phone, the SRTP endpoint is not aware the media source changed because the SDP offer from the Oracle Communications Session Border Controller is the same as original invite. The result is that some devices drop packets because of Synchronization Source Identifier (SSRC) values mismatch, unexpected jumps in sequence number, sequence number reversions back to 1 triggering replay attack defense, and so forth. In certain environment is has been found that re-keying on every re-invite eliminates all these issues especially in customer setups that use Microsoft Lync products.

The processing of standard RE-INVITES (those containing an SDP offer) and offerless RE-INVITES is shown below.

With SDP:



No SDP:



If the re-invite message is a refresh and **srtp-rekey-on-reinvite** is enabled, the outbound crypto will change but the SDP version will not be incremented on the outgoing invite. If this scenario causes incompatibility issues with customer equipment then add the unique-sdp-id option to **media-manager**, **option** configuration so the Oracle Communications Session Border Controller increments the SDP version in the outgoing invite.

SRTP Re-keying Configuration

Configure **srtp-rekey-on-reinvite** to enable the negotiation and generation of new SRTP keys upon the receipt of a SIP RE-INVITE message that contains SDP.

Confirm that an **sdes-profile** exists.

In the following procedure, change the default state to enabled.

1. Access the **sdes-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
```

```
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

2. Type **select** to choose and configure an existing object.

```
ORACLE(sdes-profile)# select
<name>:
1:  name=sdesprofile01

selection: 1
ORACLE(sdes-profile)#
```

3. **srtp-rekey-on-reinvite**—Set this parameter to **enabled** for re-keying upon the receipt of an SIP reINVITE that contains SDP.
4. Type **done** to save your configuration.

Modified ALCI Configuration Elements

The action parameter in security-policy configuration mode has been modified to accept additional values, srtp and srtcp.

- From superuser mode, use the following command sequence to access media-sec-policy configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# security-policy
ORACLE(security-policy)# action ?
<enumeration> action (default: ipsec)
ipsec, allow, discard, srtp, srtcp
ORACLE(security-policy)#
```

Refer to Security Policy for sample Security Policies.

The **show security** command has been updated with an **srtp** option.

```
ORACLE# show security srtp
sad
spd
statistics
SRTP Statistics
status
```

The **srtp** option is similar to the **ipsec** option save for the **sad** sub-option that provides data for only SRTP SAs.

The **show sa stats** command has been updated with an **srtp** option.

```
ORACLE# show sa stats <ENTER>    Show statistics summary of all Security
Associations
<ike>       Show statistics for IKE Security Associations
<ims-aka>   Show statistics for IMS-AKA Security Associations
<srtp>      Show statistics for SRTP Security Associations
sd# show sa stats srtp
```

```
20:06:24-114
SA Statistics                    ---- Lifetime ----
                              Recent     Total      PerMax
SRTP Statistics
ADD-SA Req Rcvd               0          0          0
ADD-SA Success Resp Sent      0          0          0
ADD-SA Fail Resp Sent         0          0          0
DEL-SA Req Rcvd               0          0          0
DEL-SA Success Resp Sent      0          0          0
DEL-SA Fail Resp Sent         0          0          0
```

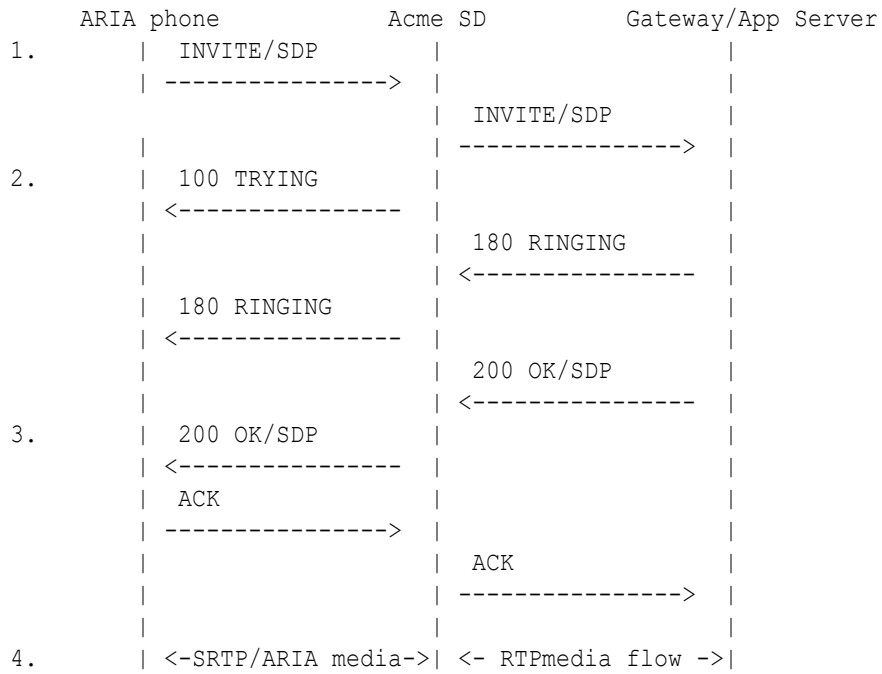
ARIA Cipher Support

Previous and the current Oracle Communications Session Border Controller releases have provided support for the Secure Real-Time Transport Protocol (SRTP), as defined in RFC 3711, to encrypt and authenticate Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol (RTCP) media streams. Concurrent support for Session Description Protocol Security Descriptions (SDES) enabled the exchange of SRTP keying material. These releases have supported a single encryption algorithm, Advanced Encryption System (AES) counter mod with 128-bit keys

This release supports ARIA, a block cipher selected by the Korean Agency for Technology and Standards as a standard cryptographic Technique. The Oracle Oracle Communications Session Border Controller now supports the ARIA cipher with a 192-bit key in counter mode for RTP and RTCP encryption; authentication is supported by HMAC_SHA1 with either 32-bit or 80-bit keys.

Call Flow

An example call flow between a ARIA endpoint, the OracleSD, and a Gateway/Application Server illustrates a successful call establishment where the call is originated from an ARIA enabled phone and destined to a core network server.



1. The ARIA-enabled phone sends an INVITE request to the SD with the crypto attribute in the SDP specifying the ARIA 192 CM cipher for encryption and HMAC_SHA1_80 for authentication. The crypto attribute also has the master key encoded in base-64 format, as well the mki parameters (optionally). The SD forwards the INVITE to the called party via the gateway according to the media-security-policy on the outbound realm.
2. The SD sends provisional response to INVITE request
3. Assuming that the SD gets successful answer from called party, the SD sends a 200 OK response to the caller, with the crypto attribute in the accompanying SDP specifying the ARIA 192 CM cipher for encryption and HMAC_SHA1_80 for authentication. The crypto attribute also has the master key, as well the mki parameters (optionally).
4. The ARIA-enabled phone acknowledges the reception of 200 OK final response. At this point, encrypted SRTP traffic using the ARIA 192 counter mode cipher flows between the phone and the SD, and unencrypted traffic flows between the SD and the core network.

ARIA Support Configuration

ARIA support is enabled at the sdes-profile level.

1. Use the following command sequence to move to **sdes-profile** Configuration Mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(sdes-profile)#
```

2. Use the **crypto-list** parameter to specify the crypto suite used for SDES-based encryption.

   Use either aria_cm_192_hmac_32, or aria_cm_192_hmac_32 to specify ARIA encryption.

```
ORACLE(sdes-profile)# crypto-list aria_cm_192_hmac_80
ORACLE(sdes-profile)#
```

3. Use **done**, **exit**, and **verify-config** to complete cipher suite selection.

DTLS-SRTP

The Oracle Communications Session Border Controller (SBC) supports Datagram Transport Layer Security (DTLS) to establish SRTP media traffic over UDP. You configure DTLS-SRTP security profiles and apply them to the realms that include end stations that request this security. The SBC uses DTLS within the context of SRTP (DTLS-SRTP) per RFC 5764. This DTLS-SRTP feature provides for secure media, supports the same transfer scenarios supported for SDES-SRTP, and also supports unattended transfer, and music on hold scenarios.

DTLS operation on the SBC is equivalent to SDES operation. It provides security against common evesdropping, tampering and message forgery. DTLS can operate over UDP. The use of UDP can eliminate some delay associated with connection protocols.

DTLS-SRTP differs from previous attempts to secure media traffic where the authentication and key exchange protocol, such as with Multimedia Internet KEYing (MIKEY) RFC3830, is piggybacked in the signaling message exchange. With DTLS-SRTP, establishing the protection of the media traffic between the endpoints is done by the media endpoints with only a cryptographic binding of the media keying to the SIP/SDP communication.

DTLS-SRTP Overview

RFC 5763 specifies using DTLS with SRTP, called DTLS-SRTP. Within this architecture, DTLS, based on TLS, provides key management, negotiation of parameters, and secure data transfer. SRTP provides confidential message authentication and replay security. Combining these protocols as SRTP-DTLS establishes fully secure SRTP flows.

Key exchange via text-based SDP is unacceptable in that malicious network elements can easily eavesdrop and obtain plaintext keys, compromising the privacy and integrity of the encrypted media stream. Consequently, the SDP exchange must be protected by a security protocol.

While the SRTP framework provides encryption and authentication procedures and defines the set of default cryptographic transforms required for RFC compliance, it does not specify a key management protocol to securely derive and exchange cryptographic keys. Similar to SDES deployment, these missing functions need to come from another mechanism. DTLS-SRTP, defined by RFC 5763 and RFC 5764 can provide this means of implementing key management for SRTP.

On the SBC, DTLS-SRTP operation begins with the caller issuing a SIP INVITE with SDP parameters that requests a DTLS exchange between the end stations. The callee processes the SIP signaling and SDP request, ultimately issuing a SIP 200 OK to the caller, which is acknowledged. At this point, the DTLS server begins a DTLS handshake sequence between the media endpoints, within which the end stations confirm each others' identity and establish the cryptography to be used for each flow. Once confirmed, the end stations begin exchanging SRTP media.

DTLS-SRTP secures flows between itself and both the caller and callee. The architecture establishes a client-server relationship. Mutual authentication is required. Although it supports features such as early media, the architecture supports an active station tearing down the call if authentication from the other side fails.

The architecture also uses certificates as a means of confirming identity for both the signaling and media flows. These certificates can be self-signed and do not refer to an authority for confirmation. Instead, the end stations hash the certificates and create a fingerprint for use by the opposing end station to verify that the same end-station performing the signaling is also the source of the media. Finally, the architecture establishes the crypto-suite and exchange keys to be used to encrypt and decrypt each flow.

SBC Support for DTLS-SRTP

The SBC aligns with the applicable standards to support DTLS-SRTP, including RFC 5763 and RFC 5764. Alignment with these standards defines much of the SBC behavior with respect to supporting DTLS for SRTP.

The SBC supports broad, standards-based requirements including:

- Authenticating the endpoints using fingerprints
- DTLS Extension to Establish Keys for the SRTP
- Support of the "use_srtp” extension without an MKI value defined
- Support for multiple media sessions from an endpoint, such as audio and video, with separate DTLS session establishment exchange per media session
- Using information within SIP/SDP sessions to identify DTLS-SRTP sessions

- Send DTLS alert message if the system terminates the call because it does not support any of the SRTP protection profiles offered by the client.

Important SBC behavior related to DTLS-SRTP includes:

- If the peer's DTLS certificate matches the peer's a=fingerprint If the validation is successful, then the system stops the **dtls-complete-timeout**. If this validation fails, the system terminates the media session.
- Whenever the **dtls-complete-timeout** expires, the system terminates the media session.
- The system drops any SRTP/SRTCP packets it receives before it has processed the DTLS-SRTP keys.
- The system rejects an SDP media session offer that includes "a=fingerprint" but does not include the a=setup attribute.
- The system must have a **dtls-srtp-profile** on any realm that must be able to process or generate an SDP offer for DTLS-SRTP. For example, if there is no profile at the ingress, the system rejects the request with a 503, service unavailable message.
- The system provides both ingress and egress interworking support between DTLS-SRTP and RTP or SDES-SRTP.

Within the context of the DTLS setup procedure, the SBC utilizes the following over the specified paths:

- Signaling Path
  - Recognizing and signaling the use of DTLS in the SDP m-line over the signaling path by the parameter "UDP/ TLS/RTP/SAVP"
  - Specifying the server role to setup the media path using in "a=setup:passive" SDP
  - Presenting the fingerprint of the end station in the SDP a=fingerprint line
  - Specifying the media path using the IP address and the port exchanged in the offer/ answer SDP in the “c=” and “m=” lines
- Media Path
  - Specifying the use of DTLS-SRTP with the use_srtp extension
  - Providing its certificate for verification within the Hello exchange
  - Using the cipher suite negotiated in the client and server Hello exchange to encrypt/ decrypt the handshake messaging
  - Complying with DTLS requirement for mutual authentication
  - Using the negotiated encryption and decryption keys to encrypt/decrypt the media

The SBC supports the following SRTP encryption and authentication algorithms using the DTLS-SRTP protection profiles defined in RFC 5764:

- AES_CM_128_HMAC_SHA1_80 — Using the SRTP_AES128_CM_HMAC_SHA1_80 profile
- AES_CM_128_HMAC_SHA1_32 — Using the SRTP_AES128_CM_HMAC_SHA1_32 profile
- AEAD_AES_256_GCM

Configuring DTLS-SRTP

You configure DTLS-SRTP on the SBC by setting up **dtls-srtp-profile** sub-element, within **media-security**.

dtls-srtp-profile configuration includes:

- **name**—The name of this profile, which you enter to the **dtls-srtp-profile** parameter on a **realm-config**.

  ```
  ORACLE(dtls-srtp-profile)#name MyDtlsProfile
  ```

- **tls-profile**—The name of the **tls-profile** profile you use within this **dtls-srtp-profile**. On each realm that has a configured **dtls-srtp-profile**, the SBC includes the session attribute a=fingerprint in SDP offers. This fingerprint is the output of a hash function calculated over the certificate that the SBC presents during the DTLS handshake. You configure this certificate within a **tls-profile** and apply it to the **dtls-srtp-profile**.

  > **Note:**
  >
  > This session-level fingerprint attribute applies to both audio and video sessions for this realm, used when the SBC functions as an offerer and answerer for both DTLS connections.

- **dtls-completion-timeout**—Compliant with RFC 6347, this timeout establishes the time for the full DTLS handshake completion, consolidating timeouts for all the handshake messages. The SBC starts this timer when it receives an answer SDP from the callee, and before sending an answer back to the caller. If the DTLS handshake is completed successfully within the configured **dtls-completion-timeout** value, the SBC cancels the timer. If the DTLS handshake is not completed successfully within the timeout, the SBC tears down the media flows as part of flow guard timer processing and clears the call by sending a BYE request to both legs.
  The SBC restarts this timer for any new DTLS association.
- **preferred-setup-role**—Specifies the role the SBC should perform for DTLS handshakes. The system only allows for the **passive** setting, which establishes itself as a server.
- **crypto-suite**—Specifies the SRTP encryption and authentication algorithms the SBC offers for this flow's media. This setting is not associated with the identity verification fingerprint.

After creating you profile, you apply it to the applicable **realm-config**.

```
ORACLE(realm-config)#dtls-srtp-profile MyDtlsProfile
```

> **Note:**
>
> This version of the SBC requires successful rtcp-mux (RTCP multiplexing) negotiation during DTLS/SRP negotiations. As such, you must enable the **dtls-srtp-profile** parameter on each realm for which you have configured a **dtls-srtp-profile**.
>
> ```
> ORACLE(realm-config)#rtcp-mux enabled
> ```
>
> The SBC does not support non-rtcp-mux flows and rejects calls that do not include successful rtcp-mux negotiation.

Related Configuration

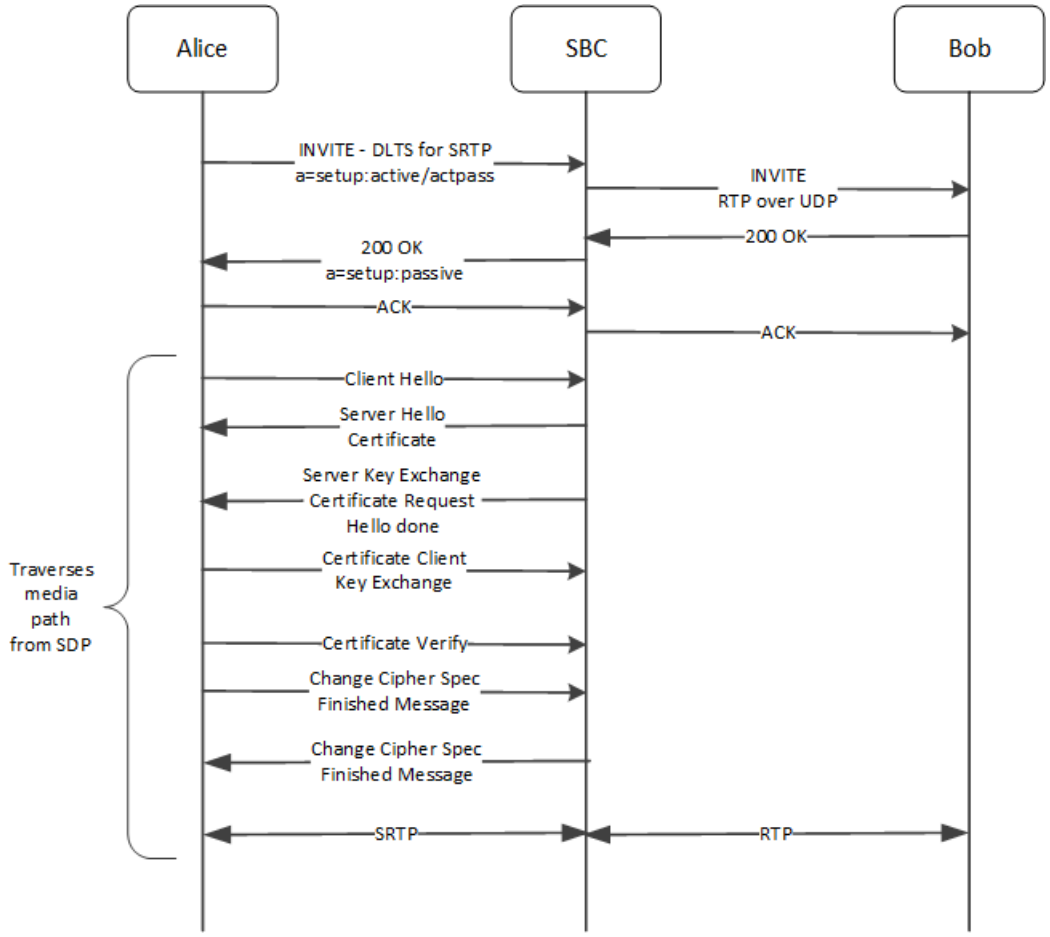
Related configuration that impacts DTLS operation includes:

- When you need different SRTP B2BUA termination by the system on both ingress and egress realms, such as SDES-SRTP and DTLS-SRTP on opposite sides, you must configure a **media-sec-policy** on both the ingress and egress realms.
  - For DTLS-SRTP at an ingress realm and SDES-SRTP at the egress realm, you must configure an applicable **media-sec-policy** and a **dtls-srtp-profile** on the ingress realm. In addition:
    - Ingress **media-sec-policy**—Set **mode** to **any** and **protocol** set to **none**
    - Egress **media-sec-policy**—Set **mode** to **srtp**, and **protocol** set to **sdes**.
  - For SDES-SRTP at an ingress realm and DTLS-SRTP at the egress realm, you must configure an applicable **media-sec-policy** and a **dtls-srtp-profile** on the egress realm. In addition:
    - Ingress **media-sec-policy**—Set **mode** to **srtp**, and **protocol** set to **sdes**.
    - Egress **media-sec-policy**—Set **mode** to **any** and **protocol** set to **none**
  - A **media-sec-policy** is not required when both sides are DTLS-SRTP.
- If the SBC receives an SDP answer with 2 m-lines that includes DTLS and any other protocol, such as SDES, for the same media type, it terminates the SIP session, clearing the call by sending BYE with reason header with cause code “488 Not Acceptable Here”.
- If you configure SDES on a realm, the SBC always includes the session attribute a=crypto in the SDP offer it sends.
- The SBC accepts only DTLS messages for the port number on which it advertises SRTP/SRTCP for media flows that:
  - had DTLS offered by the peer
  - had DTLS accepted by the peer
  - the SBC has offered DTLS has but has not yet received an SDP answer

Example DTLS-SRTP Flows

The call flows below present the SBC handling DTLS-SRTP signaling within the context of SIP calls and DTLS media security setup signaling. Note that these flows begin with ingress flowing to the SBC (eg, from Alice) and egress flowing from the SBC (eg, to Bob).

In this flow, the SBC acts as a B2BUA supporting Alice as a DTLS server and Bob as an RTP leg end point. This call flow assumes that the end points are not behind a NAT and no error is encountered during negotiation. The description does not cover the SIP signaling that sets up RTP with Bob. The flow diagram does not include some of the SIP signaling for brevity.



The call flow begins with Alice setting up a DTLS-SRTP call to Bob, with the SBC acting as B2BUA between them. Steps 1 through 3 take place on the SIP signaling paths; the remaining steps take place on the media path.

1. Alice signals, using the “m” line, that media transport uses DTLS-SRTP. The “a=setup:active” attribute indicates that Alice is the DTLS client and, therefore, initiates the DTLS connection for the media key exchange.
   In addition, Alice calculates the hash of its certificate and populates it in the “a=fingerprint” attribute.

```
INVITE sip:2001@192.168.123.16:5060 SIP/2.0
Via: SIP/2.0/udp 192.168.123.2:5060;branch=z9hG4bK-2
From: 1001 <sip:1001@test.example.com>;tag=1001
To: 2001 <sip:2001@test.example.com>
...

c=IN IP4 172.16.123.9
t=0 0
**m=audio 11000 UDP/TLS/RTP/SAVP 0
a=rtpmap:0 PCMU/8000
a=rtcp-mux
**a=sendrecv
**a=fingerprint: SHA-1
49:27:AF:1D:BA:94:2B:00:E9:33:0A:9E:3A:B5:93:6D:EF:11:E4:55
**a=setup:active
```

2. The SBC sends an INVITE to Bob. The RTP does not contain any security parameters because Bob's realm does not use DTLS over SRTP or SDES. Bob's end station negotiates the connection with the SBC as normal audio RTP over UDP.
3. The SBC handles the reply, up to and including the 200 OK from Bob and to Alice. The SDP that the SBC sends to Alice includes:
   - The “m” attribute, indicating media transport uses DTLS-SRTP.
   - The “a=setup:passive” attribute indicating that the SBC acts as the TLS server, receiving the DTLS connection from Alice.
   - The “a=fingerprint” attribute. This is the SBC certificate hash, used to identify the key that it presents during the ensuing DTLS handshake to authenticate itself to the peer. This handshake uses the media path.

```
SIP/2.0 200 OK
Via: SIP/2.0/udp 192.168.123.2:5060;branch=z9hG4bK-2
From: 1001 <sip:1001@test.example.com>;tag=1001
To: 2001 <sip:2001@test.example.com>;tag=2001
...

c=IN IP4 192.168.123.16
t=0 0
**m=audio 10000 UDP/TLS/RTP/SAVP 0
a=rtpmap:0 PCMU/8000
**a=setup:passive
**a=fingerprint:sha-1
AF:F2:99:EF:BF:9D:24:B4:1F:F4:87:83:47:5A:F1:09:1F:2F:89:A1
```

4. With the SIP signaling setup, Alice begins the DTLS signaling across the media path by sending a DTLS Client Hello to the SBC. The media path is between the IP address and the port exchanged in the offer/answer SDP in the “c=” and “m=” attributes respectively. The Hello includes the “use_srtp” extension which contains the SRTP protection profile preferred by Alice.
5. The SBC responds by sending a DTLS Server Hello to Alice. This Hello includes:
   - The “use_srtp” extension.
   - The DTLS Certificate message that contains the system's certificate. This certificate includes the which public key of the SBC.
6. Alice calculates the hash of the certificate and compares it against the SBC fingerprint sent in the answer SDP. This comparison confirms that the party involved in the signaling path is same as the party involved in the media path, thereby authenticating the SBC.

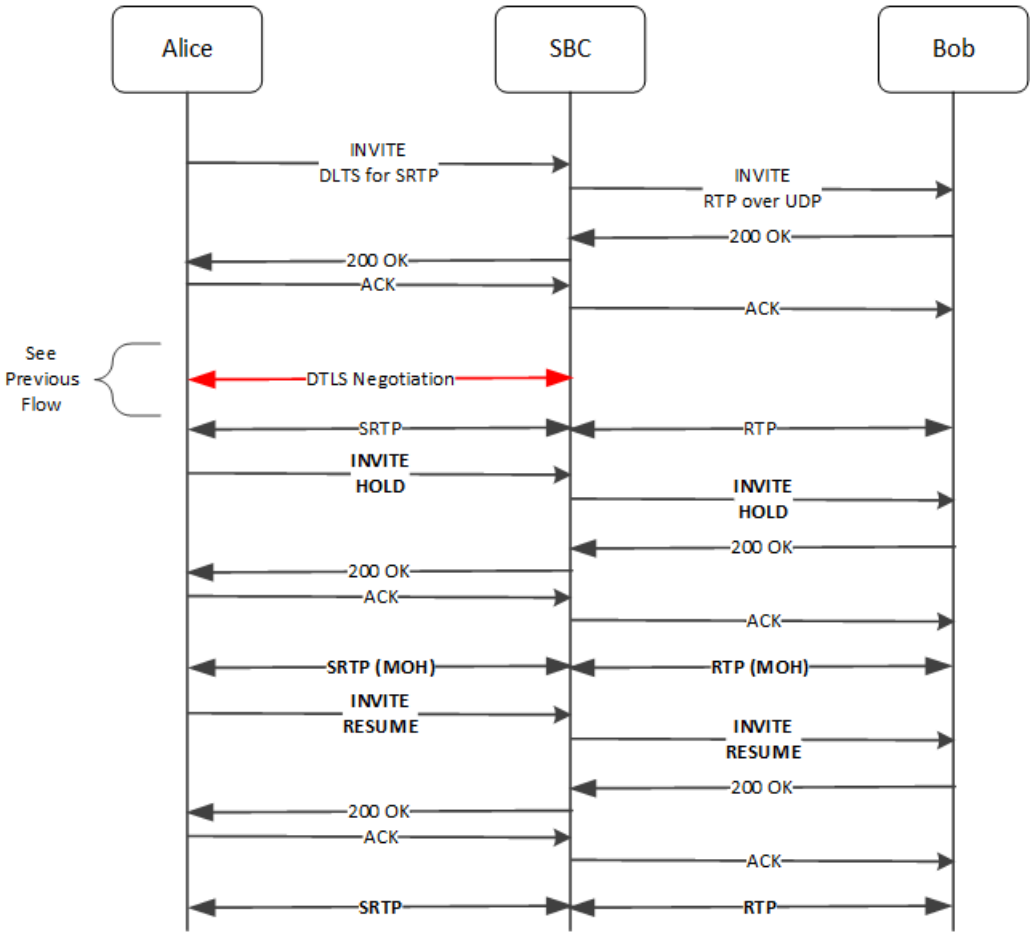
   > **Note:**
   >
   > The cipher suite negotiated in the Hello exchange protects the DTLS handshake message, not the media. Media is protected by the negotiated SRTP protection profile.

   At this point, all aspects of the Server Hello are done.
7. The SBC sends Alice a DTLS Server Key Exchange message, including a Certificate Request, based on the key exchange algorithm negotiated in the DTLS cipher suite.

8. Alice sends the SBC a DTLS Certificate, Client Key Exchange, based on the key exchange algorithm negotiated in the DTLS cipher suite. This message establishes a pre-master secret between the client and server.
9. The SBC calculates the hash of the certificate and compares it against Alice's fingerprint sent in the answer SDP. This confirms that the party involved in the signaling path is same as the party involved in the media path, thereby authenticating Alice.
10. Alice send a DTLS Certificate Verify message to the SBC, verifying the client certificate.
11. Alice send a DTLS Change Cipher Spec, Finished Message message to the SBC, The client sends these messages to signal the transition. Ensuing TLS control messages are protected by the negotiated DTLS cipher suite. Since the use_srtp extension is negotiated, all application data is assumed to be RTP/RTCP packets and are protected by SRTP. Also it serves as an indication that the key exchange and authentication process with the peer entity is successful.
12. The SBC send a DTLS Change Cipher Spec, Finished Message message to Alice. At this point, both the SBC and Alice are in receipt of the SRTP keying material, have calculated the SRTP keys using the TLS master as an input parameter to the SRTP KDF. Both stations start encrypting media using SRTP between the address and ports specified in the SDP negotiation.

Music on Hold

The SBC supports DTLS media flows during hold and music on hold scenarios within the context of SIP signaling. In the example below, the callee has already established an SRTP flow across the media plane, in this case using the SBC as the UAS. The diagram condenses the DTLS negotiation, covered above, using a single line.



Alice initiates the music on hold scenario using the SDP sendonly parameter. The SBC maintains DTLS-SRTP flows until Alice issues the sendrecv parameter, upon which the SRTP and RTP call media resumes.

1. Alice sends a new INVITE to the SBC on the signaling plane using the a=sendonly parameter to signal the hold, and the fingerprint to validate it as the same end station that setup the SRTP flow..

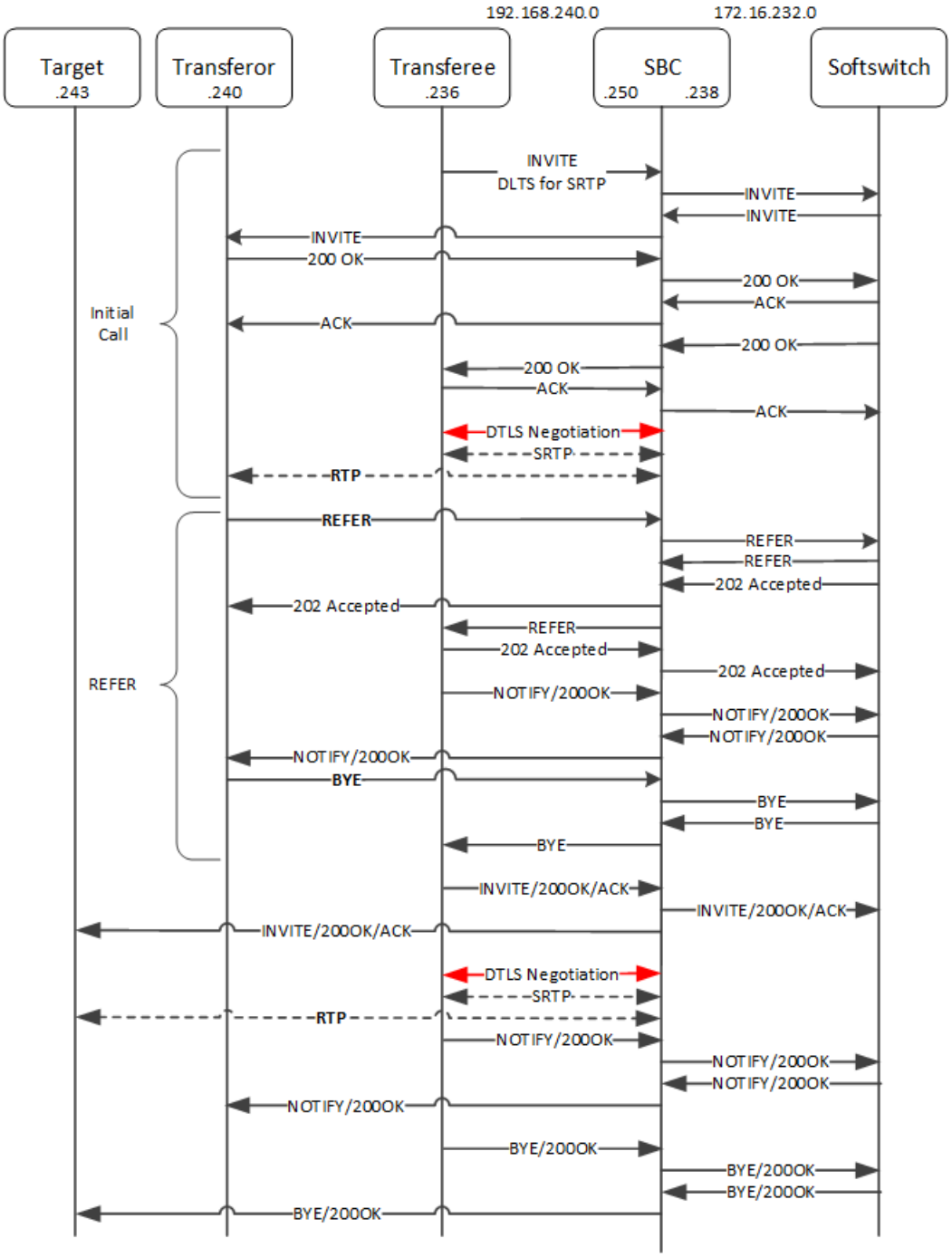
```
SIP/2.0 200 OK
Via: SIP/2.0/udp 192.168.123.2:5060;branch=z9hG4bK-2
From: 1001 <sip:1001@test.example.com>;tag=1001
To: 2001 <sip:2001@test.example.com>;tag=2001
...

c=IN IP4 192.168.123.16
t=0 0
m=audio 10000 UDP/TLS/RTP/SAVP 0
a=rtpmap:0 PCMU/8000
**a=sendonly
**a=fingerprint:sha-1
AF:F2:99:EF:BF:9D:24:B4:1F:F4:87:83:47:5A:F1:09:1F:2F:89:A1
```

2. The SBC subsequently sends an INVITE to Bob that also includes a=sendonly.
3. Bob replies to the SBC with a 200 OK that includes a=recvonly.
4. The SBC then replies to Alice the with a 200 OK that includes a=recvonly and its fingerprint.

5. At this point, Alice restarts the SRTP flow with music as the media.
6. To remove the hold, Alice sends another INVITE that includes a=sendrecv and its fingerprint.
7. The SBC follows suit with Bob.
8. Upon acknowledgment, Alice resumes the SRTP flow with the SBC, which resumes the RTP flow with Bob.

Blind Call Transfer using Unattended REFER

The SBC supports the referral of DTLS media flows that use SIP REFER. In the example below, the caller reaches the softswitch. which directs the call to the callee. In this case, the SBC manages the call, supporting SRTP/RTP interworking. When the callee issues a REFER, it becomes the transferor. Both the SBC and the softswitch support the REFER, ending the original call and performing a new SIP INVITE sequence. The SBC also performs a new DTLS-SRTP negotiation, which results in a new SRTP/RTP interworking call. The diagram condenses the DTLS negotiation using a single line.



This scenario uses SIP REFER to establish a new SRTP flow to the refer target.

1. The scenario begins with a caller using the SBC and softswitch to establish SRTP from the caller to the SBC and RTP from the SBC to the callee. This setup uses DTLS-SRTP as previously described, with the SDP m=audio and a=fingerprint triggering the DTLS negotiation.

```
INVITE sip:1002@192.168.240.250:5061 SIP/2.0
Via: SIP/2.0/tls 192.168.240.236:5060;branch=z9hG4bK-4
From: 1001 <sip:1001@ccenter.example.com:5060>;tag=1001
To: 1002 <sip:1002@ccenter.example.com:5061>
...
```

```
m=audio 11000 UDP/TLS/RTP/SAVP 0
a=rtpmap:0 PCMU/8000
a=sendrecv
a=fingerprint: SHA-1
49:27:AF:1D:BA:94:2B:00:E9:33:0A:9E:3A:B5:93:6D:EF:11:E4:55
```

The caller and the SBC establish DTLS-SRTP media; the softswitch releases the media and the SBC establishes RTP media with the callee.

2. The callee decides to transfer the call, issuing a SIP REFER. The caller becomes the transferor and the callee becomes the transferee.

```
REFER sip:1001@192.168.240.250:5061;transport=tls SIP/2.0
Via: SIP/2.0/tls 192.168.240.240:5060;branch=z9hG4bK-8
To: 1001 <sip:1001@ccenter.example.com>;tag=1001
From: 1002 <sip:1002@ccenter.example.com>;tag=1002
Call-id: 1-192.168.240.236
CSeq: 1 REFER
Refer-To: sip:1003@192.168.240.243:5060
Referred-By: <sip:1002@192.168.240.240:5060>
```

3. After the SBC and the softswitch have accepted the REFER, the transferee gets the REFER and issues its own 202 Accepted message. Subsequently, the transferee issues a NOTIFY to the SBC that includes a subscription to the REFER.

```
NOTIFY sip:1002@192.168.240.250:5061;transport=tls SIP/2.0
Via: SIP/2.0/tls 192.168.240.236:5060;branch=z9hG4bK-10
From: 1001 <sip:1001@ccenter.example.com:5060>;tag=1001
To: 1002 <sip:1002@ccenter.example.com:5061>;tag=1002
Call-id: 1-192.168.240.236
CSeq: 2 NOTIFY
Event: refer
Max-Forwards: 70
Subscription-State: active;expires=120
```

4. The SBC provides this NOTIFY to the softswitch and the transferor, upon which the transferor issues a BYE to the SBC. The BYE makes its way to the transferee through the SBC and the softswitch.
5. Upon receiving the BYE, the transferee issues an new INVITE sequence that traverses the SBC, the softswitch and, reaches the REFER target. These INVITEs include one to setup DTLS-SRTP between the transferee and the SBC, and RTP from the SBC to the target.
6. The transferee initiates another NOTIFY sequence that terminates at the transferor, and notifies that the REFER subscription now includes the target. This verifies to the transferor that the REFER was successful.
7. The call flow ultimately shows the transferee tearing down the call with BYE/200 OK sequences.

DTLS-SRTP Configuration

DTLS-SRTP configuration on the SBC consists of the following steps.

1. Identify the valid ciphers that use can use on the endpoint(s). You must set the cipher-list on tls-profile to match the endpoint(s), or set the list to ALL.

2. Create one or more TLS Security Policies that specify key exchange protocols and protocol-specific profiles.
3. Create one or more DTLS-SRTP profiles, which specify parameter values negotiated during the offer/answer exchange.
4. Assign a TLS Security Policy to your DTLS-SRTP profile.
5. Assign your DTLS-SRTP profile to a realm.

Configure DTLS-SRTP

A DTLS-SRTP profile specifies the parameter values offered or accepted during DTLS negotiation.

To configure DTLS-SRTP profile parameters:

Create one or more TLS Security Policies that specify key exchange protocols and protocol-specific profiles.

1. From superuser mode, use the following command sequence to access dtls-srtp-profile configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# dtls-srtp-profile
ORACLE(dtls-srtp-profile)#
```

2. Use the required **name** parameter to provide a unique identifier for this **dtls-srtp-profile** instance.

   name enables the creation of multiple **dtls-srtp-profile** instances.
3. Use the required **tls-profile** parameter to assign the TLS profile you created for this **dtls-srtp-profile** instance.
4. Use the required **dtls-completion-timeout** parameter to specify a time limit to the DTLS handshake.

   Values range from 0 (Default) to 999999 seconds.
5. Define the system as a server using the **preferred-setup-role** parameter.
   - passive—Proposes that the SBC perform the server role.
6. Use the **crypto-suite** parameter to select the encryption and authentication algorithms accepted or offered by this dtsl-srtp-profile.

   Allowable values are:
   - SRTP_AES128_CM_HMAC_SHA1_80 (default)—Enables support for the AES/128 bit key for encryption and HMAC/SHA1-180-bit digest for authentication.
   - SRTP_AES128_CM_HMAC_SHA1_32—Enables support for the AES/128 bit key for encryption and HMAC/SHA1-132-bit digest for authentication.
   - SRTP_AEAD_256_GCM
7. Use **done**, **exit**, and **verify-config** to complete configuration of this DTLS profile instance.
8. Repeat Steps 1 through 8 to configure additional DTLS-SRTP profiles.

Assign the DTLS-SRTP Profile to a Realm

DTLS-SRTP profiles become active on the SBC when you apply them to a **realm-config**.

To assign a media-security-policy to a realm:

Create one or more dtls-srtp profiles.

1. From superuser mode, use the following command sequence to access realm-config configuration mode. While in this mode, you assign an existing dtls-srtp-policy to an existing realm.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
identifier:
1. access-1
...
...
selection: 1
ORACLE(realm-config)#
```

2. Use the **dtls-srtp-profile** parameter to assign the policy to the target realm.
3. Use **done**, **exit**, and **verify-config** to complete assignment of the media-security-policy to the realm.

Reporting on DTLS-SRTP Statistics

The use of DTLS-SRTP generates specific traffic detail that you can review from the SBC ACLI.

The **show security** command produces a wide array of security-related statistics, including DTLS-SRTP status and issues. The most applicable information associated with DTLS-SRTP operation presented by **show security** includes the following:

- **show security dtls-srtp < all | realm_id >**—Displays system-wide SRTP session counts. The **show security** command is covered in the *Viewing ETC NIU Statistics* section of the *Maintenance and Troubleshooting Guide*.

The **show security dtls-srtp** can provide system-wide or realm-specific statistics on DTLS-SRTP traffic, as shown below. When you run the command without any arguments, the system displays DTLS-SRTP status on each realm. The output below indicates that there are not any active sessions on either realm. When you add the **all** argument, the system displays traffic statistics for each realm. You can narrow this output by specifying realm_ID.

```
ORACLE# show security dtls-srtp
17:40:07 - 133
core        Idle
peer        Idle
ORACLE# show security dtls-srtp all
17:40:10 - 136 Realm = core
                                -----Lifetime----
                     Recent      Total      PerMax
Packets Recv              0          3           2
Handshake Complete        0          1           1
```

```
Handshake Error           0          0          0
Fingerprint Error         0          0          0
Timeout                   0          0          0
UnsupportedSrtpProfile    0          0          0
KeyMaterialExportError    0          0          0
KeyApplyOnFlowError       0          0          0
InternalError             0          0          0
17:40:10 - 136 Realm = core

                                  -----Lifetime----
                      Recent      Total      PerMax
Packets Recv               0        122          22
Handshake Complete         0          5           5
Handshake Error            0          0           0
Fingerprint Error          0          0           0
Timeout                    0          0           0
UnsupportedSrtpProfile     0          0           0
KeyMaterialExportError     0          0           0
KeyApplyOnFlowError        0          0           0
InternalError              0          0           0
```

In addition, the **show datapath** command can display statistics related to DTLS-SRTP. This command, however, is extremely complex and can produce sensitive and high overhead output. Oracle recommends you contact technical support for assistance with the **show datapath** command.

Secure and Non-Secure Flows in the Same Realm

To simplify deployments, the Oracle Communications Session Border Controller allows secure and non-secure flows in the same realm.This broadened set of capabilities means the Oracle Communications Session Border Controller can support RTP and SRTP flows, and it can support a larger group of UAs that might have varying SRTP abilities. Prior to this release, when a cryptographic session arrived at the Oracle Communications Session Border Controller and failed to match an applicable media security profile, it was rejected.

This broadened support for secure and non-secure flows and for UAs with various SRTP abilities is established throughout the system, residing in these configurations:

- media-sec-policy
- sdes-profile

While configurations reside there, you should also note special considerations for the **security-policy** configuration and implications for security associations.

Mode Settings in the Media Security Policy

The media security policy configuration's **mode** parameter offers three settings. It is the **any** mode that allows you to support secure and non-secure flows in the same realm.

For Incoming Flows

This section describes the way all three settings behavior for incoming flows.

- **rtp**—If the incoming media security policy associated with a realm has **rtp** set as its mode, then the Oracle Communications Session Border Controller only accepts offer SDP

containing RTP/AVP media lines. Otherwise, the Oracle Communications Session Border Controller rejects the session with a 488 Not Acceptable Here.

- **srtp**—If the incoming media security policy associated with a realm has s**rtp** set as its mode, the Oracle Communications Session Border Controller only accepts offer SDP containing RTP/SAVP media lines. Otherwise, the Oracle Communications Session Border Controller rejects the session with a 488 Not Acceptable Here.
- **any**—If the incoming media security policy associated with a realm has **any** set as its mode, the Oracle Communications Session Border Controller accepts offer SDP that has RTP/AVP media lines, RTP/SAVP media lines, or both.

For Outgoing Flows

This section describes the way all three settings behavior for outgoing flows.

- **rtp**—If the outgoing media security policy associated with a realm has **rtp** set as its mode, then the Oracle Communications Session Border Controller converts any RTP/SAVP media lines from incoming offer SDP to RTP/AVP for the offer SDP it sends out. Incoming offer SDP might look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 192.168.22.180
s=SIP Call
c=IN IP4 192.168.22.180
t=0 0
m=audio 5010 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
a=crypto:0 AES_CM_128_HMAC_SHA1_80
inline:f0oLKTuMYwXqrKa7Ch+MOBvLe8YnXnD6Kmnj4LQ2
```

The Oracle Communications Session Border Controller will take that and convert it to the following for outgoing traffic.

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 6000 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-1
```

This conversion can result in multiple media lines with RTP/AVP for the same media profile and an RTP/SAVP media line for the same media profile. To prevent duplicate lines in the SDP the Oracle Communications Session Border Controller sends, the Oracle Communications Session Border Controller inspects incoming SDP to determine is RTP/AVP and RTP/SAVP media lines exist for the same media profile. If it finds such a media profile, the Oracle Communications Session Border Controller disables the RTP/AVP (by setting the port to 0 in the outgoing offer SDP) corresponding to the RTP/AVP media line for that media profile. Doing so forces the UA answering the SDP offer to choose the media lines corresponding to the RTP/SAVP media lines in the incoming offer SDP. An SRTP-RTP session results.

The incoming offer SDP might look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 192.168.22.180
s=SIP Call
c=IN IP4 192.168.22.180
t=0 0
m=audio 5012 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 5010 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
a=crypto:0 AES_CM_128_HMAC_SHA1_80
inline:f0oLKTuMYwXqrKa7Ch+MOBvLe8YnXnD6Kmnj4LQ2
```

And the outgoing offer SDP will look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 0 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 6002 RTP/AVP 0 8 18 0 101
```

```
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
```

If an incoming offer has both RTP/AVP and RTP/SAVP media lines, and there exists duplicate media descriptions for the same codec, then Oracle Communications Session Border Controller removes the duplicate description.

- **srtp**—If the outgoing media security policy associated with a realm has s**rtp** set as its mode, the Oracle Communications Session Border Controller converts any RTP/AVP media lines from an incoming offer SDP to RTP/SAVP for the offer SDP the Oracle Communications Session Border Controller sends.
  The incoming offer SDP might look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 192.168.22.180
s=SIP Call
c=IN IP4 192.168.22.180
t=0 0
m=audio 5012 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
```

  And the outgoing offer SDP will look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 6000 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-1
a=crypto:0 AES_CM_128_HMAC_SHA1_80
inline:f0oLKTuMYwXqrKa7Ch+MOBvLe8YnXnD6Kmnj4LQ2
```

  This conversion might result in multiple media lines with RTP/SAVP for the same media profile if the incoming offer SDP has an RTP/AVP media line and an RTP/SAVP media for

the same media profile. To prevent multiple identical media lines in the SDP it sends, the Oracle Communications Session Border Controller inspects the incoming SDP to determine whether both RTP/AVP and RTP/SAVP media lines exist for the same media profile. If it finds such a media profile, the Oracle Communications Session Border Controller disables the RTP/SAVP (by setting the port to 0 in the outgoing offer SDP) corresponding to the RTP/AVP media line for that media profile. Doing so forces the UA answering the SDP offer to choose the media lines corresponding to the RTP/SAVP media lines in the incoming offer SDP. An SRTP-SRTP session results.

The incoming offer SDP might look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 192.168.22.180
s=SIP Call
c=IN IP4 192.168.22.180
t=0 0
m=audio 5012 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 5010 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
a=crypto:0 AES_CM_128_HMAC_SHA1_80
inline:f0oLKTuMYwXqrKa7Ch+MOBvLe8YnXnD6Kmnj4LQ2
```

And the outgoing offer SDP will look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 0 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 6002 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

```
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-1
a=crypto:0 AES_CM_128_HMAC_SHA1_80
inline:f0oLKTuMYwXqrKa7Ch+MOBvLe8YnXnD6Kmnj4LQ2
```

If an incoming offer has both RTP/AVP and RTP/SAVP media lines, and there exists duplicate media descriptions for the same codec, then Oracle Communications Session Border Controller removes the duplicate description.

- **any**—If the outgoing media security policy associated with a realm has **any** set as its mode, the Oracle Communications Session Border Controller creates offer SDP based on the value configured in the **egress-offer-format**, which can be set in the **sdes-profile** configuration.
  - If the value is **same-as-ingress**, the Oracle Communications Session Border Controller leaves the profile of the media lines unchanged.
  - If the value is **simultaneous-best-effort**, the Oracle Communications Session Border Controller inspects the incoming offer SDP and:
    * Adds an RTP/SAVP media line for any media profile that has only the RTP/AVP media profile
    * Adds an RTP/AVP media line for any media profile that has only the RTP/SAVP media profile
      Should the media profile in the incoming offer SDP already have two media lines (one fore RTP/AVP and on for RTP/SAVP), the Oracle Communications Session Border Controller does not have to make these additions. It will map the media lines in the answer it receives with the media lines from the incoming offer SDP. It will also ensure that media lines in the answer SDP it sends match the media lines from the incoming offer SDP.

Security Associations for RTP and RTCP

With RTP and SRTP supported in the same realm, you want to configure your SRTP security policies to preserve system resources. You need to do to avoid session agent interaction that can have an adverse impact on the number of sessions.

To do so, check the **local-ip-match-address** for the STRP security policy has an IP address different from the all steering pool IP addresses for realms requiring both RTP and SRTP. The Oracle Communications Session Border Controller recognizes this difference automatically and sets the connection address of media lines in SDP accordingly:

- The connection address for RTP media lines is the IP address of the applicable steering pool. The Oracle Communications Session Border Controller passes through RTP and RTCP packets sent by and received from the steering pool IP address. This operation requires no reference to session agents because the steering pool address does not match the IP address for the SRTP security policy's **local-ip-address-match** value.
- The connection address of the SRTP media lines continues to be the **local-ip-address-match** value from the applicable SRTP security policy.

Since RTP and RTCP packets are sent to and from the steering pool's IP address (an IP address for which there is no SRTP security policy configured), there is no reason to reference session agents.

Security Configuration for RTP and RTCP

This section shows you how to configure your Oracle Communications Session Border Controller to support secure and non-secure flows in the same realm.

To configure a security policy to support secure and non-secure flows in the same realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **security** and press Enter.

```
ORACLE(configure)# security
ORACLE(security)#
```

3. Type **media-security** and press Enter.

```
ORACLE(security)# media-security
ORACLE(media-security)#
```

4. Type **media-sec-policy** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

```
ORACLE(media-security)# media-sec-policy
ORACLE(media-sec-policy)#
```

5. Type **inbound** to enter the setting for inbound flows.

```
ORACLE(media-sec-policy)# inbound
ORACLE(inbound)#
```

6. **mode**—Enter the mode that you want to use for inbound flows. You can choose from **rtp**, **srtp**, and **any**.
7. **protocol**—Change this value from **sdes** to **none**. Use the **done** command to save your work, and exit the **inbound** configuration.
8. Type **outbound** to enter the setting for inbound flows.

```
ORACLE(media-sec-policy)# outbound
ORACLE(outbound)#
```

9. **mode**—Enter the mode that you want to use for outbound flows. You can choose from **rtp**, **srtp**, and **any**.
10. **protocol**—Change this value from **sdes** to **none**. Use the **done** command to save your work, and exit the **outbound** configuration.
11. Type **done** and continue.

    To set the egress offer format for an SDES profile configuration:

12. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

13. Type **security** and press Enter.

```
ORACLE(configure)# security
ORACLE(security)#
```

14. Type **media-security** and press Enter.

```
ORACLE(security)# media-security
ORACLE(media-security)#
```

15. Type **sdes-profile** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

```
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

16. **egress-offer-format—Choose an egress offer format for this profile to use when you set the outbound mode in the media security policy to any. You can select one of two values:**
    - If the value is **same-as-ingress (default)**, the Oracle Communications Session Border Controller leaves the profile of the media lines unchanged.
    - If the value is **simultaneous-best-effort**, the Oracle Communications Session Border Controller inspects the incoming offer SDP and:
      – Adds an RTP/SAVP media line for any media profile that has only the RTP/AVP media profile
      – Adds an RTP/AVP media line for any media profile that has only the RTP/SAVP media profile
17. Type **done** to save your work and continue.

Supporting UAs with Different SRTP Capabilities

To support UAs with different levels of SRTP capabilities, the **use-ingress-session-params** parameter appears in the **sdes-profile** configuration. The values for this parameter allow the Oracle Communications Session Border Controller to accept and (where applicable) mirror the UA's proposed cryptographic session parameters:

- **srtp-auth**—Decides whether or not authentication is performed in SRTP
- **srtp-encrypt**—Decides whether or not encryption is performed in SRTP
- **srtcp-encrypt**—Decides whether or not encryption is performed in SRTCP

Using these possible values, the Oracle Communications Session Border Controller accepts the corresponding incoming session parameters.

Receiving Offer SDP

When the Oracle Communications Session Border Controller receives offer SDP with applicable session parameters, it uses the same session parameters in its answer SDP (if it

can support the same). This is true even if the value for that session parameter differs from the available media security profile.

Consider this example: An SDES profile is applied for incoming direction for a media security policy configured with the **srtcp-encrypt** value set to **enabled**. With the **use-ingress-session-params** parameter set to **srtcp-encrypt** for the SDES profile, the Oracle Communications Session Border Controller accepts the offer SDP and also sets UNENCRYPTED_SRTCP for the cryptographic attributes in its answer SDP. When the call connects, the Oracle Communications Session Border Controller does not encrypt or decrypt SRTCP packets. Without the SDES profile set this way, the Oracle Communications Session Border Controller would reject offer SDP if any of its cryptographic attributes showed UNENCRYPTED_SRTCP in its session parameters list.

Receiving Answer SDP

When the Oracle Communications Session Border Controller receives answer SDP with the accepted session parameter, the value for the same session parameters that the Oracle Communications Session Border Controller uses might or might not be the same as the incoming value. Configuration of the outbound media security profile controls the value used because the Oracle Communications Session Border Controller makes offer SDP, which cannot be changed, with the session parameters based on the outgoing media security profile.

Consider this example: An SDES profile is applied for incoming direction for a media security policy configured with the **srtcp-encrypt** value set to **enabled**, so the cryptographic attributes in the SDP the system sends **do not have the UNENCRYPTED_SRTCP** session parameters. If the UNENCRYPTED_SRTCP appears in the corresponding answer SDP it receives, the Oracle Communications Session Border Controller accepts it if the **srtcp-encrypt** value appears in the **use-ingress-session-params** parameter. But the Oracle Communications Session Border Controller still performs SRTCP encryption. When the call connects, the Oracle Communications Session Border Controller encrypts outgoing SRTCP packets but does not decrypt incoming SRTCP packets. So if the UA (receiving the system's offer SDP) does not support SRTCP decryption, it will likely reject the offer SDP.

SDES Profile Configuration

To set the ingress session parameters for an SDES profile configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **security** and press Enter.

```
ORACLE(configure)# security
ORACLE(security)#
```

3. Type **media-security** and press Enter.

```
ORACLE(security)# media-security
ORACLE(media-security)#
```

4. Type **sdes-profile** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

```
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

5. **use-ingress-session-params—Enter the list of** values for which the Oracle Communications Session Border Controller will accept and (where applicable) mirror the UA's proposed cryptographic session parameters:
   - **srtp-auth**—Decides whether or not authentication is performed in SRTP
   - **srtp-encrypt**—Decides whether or not encryption is performed in SRTP
   - **srtcp-encrypt**—Decides whether or not encryption is performed in SRTCP

```
ACMEPACKET(sdes-profile)# use-ingress-session-params srtp-auth srtp-
encrypt srtcp-encrypt
```

6. Type **done** to save your work and continue.

Refining Interoperability

To refine any remaining interoperability issues, you can use the options parameter in the **media-sec-policy** and **sdes-profile** configurations.

Common values to configure an option are **include-local-id** and **include-remote-id**.

- **include-local-id**—The Oracle Communications Session Border Controller includes the IDi in the I_MESSAGE and the IDr in the R_MESSAGE.
  When used for the outbound direction of a media security policy, the IDi is included in the I_MESSAGE the Oracle Communications Session Border Controller sends. The content of the IDi is the value of the Contact header found in the INVITE message.
- **include-remote-id**—The system includes the IDr in the I_MESSAGE.

HMU Support for RTP to SRTP Interworking

The Oracle Communications Session Border Controller (SBC) supports Real-Time Transport Protocol (RTP) to Secure Real-Time Transport Protocol (SRTP) interworking Function by monitoring and correcting unexpected changes to session continuity information. You can enable the **hide-egress-media-update** parameter on the applicable realm.

RFC 3350 does not require RTP to maintain sequential packet sequence numbering. In contrast, STRP does not allow significant packet sequence number changes or resets to zero. To compensate for this, the SBC can detect such changes and calculate and transmit the correct values to the SRTP end station when needed.

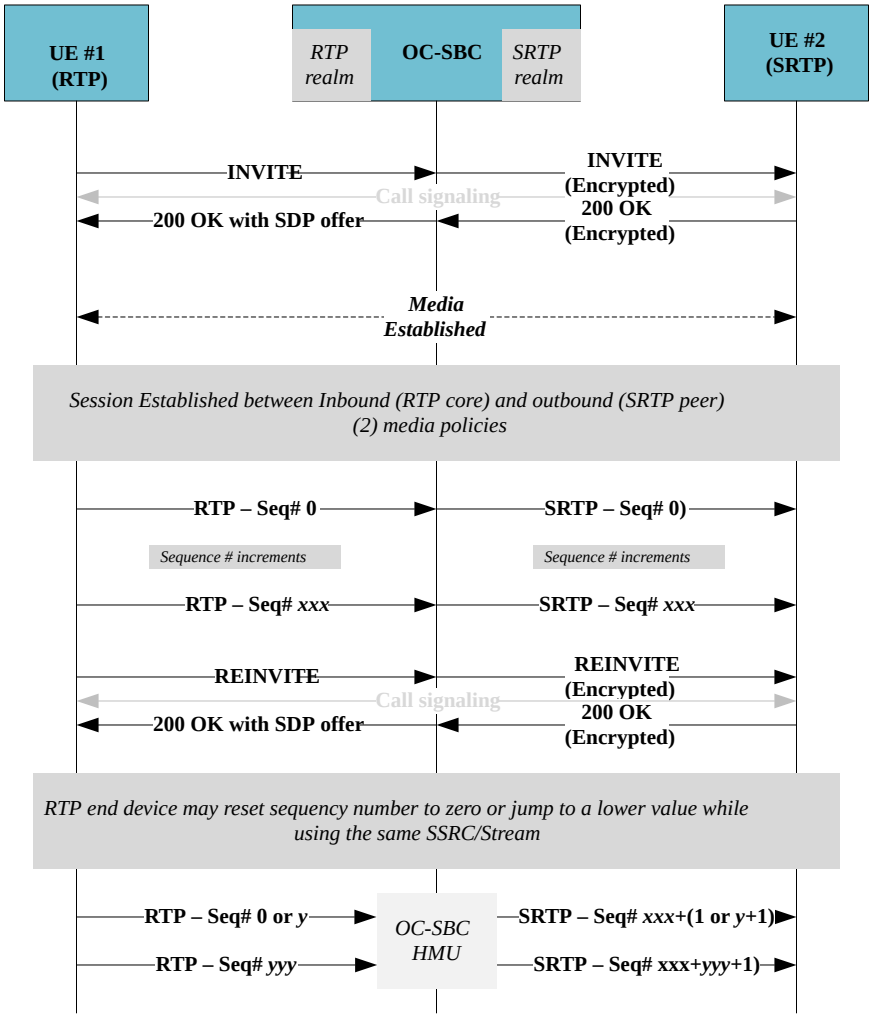
When configured to support RTP to SRTP interworking, the HMU function latches previous Synchronization Source (SSRC) sequence numbers and timestamps from RTP packets and watches for changes to ensuing sequence numbers on an ongoing basis. Sequence number changes the HMU function acts on include resets to zero and jumps downward. The HMU logic performs calculations on the latched sequence number, and populates the egress packet with a new sequence number that the SRTP end station can recognize as valid. When the sequence number in the RTP changes and the HMU is disabled, the SBC cannot support RTP to SRTP interworking and cannot forward the packets.

SRTP considers downward sequence number changes greater than 127 as indicating the packet is a replay packet that should be discarded. The HMU function monitors for sequence number decreases greater than 127 and resets to zero. When the SBC detects such a change, it invokes the HMU logic that sets the prescribed values in the SRTP traffic before egress.

> **Note:**
>
> Configuration on the ingress realm differs from standard HMU configuration, which you configure on the egress realm. Similarly, bi-directional HMU is not relevant for RTP to SRTP interworking.

For example, consider configuring for single-ended SRTP sessions between a core (unencrypted) realm and a peer (encrypted) realm. To do this, you configure the core realm media security policy (inbound and outbound) to RTP mode. In addition, you configure the peer realm media security policy (inbound and outbound) to SRTP mode. After the SBC establishes the session flows through signaling, it applies the media security policy to ingress RTP packets from the inbound realm and transmits them through the outbound realm as SRTP.

The following call flow depicts the SBC using HMU to support RTP to SRTP interworking. The call sets up normally with RTP and SRTP interworking properly. The RE-INVITE from UE #1 triggers the HMU logic, which manages the RTP packet sequence numbers and prevents the SRTP leg from dropping media packets, or eventually, the call.



HMU Support for RTP to SRTP Interworking extends the HMU feature, which operates when you apply the **hide-egress-media-update** parameter to all media traffic on a realm. Using **hide-egress-media-update** allows you to limit HMU processing to the targeted RTP and SRTP interworking traffic.

> **Note:**
>
> SBC does not support HMU for RTCP or SRTCP packets. Regardless of HMU configuration, the SBC supports only up to 7 SSRC changes per SRTP session. Also, if HMU is disabled, the SBC supports only up to 7 SSRC changes per SRTP session for RTP and RTCP packets.

See Hiding Problematic Media Updates for general information on HMU, including the HMU state machine, RTC, and HA support.

Multi-system Selective SRTP Pass-through

Prior to Release S-CX6.3F1, Oracle Communications Session Border Controller provided a single Secure Real-time Transport Protocol (SRTP) operational mode, referred to as SRTP Media Proxy Mode. In this original processing mode, each Oracle Communications Session Border Controller in the SRTP media path served as a proxy for the media — always decrypting inbound traffic, and encrypting outbound traffic.

Release S-CX6.3F1 introduces support for a new, alternative processing mode, referred to as Multi-system Selective SRTP Pass-through Mode. In this new mode encryption and decryption of SRTP media is handled by the SRTP endpoints, the calling and called parties. Off-load of the processor-intensive encryption/decryption provides the Oracle Communications Session Border Controller with the ability to handle a larger number of simultaneous SRTP sessions.

With Multi-system Selective SRTP Pass-through enabled, the Oracle Communications Session Border Controller can configured to selectively allow hair-pinned (spiral) SRTP packets to pass through the Oracle Communications Session Border Controller without encryption or decryption.

> **Note:**
>
> hair-pinned calls are those calls where the calling and called parties are within the same realm and/or within the same sub-network.

Constraints

Multi-system Selective SRTP Pass-through support has the following constraints:

1. The realm, or realms in which the calling and called parties are located are configured for Multi-system Selective SRTP Pass-through support.
2. The call session does not require SIP—INFO to RFC 2833 tone translation.
3. The call session does not require SDES and can be used for the exchange of SRTP keying material. Both the calling and called parties must support the same key exchange protocol.

4. Multi-system Selective SRTP Pass-through support should not be enabled in topologies where core-side application servers may change the c-line to inject a media server or some other media device in the media path. In such cases, SRTP should be terminated at the SD for each endpoint, so that the media server receives unencrypted media. In the other case, where the c-line is not subject to modification, Multi-system Selective SRTP Pass-through can be enabled.

   If any of these conditions are not met, Multi-system Selective SRTP Pass-through processing cannot be provided, and the call is serviced as specified by SRTP Media Proxy Mode.

Operational Overview

To set up Multi-system Selective SRTP Pass-through, the ingress and egress Oracle Communications Session Border Controllers (which can, in fact, be a single Oracle Communications Session Border Controller) exchange the SDES keying material that they receive from their respective endpoint so that the Oracle Communications Session Border Controller peer can pass the material to its adjacent endpoint. The endpoint to endpoint exchange of keying material enables the endpoints themselves to generate encryption/decryption keys.

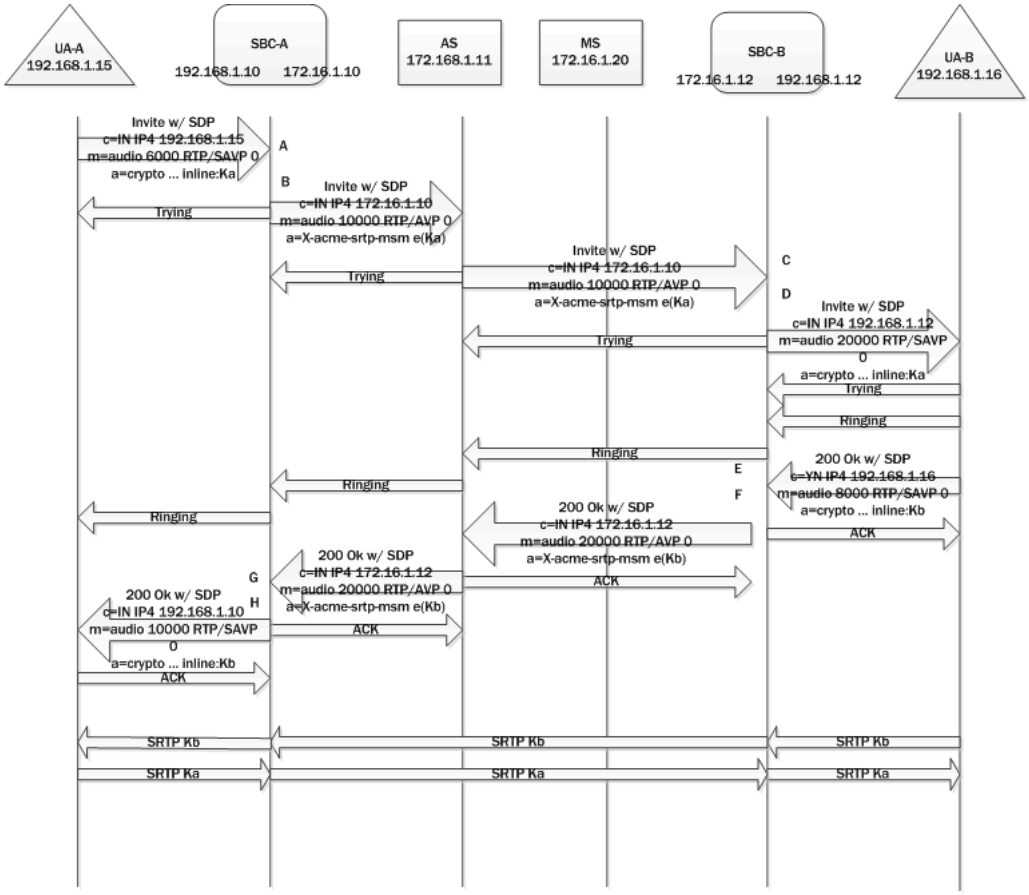
The actual exchange of keying material takes place in SIP messages (specifically, INVITE, 200 OK, and ACK) that carry offer or answer SDPs. Encrypted keying material is conveyed within a media attribute for each SRTP session. The name of the media attribute is configurable.

When either Oracle Communications Session Border Controller receives the encrypted keying material sent by its remote peer, it decrypts the media attribute and passes the plaintext attribute to its endpoint. Consequently, subsequent SRTP packets from the endpoints pass through the Oracle Communications Session Border Controllers without being decrypted and encrypted because both endpoints (now is possession of each others keying material) are able to decrypt the SRTP packets received from the other.

Call Flows

The following three sections describe call signalling for common scenarios.

Call Setup



192.168.1.15

A: The calling party sends an INVITE with SDP to SBC-A. The offer SDP contains an SDES crypto attribute within the SRTP media line.

B: Since Multi-system Selective SRTP Pass-through is enabled within the ingress realm, SD-A adds an a=X-acme-srtp-msm media attribute. The a=X-acme-srtp-msm attribute contains a cookie that includes an encryption of the SDES crypto attribute present in the SDP. The encryption is done using the shared secret configured for encrypting SRTP Pass-through information.

C: SBC-B receives the offer SDP that has the cookie sent by SBC-A. It is assumed that the proxies that forward the offer SDP sent by SBC-A preserve and forward the cookie added by SBC-A.

D: SBC-B checks if the egress realm has Multi-system Selective SRTP Pass-through enabled. If so, SBC-B decrypts the cookie using the shared secret to retrieve the SDES crypto attribute. SBC-B adds the SDES crypto attribute retrieved from the cookie to the offer SDP sent to UA-B.

E: The called party sends an answer SDP with a SDES crypto attribute on the SRTP media line to SBC-B.

F: SBC-B checks if it has received a cookie in the offer SDP and adds the cookie to the answer SDP. The cookie contains an encryption of the SDES crypto attribute received in the answer

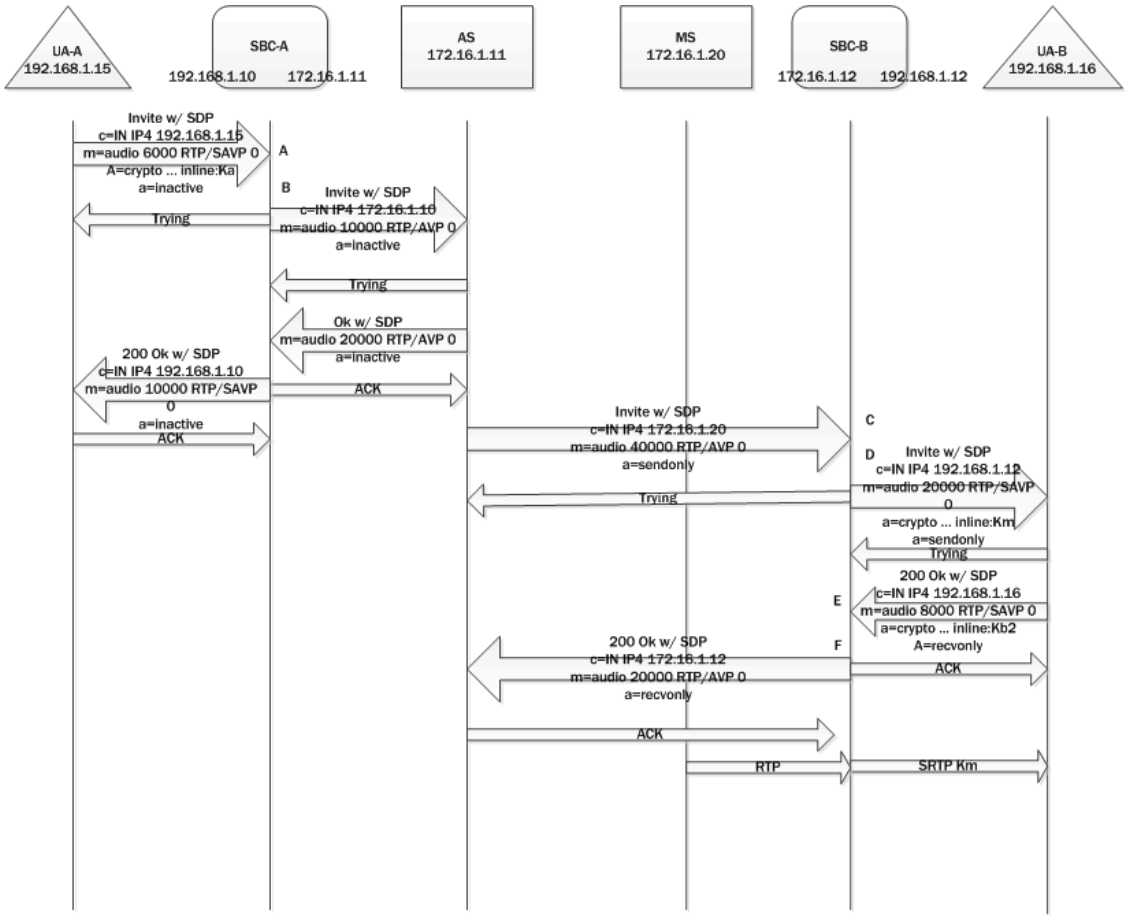
SDP from UA-B. SBC-B does not install any SA or media policy so that SRTP packets from/to UA-B can pass through SBC-B without any decryption/encryption.

G: SBC-A receives the answer SDP that has the cookie sent by SBC-B.

H: SBC-A decrypts the cookie using the shared secret to retrieve the SDES crypto attribute. SBC-A adds the SDES crypto attribute retrieved from the cookie to the answer SDP. SBC-A does not install any SA or media policy so that SRTP packets from/to UA-A can pass through SBC-A without any decryption/encryption.

Music on Hold

After a call is established, one of the endpoints can put the other endpoint on hold. An application server might intercept the re-INVITE from one endpoint (for putting the other on hold) and implement MoH as follows.



A: Endpoint UA-A sends an offer SDP for hold to SBC-A.

B: SBC-A forwards offer SDP without any cookie since there will be no media going from/to UA-A.

C: SBC-B receives an offer SDP that is addressed to the MS without any cookie.

D: Because there is no cookie in the ingress offer SDP, SD-B generates its own SDES crypto attributes for the egress offer SDP sent to UA-B.

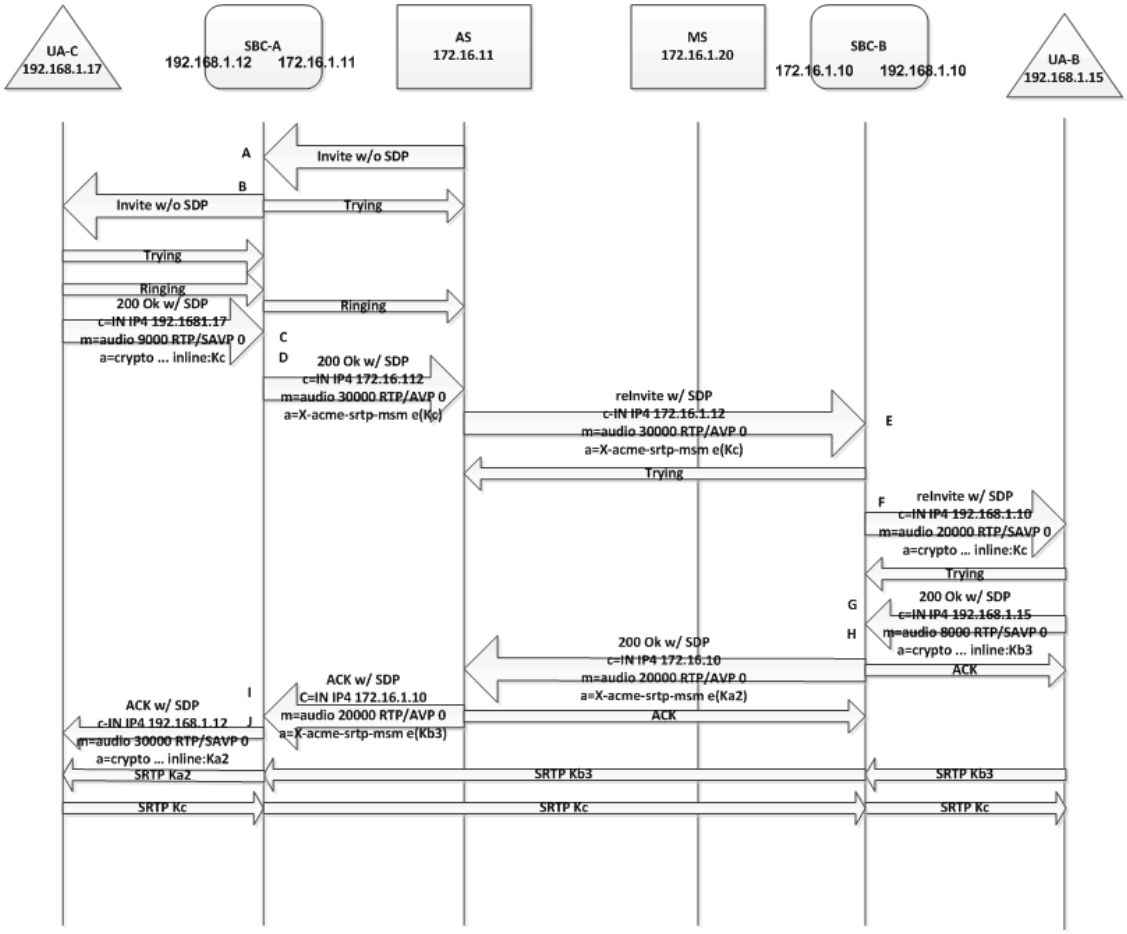
E: SBC-B receives am answer SDP from UA-B.

F: SBC-B sends its answer SDP without any cookie and encrypts SRTP packets going to UA-B. Note that there will be no SRTP packets going from UA-B to SBC-B.

As a result, MoH is heard by UA-B as it decrypts SRTP packets encrypted by SBC-B. When UA-A resumes, the steps to establish media between UA-A and UA-B will be the same as the steps used for call setup.

Once the media is reestablished between UA-A and UA-B media travelling between the two UAs will not be decrypted by either SBC.

Call Transfer



A call transfer can also happen after a call is established. For example, endpoint UA-B can transfer UA-A to another endpoint, UA-C. UA-B can make a blind transfer or an attended transfer. The following diagram describes a blind call transfer with SRTP Pass-through enabled. Note it does not show the SIP message exchange between UA-B and the Application Server to facilitate the call transfer (i.e. the INVITE from UA-B to put the call on hold and the REFER/NOTIFY message exchange between UA-B and the Application Server). After UA-A is put on hold and the transfer target (that is, UA-C) is received from UA-B, the Application Server attempts to establish a call to UA-C. After the call to UA-C is established, the Application Server takes UA-A off hold and connects its media session to that of UA-C.

A: SBC-B receives an INVITE (from the Application Server to invite UA-C) without an offer SDP.

B: SBC-B sends an INVITE without an offer SDP to UA-C.

C: SBC-B receives a 200 OK response with an offer SDP that has a SDES crypto attribute on the SRTP media line from UA-C.

D: Since Multi-system Selective SRTP Pass-through is enabled within the ingress realm, SBC-B adds a cookie to the egress offer SDP that is sent to the core realm.

E: SBC-A receives a re-INVITE to UA-A with the offer SDP that has the cookie sent by SBC-B.

F: Since Multi-system Selective SRTP Pass-through is enabled within the ingress realm, SBC-A adds the SDES crypto attribute retrieved from the cookie to the offer SDP sent to UA-A.

G: SBC-A receives an answer SDP that has a SDES crypto attribute on the SRTP media line from UA-A.

H: SBC-A checks if it has a cookie in the offer SDP and adds the cookie to the answer SDP that is sent to the core realm. The cookie contains an encryption of the SDES crypto attribute received in the answer SDP from UA-B. SBC-A stops the encryption of any SRTP packets going to UA-B (set up for MoH) so that SRTP packets from/to UA-B can now pass through SBC-A without any decryption/encryption.

I: SBC-B receives the answer SDP that has the cookie sent by SBC-A.

H: SBC-B decrypts the cookie using the shared secret to retrieve the SDES crypto attribute. SBC-B adds the SDES crypto attribute retrieved from the cookie to the answer SDP.

After the call to UA-C is established and the call to UA-A is resumed (with the resulted media going between UA-A and UA-C) the Application Server disconnects the call with UA-A. Note that steps C-J are essentially the same steps as steps A-H in the Call Setup example. The difference is that the offer SDP from C comes to SD-B in a 200 OK response and the answer SDP sent by SBC-B to C is in the ACK.

Early Media

Different application servers may implement early media in different ways. In general, the SBC supports early media in the following way.

If the SBC receives an SDP without any cookie in a provisional response, the SBC generates its own SRTP crypto context and exchanges it with the caller via the SDP included in the provisional response. The SBC continues to decrypt and encrypt early media packets going to and from the caller. The SBC stops decrypting and encrypting only if it receives a final response with an answer SDP that signals that SRTP Pass-through should be enabled.

Multi-system Selective SRTP Pass-through with Media Release

When SRTP Pass-through is allowed, the hair-pinned media can also be released to the network if media release is configured — that is, if realm-config parameters **msm-release** is **enabled**, **mm-in-realm** is **disabled**, or **mm-in-network** is **disabled**. If SRTP Pass-through is not allowed, media release will not be allowed either.

Multi-system Selective SRTP Pass-through Configuration

Use the following procedure to enable Multi-system Selective SRTP Pass-through within a specific realm.

1. Use the following command sequence to move to **realm-config** Configuration Mode.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Use the **srtp-msm-passthrough** parameter to enable Multi-system Selective SRTP Pass-through within a specific realm.

   By default, pass-through support is disabled.

```
ORACLE(realm-config)# srtp-msm-passthrough enabled
ORACLE(realm-config)#
```

3. Use **done**, **exit**, and **verify-config** to complete enabling Multi-system Selective SRTP Pass-through within the current realm.

   verify-config checks that the **srtp-msm-password** parameter has been configured, and outputs an error if it has not been configured. **verify-config** also checks other configuration settings that conflict with Multi-system SRTP Pass-through operation. Among these possible mis-configurations are the following.

   rfc2833-mode set to preferred on a SIP interface within a realm that has srtp-msm-passthrough enabled

   rfc2833-mode set to preferred and app-protocol set to SIP on a session-agent within a realm that has srtp-msm-passthrough enabled.

4. If required, repeat Steps 1 through 3 to enable Multi-system Selective SRTP Pass-through on additional realms.

   Use the following procedure to specify values needed to support the exchange of SDES keying information.

5. Use the following command sequence to move to **security** Configuration Mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)#
```

6. Use the **srtp-msm-attr-name** parameter to specify the name of the media attribute used to convey SDES keying information within a SDP media description.

   A valid attribute name must consist of characters from the US-ASCII subset of ISO-10646/UTF-8 as specified in RFC 2327, *SDP: Session Description Protocol*. IANA-registered names should not be used. Values should begin with an X-1 prefix to prevent collision with registered values.

   In the absence of a specified attribute name, the SD provides a default value of X-acme-srtp-msm.

```
ORACLE(security)# srtp-msm-attr-name X-key-material
ORACLE(security)#
```

7. Use the **srtp-msm-password** parameter to provide the shared secret used to derive the key for encrypting SDES keying material that is placed in the media attribute of an SDP media description. Ingress keying material is encrypted using this shared secret before being forwarded to the network core. On egress, the encrypted keying material is decrypted with this same key.

Allowable values are characters strings that contain a minimum of 8 and a maximum of 16 characters.

```
ORACLE(security)# srtp-msm-password IsHeEleemosynary
ORACLE(security)#
```

8. Use **done**, **exit**, and **verify-config** to complete necessary configuration.

   verify-config checks that the **srtp-msm-password** parameter has been configured, and outputs an error if it has not been configured. **verify-config** also checks other configuration settings that conflict with Multi-system SRTP Pass-through operation. Among these possible mis-configurations are the following.

   rfc2833-mode set to preferred on a SIP interface within a realm that has srtp-msm-passthrough enabled

   rfc2833-mode set to preferred and app-protocol set to SIP on a session-agent within a realm that has srtp-msm-passthrough enabled.

Statistics

The number of media sessions set up with Multi-systems Selective SRTP Pass-through is tracked and included in the output of the **show mbcd statistics** command.

SRTP Re-keying

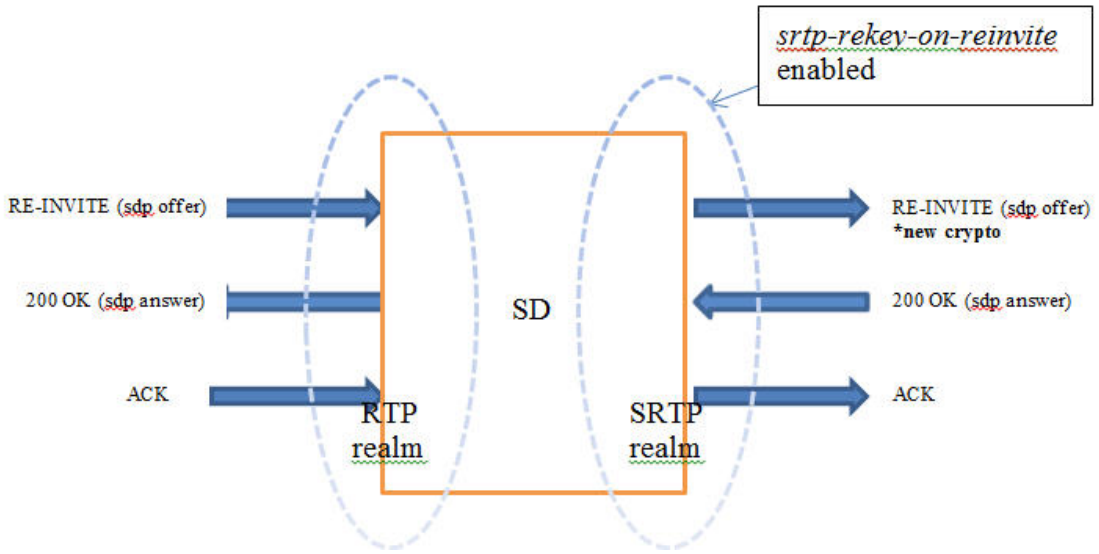
Initialization of SRTP re-keying is supported by the Oracle Communications Session Border Controller.

The Oracle Communications Session Border Controller can generate a new outbound crypto attribute in the SDP offer in a SIP re-INVITE when the **srtp-rekey-on-reinvite** parameter is set to **enabled**. The system generates the attribute regardless of the state of the flow, active or not.

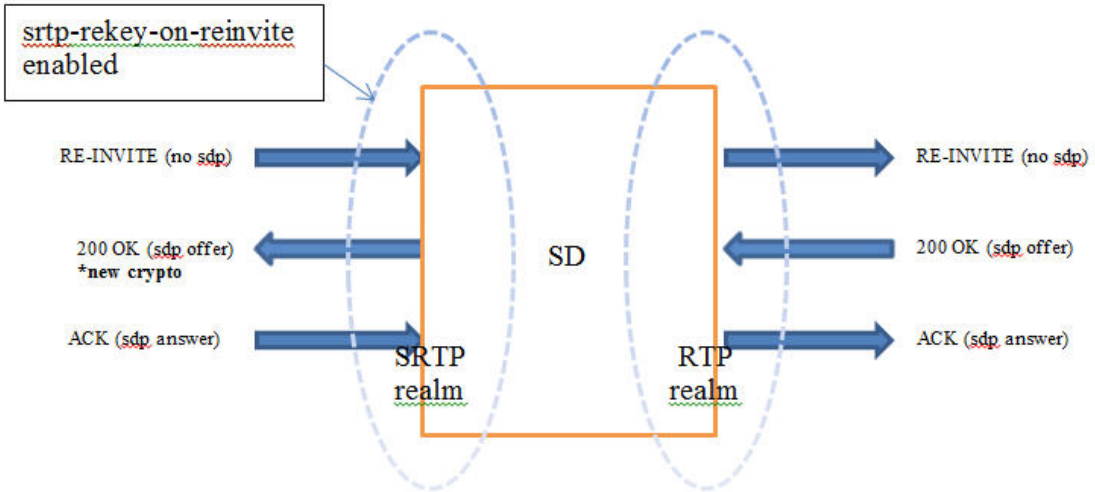
This capability is important for some clients that reside on the SRTP side in a single SRTP termination mode configuration. Any media changes that happen in the RTP side are hidden by the Oracle Communications Session Border Controller. This concealment may cause issues in some configurations, where media servers are involved. When the media changes from media server to called phone, the SRTP endpoint is not aware the media source changed because the SDP offer from the Oracle Communications Session Border Controller is the same as original invite. The result is that some devices drop packets because of Synchronization Source Identifier (SSRC) values mismatch, unexpected jumps in sequence number, sequence number reversions back to 1 triggering replay attack defense, and so forth. In certain environment is has been found that re-keying on every re-invite eliminates all these issues especially in customer setups that use Microsoft Lync products.

The processing of standard RE-INVITES (those containing an SDP offer) and offerless RE-INVITES is shown below.

With SDP:



No SDP:



If the re-invite message is a refresh and **srtp-rekey-on-reinvite** is enabled, the outbound crypto will change but the SDP version will not be incremented on the outgoing invite. If this scenario causes incompatibility issues with customer equipment then add the unique-sdp-id option to **media-manager**, **option** configuration so the Oracle Communications Session Border Controller increments the SDP version in the outgoing invite.

SRTP Re-keying Configuration

Configure **srtp-rekey-on-reinvite** to enable the negotiation and generation of new SRTP keys upon the receipt of a SIP RE-INVITE message that contains SDP.

Confirm that an **sdes-profile** exists.

In the following procedure, change the default state to enabled.

1. Access the **sdes-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
```

```
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

2. Type **select** to choose and configure an existing object.

```
ORACLE(sdes-profile)# select
<name>:
1:  name=sdesprofile01

selection: 1
ORACLE(sdes-profile)#
```

3. **srtp-rekey-on-reinvite**—Set this parameter to **enabled** for re-keying upon the receipt of an SIP reINVITE that contains SDP.
4. Type **done** to save your configuration.

IPSec Support

The Oracle Communications Session Border Controller offers IPSec for securing signaling, media, and management traffic at the network layer.

Supported Protocols

The SBC's IPSec implementation supports all required tools for securing Internet communication via the IPSec protocol suite. The following paragraphs list and explain the protocols within the IPSec suite that the SBC supports. This chapter does not explain how to design and choose the best protocols for your application.

AH vs. ESP

The SBC supports the two encapsulations that IPSec uses to secure packet flows. Authentication Header (AH) is used to authenticate and validate IP packets. Authentication means that the packet was sent by the source who is assumed to have sent it.

> **Note:**
>
> AH is incompatible with NAT. Validation means that the recipient is assured that the packet has arrived containing the original, unaltered data as sent.

ESP (Encapsulating Security Payload) provides AH's authentication and validations and extends the feature set by ensuring that the IP packet's contents remain confidential as they travel across the network. Using an encryption algorithm that both peers agree upon, ESP encrypts a full IP packet so that if intercepted, an unauthorized party cannot read the IPSec packet's contents.

Tunnel Mode vs. Transport Mode

In addition to its security encapsulations, the IPSec suite supports two modes: tunnel mode and transport mode. Tunnel mode is used most often for connections between gateways, or between a host and a gateway. Tunnel mode creates a VPN-like path between the two gateways and encapsulates the entire original IP packet. Transport mode is used to protect

end-to-end communications between two hosts providing a secured IP connection and encrypts just the original payload.

> **Note:**
>
> Traffic sent through the inner IPSec tunnel must be on the same VLAN-slot-port network-interface combination as where the outer tunnel is configured. This is because IPSec tunnel mode does not carry any L2 information for the inner packet. Once the SBC decrypts (de-tunnel) the received packet, it uses the L2 header from the original packet for the lookup. Therefore, if the SBC uses different vlan/slot/port for the inner network, lookups will fail.

Cryptographic Algorithms

IPSec works by using a symmetric key for validation and encryption. Symmetric key algorithms use the same shared secret key for encoding and decoding data on both sides of the IPSec flow. The SBC's IPSec feature supports the following encryption algorithms:

- aes-128-cbc
- aes-256-cbc
- aes-128-ctr
- aes-256-ctr

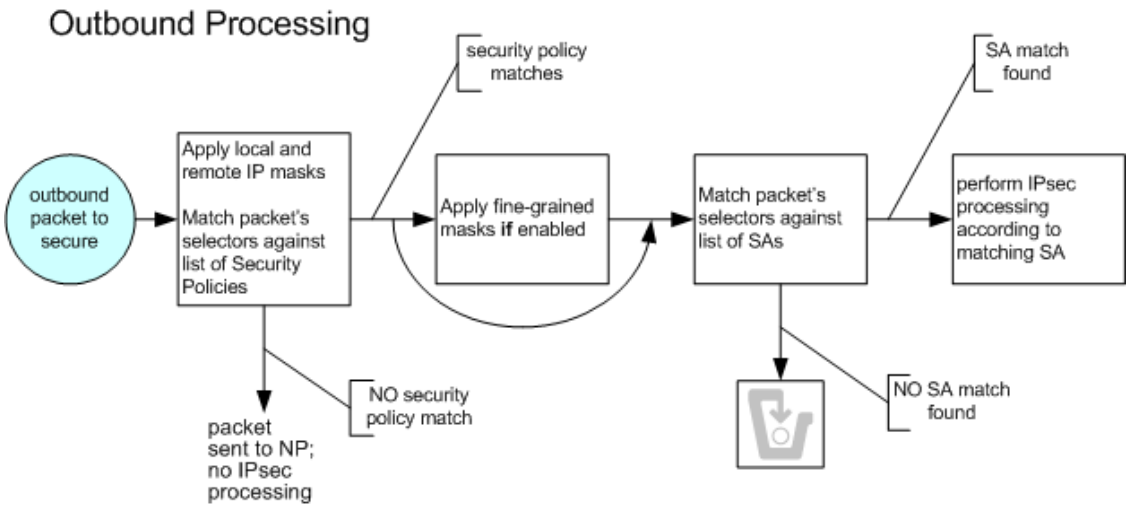
The SBC can quickly generate keys for all of the above mentioned algorithms from the CLI using the **generate-key** command. Only manual keying is currently supported for both hash authentication and data encryption. Therefore, all keys must be provisioned on the SBC by hand.

IPSec Implementation

The SBC uses separate logic for processing IPSec packets based on whether the traffic is inbound or outbound. The configuration is divided into two pieces, the security policy and the security association (SA). Both the SA and security policies have a directional attribute which indicates if they can be used and/or reused for inbound and outbound traffic.

Outbound Packet Processing

The following diagrams show the steps the SBC follows when processing outbound IPSec traffic. Details of each step are described in the following sections.



Security Policy

The SBC first performs a policy lookup on outbound traffic to test if it should be subjected to IPSec rules. A security policy, local policy applicable for IPSec functionality, defines the matching criteria for outgoing network traffic to secure. It is configured on a network interface basis.

Configuring a security policy is similar to a local policy, with additional IPSec-specific parameters. Unlike a local policy, used for routing, a security policy is used for packet treatment. As with a local policy, a set of selector values is matched against the outbound flow's following characteristics:

- VLAN
- Source IP address (plus mask)
- Source IP port
- Destination IP address (plus mask)
- Destination IP port
- Transport Protocol

Each of these selection criteria can be defined by a wildcard except for the VLAN ID, which can be ignored. This flexibility aids in creating selection criteria that ranges from highly restrictive to completely permissive. In addition to the main traffic matching criteria, a priority parameter is used to prioritize the order that configured security policies are checked against. The #0 policy is checked first, #1 policy is checked next, continuing to the lowest prioritized policy being checked last.

Once the outbound traffic matches a policy, the SBC proceeds to the next step of outbound IPSec processing. If no matching security policy is found, the default pass-through policy allows the packet to be forwarded to the network without any security processing.

Fine-grained policy Selection

After a positive match between outbound traffic and the configured selectors in the security policy, the Oracle Communications Session Border Controller can perform a calculation between a set of fine-grained packet selectors and the outbound packet. The fine-grained policy masking criteria are:

- Source IP subnet mask
- Destination IP subnet mask
- VLAN mask

By default, the fine-grained security policy is set to match and pass all traffic untouched to the security association (SA) portion of IPSec processing.

Fine-grained policy selection works by performing a logical AND between outbound traffic's fine-grained selectors and the traffic's corresponding attributes. The result is then used to find the matching SA. Applying a fine-grained mask has the effect of forcing a contiguous block of IP addresses and/or ports to appear as one address and or port. During the next step of IPSec processing, when an SA is chosen, the Oracle Communications Session Border Controller in effect uses one SA lookup for a series of addresses. Without fine-grained policy selection, unique SAs must always be configured for outbound packets with unique security policy selectors.

Security Associations

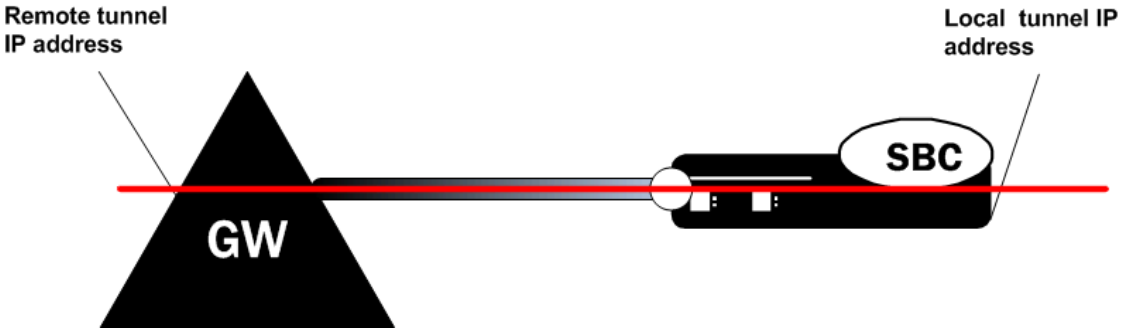
After the Oracle Communications Session Border Controller determines that outgoing traffic is subject to IPSec processing, and optionally applies fine-grained masking, an SA lookup is performed on the traffic. An SA is the set of rules that define the association between two endpoints or entities that create the secured communication. To choose an SA, the Oracle Communications Session Border Controller searches for a match against the outgoing traffic's SA selectors. SA selectors are as follows:

- VLAN
- Source IP address
- Source IP port
- Destination IP address
- Destination IP port
- Transport Protocol

If there is a match, the Oracle Communications Session Border Controller secures the flow according to security parameters defined in the SA that the Oracle Communications Session Border Controller chooses. The packet is then forwarded out of the Oracle Communications Session Border Controller. If no match is found, the packets are discarded, and optionally dumped to secured.log if the log-level is set to DEBUG.

Secure Connection Details

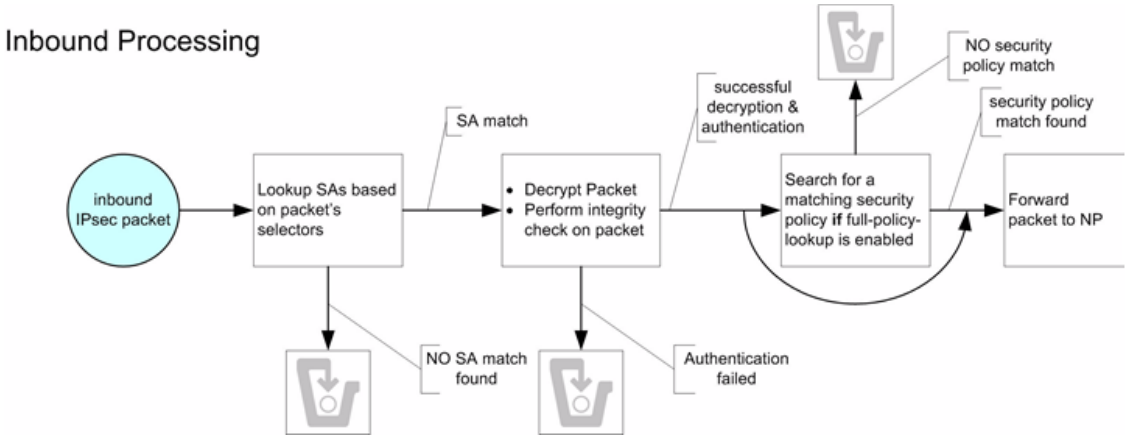
Several parameters define an IPSec connection between the SBC and a peer. When planning an IPSec deployment, the primary architectural decisions are which IPSec protocol and mode to use. The two choices for IPSec protocol are ESP or AH, and the two choices for IPSec mode are either tunnel or transport. IPSec protocol and mode are both required for an SA configuration. When creating an IPSec tunnel (tunnel mode), the SA must also define the two outside IP addresses of the tunnel.



The authentication algorithm and the authentication key must always be configured. The SBC supports hmac-sha2 authentication algorithms. Because only manual keying is supported, the key must be entered by hand. When encryption is required, the encryption algorithm and the encryption key must be configured. When using the two encryption protocols that operate in AES counter mode (RFC 3686), an additional nonce value is required. In addition, the security parameter index (SPI) must be configured for each SA. All SPI values mst be unique as well, across all SAs.

Inbound Packet Processing

The following diagram shows the steps the system follows when processing inbound IPSec traffic. Details of each step are described in the following sections.



IP Header Inspection

Processing inbound IPSec packets begins by the Oracle Communications Session Border Controller inspecting an inbound IP packet's headers. If the packet is identified as IPSec traffic, as determined by the presence of an AH or ESP header, an SA policy lookup is performed. If the traffic is identified as non-IPSec traffic, as determined by the lack of an IPSec-type (AH or ESP) header, it still is subject to a policy lookup. However, due to the default allow policy, the packet is forwarded directly to the NP, without any security processing.

SA Matching

The Oracle Communications Session Border Controller proceeds to match the inbound IPSec traffic's selectors against configured SAs. Inbound selector masking is performed where noted. These selectors are:

- VLAN (plus mask)
- Source IP address (plus mask)
- Source IP port
- Destination IP address (plus mask)
- Destination IP port
- Transport Protocol
- SPI

If no matching SA is found, the packets are discarded, and optionally dumped to secured.log if the log-level is set to DEBUG. When the Oracle Communications Session Border Controller finds a matching SA, the packet is authenticated and decrypted according to the configuration and sent to the Oracle Communications Session Border Controller's NP for continued processing.

Inbound Full Policy Lookup

Inbound traffic can optionally be subjected to a full policy lookup, after decryption and authentication. A full policy lookup checks if a security policy exists for this inbound traffic before the Oracle Communications Session Border Controller sends the decrypted packet to the NP. If no matching security policy is found, even after a successful SA match, the packets are discarded, and optionally dumped to secured.log if the log-level is set to DEBUG.

Full policy lookups are useful for traffic filtering. If you wish to drop traffic not sent to a specific port (e.g. drop any traffic not sent to port 5060), a security policy with specific remote-port-match parameter would be used to define what is valid (i.e., not dropped).

HA Considerations

Anti-replay mechanisms, running on IPSec peers, can cause instability with the Oracle Communications Session Border Controllers configured in an HA pair. The anti-replay mechanism ensures that traffic with inconsistent (non-incrementing) sequence numbers is labeled as insecure, assuming it could be part of a replay attack. Under normal circumstances, this signature causes the remote endpoint to drop IPSec traffic.

When a failover occurs between HA peers, the newly-active system starts sending traffic with the IPSec sequence number starting at 0. A remote system's anti-replay mechanism observes this and labels the traffic as insecure. It is therefore recommended that anti-replay protection not be used with Oracle Communications Session Border Controllers in an HA configuration. This situation does not create any problems as long as IPSec peers are not configured to use anti-replay mechanisms.

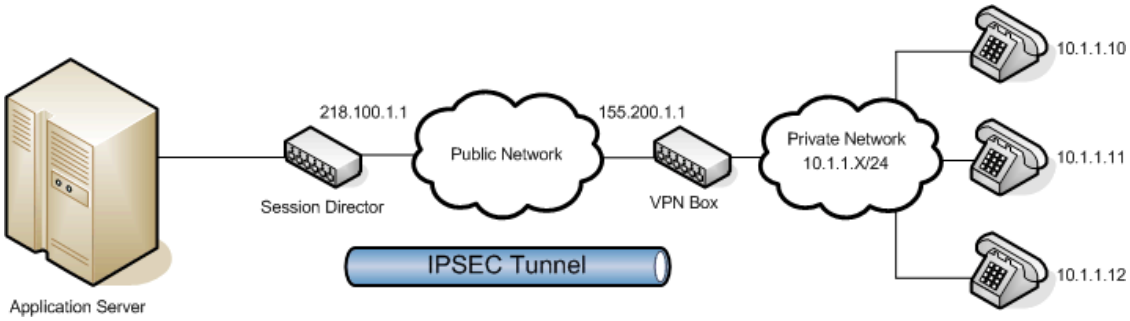
Packet Size Considerations

The security processor supports receipt of jumbo frames up to 9K (9022 bytes with VLANs, 9018 without). Under normal operation the default outgoing maximum packet size of 1500 bytes is used. This packet size includes the IPSec headers, which will result in less space for packet data (SIP signaling, RTP, etc...).

IPSec Application Example

In this example, the Oracle Communications Session Border Controller terminates an IPSec tunnel. The remote side of the tunnels is a dedicated VPN appliance in the public Internet. Behind that VPN appliance are three non-IPSec VoIP phones. In this scenario, the VPN box

maintains the IPSec tunnel through which the phones communicate with the Oracle Communications Session Border Controller.



Without the fine-grained option (or alternatively IKE), an SA entry would need to be configured for each of the three phones, communicating over the IPSec tunnel (resulting in 3 tunnels).

This does not scale for manual-keying with a large number of endpoints. Using the fine-grained configuration as well as the inbound SA mask allows any number of endpoints on the 10.1.1.X network to use a single security association (a coarse-grain configuration). The configuration in this example follows:

A packet sent from the Oracle Communications Session Border Controller to any of the phones will match the policy pol1. The remote-ip-mask parameter of the fine-grained configuration will then be masked against the remote-ip, resulting in a SA selector value of 10.1.1.0. This matches security-association sa1, and the packet will be secured and sent over the tunnel. The tunnel-mode addresses in the security-association represent the external, public addresses of the termination points for the IPSec tunnel.

Packets returning from the 10.1.1.0 side of the IPSec tunnel will first match the tunnel-mode local-ip-addr of 218.100.1.1. The packets will then be decrypted using the SA parameters, and the tunneled packet will be checked against the remote-ip-addr field of the SA.

If the fine-grained mask had not been used, three discrete SAs would have to be configured: one for each of the three phones.

```
ORACLE(manual)# show
        manual
                name                            assoc1
                spi                             1516
                network-interface               lefty:0
                local-ip-addr                   100.20.50.7
                remote-ip-addr                  100.25.56.10
                local-port                      60035
                remote-port                     26555
                trans-protocol                  ALL
                ipsec-protocol                  esp
                direction                       both
                ipsec-mode                      tunnel
                auth-algo                       hmac-sha-512
                encr-algo                       aes-256-cbc
                auth-key
                encr-key
                aes-ctr-nonce                   0
                tunnel-mode
                        local-ip-addr                   100.20.55.1
                        remote-ip-addr                  101.22.54.3
```

IPSec Configuration

The following example explains how to configure IPSec on your Oracle Communications Session Border Controller.

> **Note:**
>
> If you change the phy-interface slot and port associated with any SAs or SPDs, the Oracle Communications Session Border Controller must be rebooted for the changes to take effect.

Configuring an IPSec Security Policy

To configure an IPSec security policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter to access the security path of the configuration menu.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **ipsec** and press Enter.

   ```
   ORACLE(security)# ipsec
   ORACLE(ipsec)#
   ```

4. Type **security-policy** and press Enter. The prompt changes to let you know that you can begin configuration.

   ```
   ORACLE(ipsec)# security-policy
   ORACLE(security-policy)#
   ```

5. **name**—Enter a name for this security policy. This parameter is required and has no default.
6. **network-interface**—Enter the network interface and VLAN where this security policy applies in the form: interface-name:VLAN
7. **priority**—Enter the priority number of this security policy. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—3071
8. **action**—Enter the action the Oracle Communications Session Border Controller should take when this policy matches outbound IPSec traffic. The default value is **ipsec**. The valid values are:
   - **ipsec**—Continue processing as IPSec traffic
   - **allow**—Forward the traffic without any security processing

- **discard**—Discard the traffic

9. **direction**—Enter the direction of traffic this security policy can apply to. The default value is **both**. The valid values are:
   - **in**—This security policy is valid for inbound traffic
   - **out**—This security policy is valid for outbound traffic
   - **both**—This security policy is valid for inbound and outbound traffic

   To define the criteria for matching traffic selectors for this security policy:
10. **local-ip-addr-match**—Enter the source IP address to match. The default value is **0.0.0.0**.
11. **remote-ip-addr-match**—Enter the destination IP address to match. The default value is **0.0.0.0.**
12. **local-port-match**—Enter the source port to match. A value of 0 disables this selector. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—65535
13. **remote-port-match**—Enter the destination port to match. A value of 0 disables this selector. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—65535
14. **trans-protocol-match**—Enter the transport protocol to match. The default value is **all**. The valid values are:
    - **UDP | TCP | ICMP | ALL**
15. **local-ip-mask**—Enter the source IP address mask, in dotted-decimal notation. The default value is **255.255.255.255**.
16. **remote-ip-mask**—Enter the remote IP address mask, in dotted-decimal notation. The default value is **255.255.255.255.**
17. Save your work using the ACLI **done** command.

Defining Outbound Fine-Grained SA Matching Criteria

To define outbound fine-grained SA matching criteria:

1. From within the security policy configuration, type **outbound-sa-fine-grained-mask** and press Enter. The prompt changes to let you know that you can begin configuration.
2. **local-ip-mask**—Enter the fine-grained source IP address mask to apply to outbound IP packets for SA matching. Valid values are in dotted-decimal notation. The default mask matches for all traffic.
3. **remote-ip-mask**—Enter the fine-grained destination IP address mask to apply to outbound IP packets for SA matching. Valid values are in dotted-decimal notation. The default mask matches for all traffic.
4. **local-port-mask**—Enter the local port mask for this security policy. The default value for this parameter is **0**. The valid range is:
   - Minimum—0
   - Maximum—65535

5. **remote-port-mask**—Enter the remote port mask for this security policy. The default value for this parameter is **0**. The valid range is:
   - Minimum—0
   - Maximum—65535
6. **trans-protocol-mask**—Enter the transport protocol mask for this security policy. The default value for this parameter is **0**. The valid range is:
   - Minimum—0
   - Maximum—255
7. **vlan-mask**—Enter the fine-grained VLAN mask to apply to outbound IP packets for SA matching. The default is **0x000 (disabled).** The valid range is:
   - 0x000 - 0xFFF
8. Save your work using the ACLI **done** command.

Configuring an IPSec SA

To configure an IPSec SA:

1. Access the **manual** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ipsec
ORACLE(ipsec)# security-association
ORACLE(security-association)# manual
ORACLE(manual)#
```

2. **name**—Enter a name for this security policy.
3. **network-interface**—Enter the network interface and VLAN where this security policy applies in the form: interface_name:VLAN
4. **direction**—Enter the direction of traffic this security policy can apply to. The default value is **both**. Valid values are:
   - in | out | both
5. Save your work using the ACLI **done** command.

Defining Criteria for Matching Traffic Selectors per SA

To define the criteria for matching traffic selectors for this SA:

1. From within the **manual** portion of the security association configuration, you need to set the parameters described in this process.

```
ORACLE(security-association)# manual
ORACLE(manual)#
```

2. **local-ip-addr**—Enter the source IP address to match.
3. **remote-ip-addr**—Enter the destination IP address to match.
4. **local-port**—Enter the source port to match. A value of **0** disables this selector. The default value is **0**, disabling this parameter.

Valid range is:

- Minimum—0
- Maximum—65535

5. **remote-port**—Enter the destination port to match. A value of **0** disables this selector. The default value is **0**, disabling this parameter.

   Valid range is:

   - Minimum—0
   - Maximum—65535

6. **trans-protocol**—Enter the transport protocol to match for traffic selectors for this SA. The default value is **ALL**.

   Valid values:

   - UDP | TCP | ICMP | ALL

7. **ipsec-protocol**—Select the IPSec protocol to use for this SA configuration. The default value for this parameter is **esp**. Valid values are:

   - esp | ah

8. **spi**—Enter the security parameter index. The default value is **256**. The valid range is:

   Valid range:

   - Minimum—256
   - Maximum—4294967295

9. **ipsec-mode**—Enter the IPSec mode of this SA. The default value is **transport**. The valid values are:

   - tunnel | transport

10. **auth-algo**—Enter the IPSec authentication algorithm for this SA. The default value is **hmac-sha-512**.

    Valid values are:

    - null | hmac-sha-256 | hmac-sha-384 | hmac-sha-512 | aes-xcbc-mac

11. **auth-key**—Enter the authentication key for the previously chosen authentication algorithm for this SA.

> **Note:**
>
> The specified auth-key value will be encrypted in the configuration and will no longer be visible in clear-text.

12. **encr-algo**—Enter the IPSec encryption algorithm for this SA. The default value is **null**.

    Valid values are:

    - null | aes-128-cbc | aes-256-cbc | aes-128-ctr | aes-256-ctr

13. **encr-key**—Enter the encryption key for the previously chosen encryption algorithm for this SA.

> **Note:**
>
> The specified encr-key value will be encrypted in the configuration and will no longer be visible in clear-text.

14. **aes-ctr-nonce**—Enter the AES nonce if aes-128-ctr or aes-256-ctr were chosen as your encryption algorithm. The default value is **0**.

Defining Endpoints for IPSec Tunnel Mode

To define endpoints for IPSec tunnel mode:

1. From within the **manual** portion of the security association configuration, you need to set the parameters described in this process.

```
ORACLE(security-association)# manual
ORACLE(manual)#
```

2. **local-ip-addr**—Enter the local public IP address which terminates the IPSec tunnel.
3. **remote-ip-addr**—Enter the remote public IP address which terminates the IPSec tunnel.
4. Save your work using the ACLI **done** command.

Real-Time IPSec Process Control

The **notify secured** commands force the IPSec application to perform tasks in real-time, outside of the Oracle Communications Session Border Controller reloading and activating the running configuration. The **notify secured** usage is as follows:

```
notify secured [activateconfig | nolog | log | debug | nodebug]
```

The following arguments perform the listed tasks:

- **nolog**—Disables secured logging
- **log**—Enables secured logging
- **debug**—Sets secured log level to DEBUG
- **nodebug**—Sets secured log level to INFO

Key Generation

The **generate-key** command generates keys for the supported encryption or authentication algorithms supported by the Oracle Communications Session Border Controller's IPSec implementation. The generate-key commands generate random values which are not stored on the Oracle Communications Session Border Controller, but are only displayed on the screen. This command is a convenient function for users who would like to randomly generate encryption and authentication keys. The generate-key usage is as follows:

```
generate-key [hmac-md5 | hmac-sha1 | aes-128 | aes-256 | des | 3des]
```

IDS Reporting

The Oracle Communications Session Border Controller supports a wide range of intrusion detection and protection capabilities for vulnerability and attack profiles identified to date. The IDS reporting feature is useful for enterprise customers requirement to report on intrusions and suspicious behavior that it currently monitors.

Basic Endpoint Demotion Behavior

Each session agent or endpoint is promoted or demoted among the trusted, untrusted, and denied queues depending on the **trust-level** parameter of the session agent or realm to which it belongs. Users can also configure access control rules to further classify signaling traffic so it can be promoted or demoted among trust queues as necessary.

An endpoint can be demoted in two cases:

1. Oracle Communications Session Border Controller receiving too many signaling packets within the configured time window (**maximum signal threshold** in **realm config** or **access control**)
2. Oracle Communications Session Border Controller receiving too many invalid signaling packets within the configured time window. (**invalid signal threshold** in **realm config** or **access control**)

Endpoint Demotion Reporting

The Oracle Communications Session Border Controller counts the number of endpoint or session agent promotions and demotions. Further, the Oracle Communications Session Border Controller counts when endpoints or session agents transition from trusted to untrusted and when endpoints transition from untrusted to denied queues. These counts are maintained for SIP signaling applications. They appear as the Trust->Untrust and Untrust->Deny rows in the **show sipd acls** command.

SNMP Reporting

These per-endpoint counters are available under APSYSMGMT-MIB -> acmepacketMgmt -> apSystemManagementModule -> apSysMgmtMIBObjects -> apSysMgmtMIBGeneralObjects.

MIB NAME	MIB OID	PURPOSE
apSysSipEndptDemTrustToUntrust	.1.3.6.1.4.1.9148.3.2.1.1.19	Global counter for SIP endpoint demotions from trusted to untrusted.
apSysSipEndptDemUntrustToDeny	.1.3.6.1.4.1.9148.3.2.1.1.20	Global counter for SIP endpoint demotions from untrusted to denied.

HDR Reporting

The SIP (sip-ACL-oper) HDR ACL status collection groups include the following two metrics:

- Demote Trust-Untrust (Global counter of endpoint demotion from trusted to untrusted queue)

- Demote Untrust-Deny (Global counter of endpoint demotion from untrusted to denied queue)

Endpoint Demotion SNMP Traps

An SNMP trap can be sent when the Oracle Communications Session Border Controller demotes an endpoint to the denied queue. This is set by enabling the **trap on demote to deny** parameter located in the **media manager config** configuration element.

When the **trap on demote to deny** parameter is enabled, apSysMgmtInetAddrWithReasonDOSTrap trap is sent. This trap supersedes the apSysMgmtInetAddrDOSTrap trap.

When the **trap on demote to deny** parameter is disabled the apSysMgmtInetAddrWithReasonDOSTrap trap is not sent from the Oracle Communications Session Border Controller, even when an endpoint is demoted to the denied queue.

This apSysMgmtInetAddrWithReasonDOSTrap contains the following data:

- apSysMgmtDOSInetAddressType—Blocked IP address family (IPv4 or IPv6)
- apSysMgmtDOSInetAddress—Blocked IP address
- apSysMgmtDOSRealmID—Blocked Realm ID
- apSysMgmtDOSFromURI—The FROM header of the message that caused the block (If available)
- apSysMgmtDOSReason—The reason for demoting the endpoint to the denied queue: This field can report the following three values:
  - Too many errors
  - Too many messages
  - Too many admission control failures

> **Note:**
>
> By default, this parameter is enabled for upgrade configurations, as the current behavior is to send a trap for every endpoint that is demoted to deny. However, for a new configuration created, the value to this configuration is disabled.

Trusted to Untrusted Reporting

Endpoints, however, transition to an intermediate state, untrusted, prior to being denied service. The Oracle Communications Session Border Controller provides an ACLI parameter, **trap-on-demote-to-untrusted**, that generates an SNMP trap when a previously trusted endpoint transitions to the untrusted state. Trap generation is disabled by default.

SNMP Reporting

Endpoint state transitions continue to be tracked by two counters available under APSYSMGMT-MIB -> acmepacketMgmt -> apSystemManagementModule -> apSysMgmtMIBObjects -> apSysMgmtMIBGeneralObjects.

MIB NAME	MIB OID	PURPOSE
apSysSipEndptDemTrustToUntrust	.1.3.6.1.4.1.9148.3.2.1.1.19	Global counter for SIP endpoint demotions from trusted to untrusted.
apSysSipEndptDemUntrustToDeny	.1.3.6.1.4.1.9148.3.2.1.1.20	Global counter for SIP endpoint demotions from untrusted to denied.

Endpoint Demotion Trusted-to-Untrusted SNMP Trap

The system can generate an SNMP trap when an endpoint transitions from the trusted to the untrusted state. The trap is structured as follows.

```
apSysMgmtInetAddrTrustedToUntrustedDOSTrap NOTIFICATION-TYPE
OBJECTS { apSysMgmtDOSInetAddressType,
apSysMgmtDOSInetAddress,
apSysMgmtDOSRealmID,
apSysMgmtDOSFromUri,
apSysMgmtDOSReason }
STATUS current
DESCRIPTION
"This trap is generated when an IP is placed on a untrusted list from trusted
list, and provides the ip address that has been demoted, the realm-id of that
IP, (if available) the URI portion of the SIP From header of the message that
caused the demotion."
::= { apSysMgmtDOSNotifications 5 }
```

The trap OID is 1.3.6.1.4.1.9148.3.2.8.0.5.

Endpoint Demotion Syslog Message

The system can generate a Syslog message when an endpoint is demoted. Setting the **media manager config**, **syslog-on-demote-to-deny** parameter to enabled writes an endpoint demotion warning to the syslog every time an endpoint is demoted to the denied queue. By default, this configuration option is set to disabled. The following example shows a syslogWARNING Level message:

```
Jan 15 12:22:48 172.30.60.12 ACMESYSTEM sipd[1c6e0b90] WARNING
SigAddr[access:168.192.24.40:0=low:DENY] ttl=3632 guard=798 exp=30 Demoted to
Block-List (Too many admission control failures)
```

Event Log Notification Demotion from Trusted to Untrusted

You can enable your Oracle Communications Session Border Controller to provide event log notification (a syslog message) any time it demotes an endpoint from trusted to untrusted. The log message contains this data: IP address of the demoted endpoint, the endpoint's configured trust level, and the reason for demotion. This feature is enabled with the **syslog-on-demote-to-untrusted** parameter in the media manager.

Endpoint Demotion Configuration

To configure the Oracle Communications Session Border Controller to send traps and/or write syslog messages on endpoint demotion:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **trap-on-demote-to-deny**—Set this parameter to enabled for the Oracle Communications Session Border Controller to send the apSysMgmtInetAddrWithReasonDOSTrap trap when applicable.
5. **syslog-on-demote-to-deny**—Set this parameter to enabled for the Oracle Communications Session Border Controller to write an endpoint demotion warning message to the syslog.
6. **syslog-on-demote-to-untrusted**—Change this parameter from **disabled** (default), to **enabled** so the Oracle Communications Session Border Controller will generate event notifications (syslog messages) when an endpoint becomes untrusted. For this capability to work, the IDS license must be installed on your system.
7. **trap-on-demote-to-untrusted**—Set this parameter to enabled for the Oracle Communications Session Border Controller to send the apSysMgmtInetAddrTrustedToUntrustedDOSTrap when the endpoint identified within the trap transitions from the trusted to untrusted state.
8. Save your work.

Endpoint Demotion due to CAC overage

The Oracle Communications Session Border Controller can demote endpoints from trusted to untrusted queues when CAC failures exceed a configured threshold. The Oracle Communications Session Border Controller can also demote endpoints from untrusted to denied queues when CAC failures exceed a another configured threshold.

The Oracle Communications Session Border Controller maintains CAC failures per-endpoint. The CAC failure counter is incremented upon certain admission control failures only if either one of **cac-failure-threshold** or **untrust-cac-fail-threshold** is non-zero.

The **cac failure threshold** parameter is available in the access control and realm configuration elements. Exceeding this parameter demotes an endpoint from the trusted queue to the untrusted queue. The untrust cac-failure-threshold parameter is available in the access control and realm configuration elements. Exceeding this parameter demotes an endpoint from the untrusted queue to the denied queue.

If both the **cac failure threshold** and **untrusted cac failure threshold** are configured to 0, then admission control failures are considered and counted as invalid signaling messages for determining if the **invalid-signal-threshold** parameter value has been exceeded.

CAC Attributes used for Endpoint Demotion

The Oracle Communications Session Border Controller determines CAC failures only by considering the calling endpoint's signaling messages traversing the calling realms' configuration. If an endpoint exceeds the following CAC thresholds, the Oracle Communications Session Border Controller will demote the endpoint when the CAC failure thresholds are enabled.

1. sip-interface user CAC sessions (**realm-config**, **user-cac-sessions**)
2. sip-interface user CAC bandwidth (**realm-config**, **user-cac-bandwidth**)
3. External policy server rejects a session

Authentication Failures used for Endpoint Demotion

If an endpoint fails to authenticate with the Oracle Communications Session Border Controller using SIP HTTP digest authentication OR endpoint fails authentication with an INVITE with authentication incase registration-caching is disabled, and receives back a 401 or 407 response from the registrar

When the Oracle Communications Session Border Controller receives a 401 or 407 message from the registrar in response to one of the following conditions, the endpoint attempting authentication is demoted.

- endpoint fails to authenticate with the Oracle Communications Session Border Controller using SIP HTTP digest authentication
- endpoint fails to authenticate with the Oracle Communications Session Border Controller using INVITE message when registration-caching is disabled

Endpoint Demotion Configuration on CAC Failures

To configure endpoint demotion on CAC failures:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **access-control** and press Enter.

   ```
   ORACLE(session-router)# access-control
   ORACLE(access-control)#
   ```

   If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **cac-failure-threshold**—Enter the number of CAC failures for any single endpoint that will demote it from the trusted queue to the untrusted queue.
5. **untrust-cac-failure-threshold**—Enter the number of CAC failures for any single endpoint that will demote it from the untrusted queue to the denied queue.
6. Save your work.

IDS Phase 2 (Advanced Reporting)

This feature supplements the IDS reporting and protection services. IDS Phase 2 provides users with additional tools to identify, monitor, and control suspicious, and possibly, malicious traffic patterns.

Rejected SIP Calls

IDS Phase 2 provides tools to monitor and record rejected SIP calls. A sudden or gradual increase in such calls can, but need not, indicate malicious intent.

IDS Phase 2 provides a global counter that increments with each SIP INVITE or REGISTER message that is rejected by the Acme Packet Oracle Communications Session Border Controller, and offers the option of generating a syslog message in response to call rejection.

Rejected Calls Counter

The rejected calls counter is a 32-bit global counter that records the total number of rejected SIP calls. Such calls have been rejected by the Oracle Communications Session Border Controller with the following response codes: 400, 403, 404, 405, 408, 413, 416, 417, 420, 423, 480, 481, 483, 484, 485, 488, 494, 500, 503, 505, and 604. These response codes may change with future software revisions.

The current value of the rejected calls counter is accessible via SNMP, Historical Data Recording (HDR), or the ACLI. This MIB table is apSysMgmtGeneralObjects Table (1.3.6.1.4.1.9148.3.2.1.1).

Object Name	Object OID	Description
apSysSipTotalCallsRejected	1.3.6.1.4.1.9148.3.2.1.1.25	Global counter for SIP calls that are rejected by the SBC

The sip-error HDR collection group contains a new reporting field, Call Rejects, which contains the value of the global rejected calls counter.

The ACLI command **show sipd errors** displays the contents of the rejected calls counter.

```
ORACLE# show sipd errors
12:29:13-131
SIP Errors/Events             ---- Lifetime ----
                        Recent      Total  PerMax
SDP Offer Errors             0          0       0
SDP Answer Errors            0          0       0
Drop Media Errors            0          0       0
Transaction Errors           0          0       0
Application Errors           0          0       0
Media Exp Events             0          0       0
Early Media Exps             0          0       0
Exp Media Drops              0          0       0
```

```
Expired Sessions            0          0       0
Multiple OK Drops           0          0       0
Multiple OK Terms           0          0       0
Media Failure Drops         0          0       0
Non-ACK 2xx Drops           0          0       0
Invalid Requests            0          5       2
Invalid Responses           0          0       0
Invalid Messages            0          0       0
CAC Session Drop            0          0       0
Nsep User Exceeded          0          0       0
Nsep SA   Exceeded          0          0       0
CAC BW Drop                 0          0       0
Calls Rejected              0          0       0 <--
```

Syslog Reporting of Rejected Calls

Users can choose to send a syslog message in response to the rejection of a SIP call. In the default state, rejected calls are not reported to syslog.

Use the following ACLI command sequence to enable syslog reporting of rejected SIP calls.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)# syslog-on-call-reject enable
```

The syslog-on-call-reject attribute, which is disabled by default, enables the generation of a syslog message in response to the rejection of a SIP call.

Use **done**, **exit**, and **verify-config** to complete this configuration.

Syslog messages issued in response to call rejection contain the following call-related information.

- SIP status code indicating rejection cause
- SIP method name (INVITE or REGISTER)
- Reason for denial
- Realm of calling endpoint
- Applicable local response map
- Content of Reason header (if present)
- From URI of calling endpoint
- Target URI of called endpoint
- Source and Destination IP address and port
- Transport type

The following are sample syslog messages issued in response to call rejections.

Dec 8 06:05:42 172.30.70.119 deimos sipd[205bfee4] ERROR [IDS_LOG]INVITE from source 172.16.18.100:5060 to dest 172.16.101.13:5060[UDP] realm=net172; From=sipp <sip:sipp@172.16.18.100:5060>;tag=13890SIPpTag001; target=sip:service@172.16.101.13:5060 rejected!; status=483 (Too Many Hops)

Dec 10 15:09:28 172.30.70.119 deimos sipd[2065ace8] ERROR [IDS_LOG]INVITE from source 172.16.18.5:5060 to dest 172.16.101.13:5060[UDP] realm=net172; From=sipp <sip:sipp@172.16.18.5:5060>;tag=10015SIPpTag001; target=sip:service@172.16.101.13:5060 rejected!; status=488 (sdp-address-mismatch); error=sdp address mismatch

IDS syslog messages that report rejected calls and those that report endpoint demotions now contain a string IDS_LOG, to facilitate their identification as IDS-related messages. With IDS Phase 2, IDS messages reporting either endpoint demotions or call rejections can be sent to specific, previously-configured syslog servers.

In topologies that include multiple syslog servers, use the following procedure to enable delivery of IDS-related messages to one or more specific syslog servers.

1. Use the following command sequence to move to **syslog-config** Configuration Mode.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# system-config
ORACLE(system-config)# syslog-servers
ORACLE(syslog-config)#
```

2. From the existing pool of syslog servers select the server or servers that will receive syslog messages.
3. Ensure that all selected servers are configured with the same value for the **facility** attribute.

   Allowable values are integers within the range 0 through 23.
4. Use the following command sequence to move to **system-config** Configuration Mode.

```
ORACLE(syslog-config)# done
ORACLE(syslog-config)# exit
ORACLE(system-config)#
```

5. Use the **ids-syslog-facility** attribute to enable message transfer to specific syslog servers.

   The default value, -1, disables selective message transfer. To enable transfer to a designated syslog server or servers, enter the facility value (an integer within the range 0 through 23) that you confirmed or set in Step 3.

   The following example enables the transfer of IDS syslog messages to all servers with a facility value of 16.

```
ORACLE(system-config)# ids-syslog-facility 16
ORACLE(system-config)#
```

6. Use **done**, **exit**, and **verify-config** to complete this configuration.

TCA Reporting of Denied Entries

You can construct a Threshold Crossing Alarm (TCA), which issues minor, major, and critical system alarms when the count of denied entries exceeds pre-configured values. For each issued alarm, the TCA also transmits an SNMP trap that reports the alarm state to remote SNMP agents.

After issuing a system alarm and accompanying SNMP trap, the TCA continues to monitor the number of denied entries. If the number of denied entries rises to the next threshold value, a

new, and more severe, system alarm/SNMP trap is generated. If the number of denied entries falls below the current threshold level, and remains there for a period of at least 10 seconds, a new, and less severe system alarm/SNMP trap is generated.

1. Use the following command sequence to move to **media-manager-config** Configuration Mode.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# system-config
ORACLE(system-config)# alarm-threshold
ORACLE(alarm-threshold)#
```

2. Use the **type** attribute to specify the TCS type (**deny-allocation** for denied entries TCAs), the **severity** attribute to specify the criticality of the alarm, and the **value** attribute to specify the alarm threshold.

   The following ACLI sequence defines the minor, major, and critical alarm thresholds. Not that you do not need to configure all three thresholds. Given the static deny allocation value of 32000, you can determine what the percentage value maps to.

```
ORACLE(alarm-threshold)# type deny-allocation
ORACLE(alarm-threshold)# severity minor
ORACLE(alarm-threshold)# value 80
ORACLE(alarm-threshold)# done
ORACLE(alarm-threshold)# type deny-allocation
ORACLE(alarm-threshold)# severity major
ORACLE(alarm-threshold)# value 90
ORACLE(alarm-threshold)# done
ORACLE(alarm-threshold)# type deny-allocation
ORACLE(alarm-threshold)# severity critical
ORACLE(alarm-threshold)# value 95
ORACLE(alarm-threshold)# done
```

3. Use **exit** and **verify-config** to complete the configuration.

Syslog Reporting of Denied Entries

Syslog reporting of endpoint demotions was introduced as part of IDS Phase 1 in S-C6.2.0. With IDS Phase 2, such syslog messages contain the last SIP message from the endpoint that caused the transition to the denied state. If the included SIP message increases the length of the syslog beyond 1024 bytes, the SIP message is truncated so that the syslog is no larger than 1024 bytes.

CPU Load Limiting

The transmission of IDS-related system alarms and SNMP traps is disabled when the CPU utilization rate surpasses a configured threshold percentage, reducing system resource utilization. When the threshold is exceeded, a syslog message (MINOR level) announces the termination of IDS reporting. No additional syslog messages or SNMP traps are generated until the CPU utilization rate falls below the configured threshold. The resumption of IDS reporting is announced by another syslog message, also issued at the MINOR level.

The system manages percent CPU utilization as follows:

- Begins rejecting SIP requests when the CPU reaches its throttling threshold, and

- Rejects all SIP requests, as well as stops sending IDS-related system alarms and SNMP traps, when the CPU reaches its maximum.

See the *SMP-Aware Task Load Limiting* section in the *Oracle® Communications Session Border Controller Maintenance and Troubleshooting Guide* for information on how this works and how the user can configure the CPU throttling threshold and maximum CPU utilization.

Denied Endpoints

IDS Phase 2 provides a denied endpoint counter that includes SIP endpoints. The global counter value is available via SNMP or HDR.

The global counter value is available to SNMP under APSYSMGMT-MIB, acmepacketMgmt, apSystemManagementModule, apSysMgmtMIBObjects, apSysMgmtMIBGeneralObjects. This MIB is apSysMgmtGeneralObjects Table (1.3.6.1.4.1.9148.3.2.1.1).

Object Name	Object OID	Description
apSysCurrentEndptsDenied	1.3.6.1.4.1.9148.3.2.1.1.26	Global counter for current endpoints denied

The system HDR collection group contains a new reporting field, Current Deny Entries Allocated, which contains the value of the global endpoints denied counter.

Maintenance and Troubleshooting

show sipd acls

The **show sipd acls** command includes counters that track the number of endpoints demoted from trusted to untrusted and the number of endpoints demoted from untrusted to denied. For example:

```
ORACLE# show sipd acls
...
ACL Operations          ---- Lifetime ----
                Recent      Total  PerMax
...
Trust->Untrust       0          1       1
Untrust->Deny        0          1       1
```

15

R226 Security Recommendation Compliance

The Oracle Communications Session Border Controller (SBC) provides functionality designed to comply with the R226 recommendations, a set of Information Technology Security Standard developed by the National Cybersecurity Agency of France (ANSSI)

This chapter presents the following features, which align with R226 recommendations to harden the operational security of the SBC. The features presented here require that you enable the **ANSSI R226 Compliance** entitlement.

- Bootparameter Security
- SIPREC Licensing—SIPREC cannot be used on a system without a license. The purpose of this is to present a barrier that requires external approval before an SBC user can configure and use SIPREC.
- SFTP Access Restrictions

SBC features associated with R226 compliance, but not requiring the **ANSSI R226 Compliance** entitlement are listed below with pointers to their description and configuration documentation:

- Restricting Logon to TACACS
- TACACS+ over IPsec for wancom0
- SHA-2 Authentication-Password Hashing
- Manage SSH Keys
- Secure the ACP Communication Link with TLS
- The set-boot-loader, backup-boot-loader, and delete-boot-file command

Bootparam Security

An Oracle Communications Session Border Controller ignores attempts to modify security related boot flags from the ACLI. The SBC still supports changing security related boot flags through the bootloader.

Table 15-1 Security Related Boot flags

Boot flag	Description
0x00000001	Disable all security filtering on all network interfaces
0x00000010	Enable direct Linux login on port 2200 via SSH for debugging
0x00000020	Enable the debug console
0x01000000	Enable SFTP access to protected files and directories
0x20000000	Enter failsafe mode
0x40000000	Boot directly to the Linux shell

R226 and SIPREC License Management

Enabling the R226 Certification self-entitlement disables the ability to enable SIPREC through the self-entitlement mechanism.

Once the R226 entitlement is enabled:

- The R226 entitlement cannot be disabled through the self-entitlement mechanism.
- Any previously installed SIPREC entitlement is flushed from the system.
- SIPREC is no longer an option under the `setup entitlements` command.
- SIPREC can only be enabled with a license key.

To disable the R226 self-entitlement, contact Oracle Support and follow the factory reset instructions in the *Admin Security Guide*.

SFTP Access Restrictions

In the default restricted mode, the user and admin factory accounts are restricted from adding, deleting, renaming, modifying, viewing, or listing sensitive system files when accessing the file system with SFTP. Set the boot flag to `0x01000000` to allow access to sensitive files. If the **ANSSI R226 Compliance** entitlement is enabled, boot flags can only be set through the bootloader during a reboot.

SHA-2 Authentication-Password Hashing

The Oracle Communications Session Border Controller supports SHA-2 hashing of user login passwords. The SBC hashes passwords using a randomly generated salt with 65532 iterations of the SHA-512 algorithm.

Enabling SHA-2 Password Hashing

Passwords are changed with the **secret login** command. All newly set passwords are hashed with SHA-2, the SHA-1 hash is removed, and thereafter the SBC uses SHA-2 to validate the password for that user. Oracle recommends that all users change their passwords after upgrading the system.

> **WARNING:**
>
> Regarding upgrades to this software, versions of Session Deliver Manager prior to SDM 8.1 do not support managing SHA-2 enabled SBCs. To manage an SBC, you must use SDM 8.1 with basic authentication.

> **WARNING:**
>
> If you downgrade to a release that only supports SHA-1 hashing after a user login password has been SHA-2 hashed, users will be locked out until all passwords are cleared. To clear passwords, contact Oracle Support.

16

Lawful Intercept

This section summarizes options for configuring the lawful intercept feature. It describes how the Oracle Communications Session Border Controller interoperates with mediation equipment from vendors who build lawful intercept equipment.

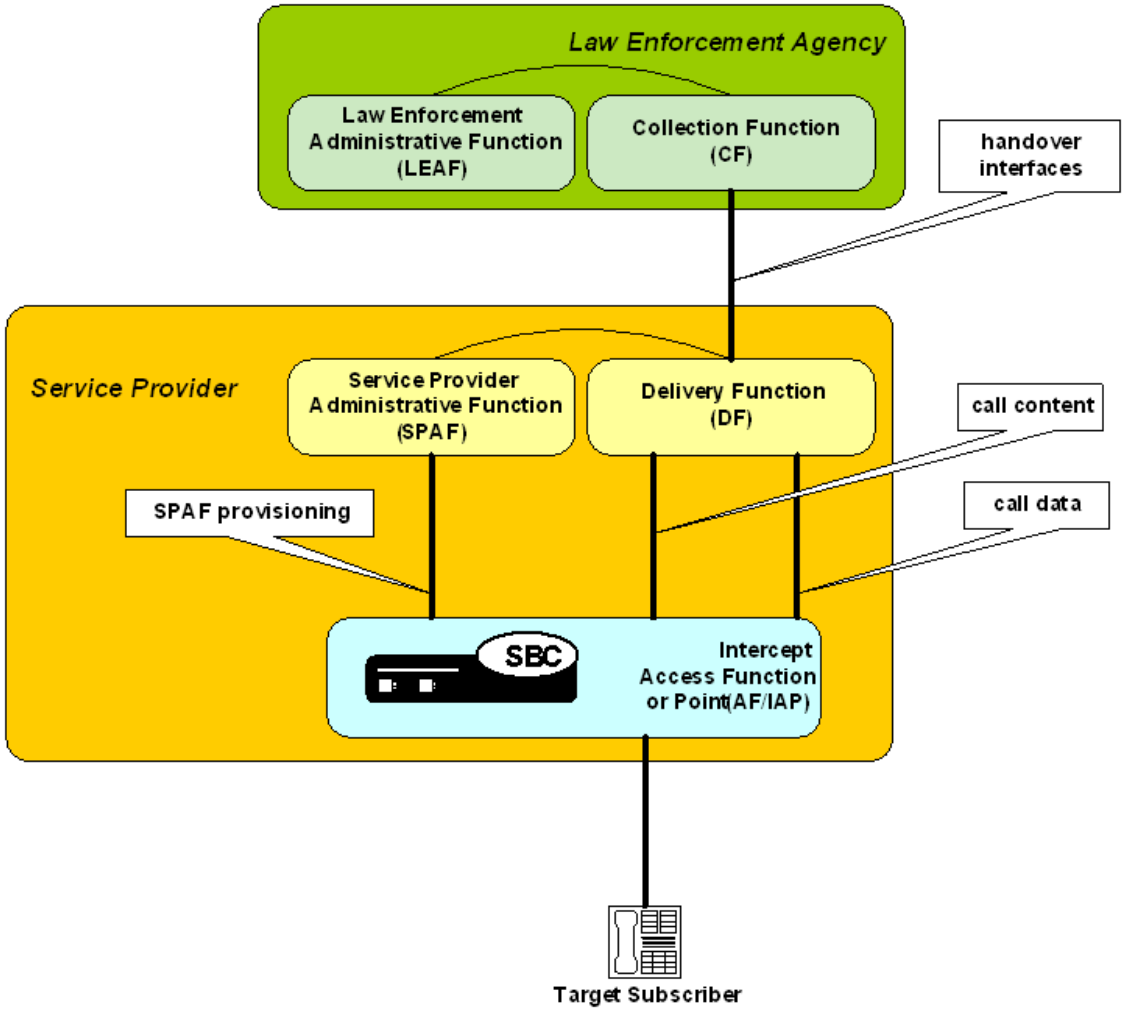
LI/CALEA consists of the interception of call content and/or the interception of call-identifying information. It requires that call information and media streams be sent to one or more law enforcement agencies in accordance with warrants for lawful interception.

You can configure your Oracle Communications Session Border Controller to support LI/CALEA functionality, enabling the Oracle Communications Session Border Controller to play a role in your Lawful Interception solution. Acting as an intercept access point (IAP), the Oracle Communications Session Border Controller can provide call data and can replicate media when signaling and media are anchored at the Oracle Communications Session Border Controller.

The Oracle Communications Session Border Controller supports LI/CALEA functionality that:

- Ensures unobtrusive intercept by hiding the network-based intercept of call information and content through topology hiding and media relay or NAT
- Intercepts and forwards call information and call content
- Interfaces with the mediation equipment [service provider administrative function (SPAF) and delivery function (DF)] for legal intercept

The following diagram provides one example of the Oracle Communications Session Border Controller deployed as part of a service provider's lawful intercept solution.



Recommendations

Calls may be lawfully intercepted by different devices in the service provider's network based on specific call flows, involvement of the device in the invoked service and where devices sit in the flow. Oracle recommends that you contact our Professional Services department to plan your use of the lawful intercept feature on your Oracle Communications Session Border Controller. Oracle Professional services can assist you with network/call flow analysis to determine which types of calls will involve the Oracle Communications Session Border Controller as an intercept access point and to recommend proper configuration.

Interoperability Using X1 X2 X3

This document describes how the Oracle Communications Session Border Controller supports X1, X2, and X3 interfaces for lawful interception of SIP calls. In this deployment, the Oracle Communications Session Border Controller acts as an interception point that receives provisioning information from an administrative function and, based on that information, provides call data and content. As with the other LI interoperability solutions that the Oracle Communications Session Border Controller supports, the X1, X2, and X3 interfaces ensure unobtrusive call intercept by hiding network-based intercept of call data and content. The Oracle Communications Session Border Controller supports intercept of call data only, or of call data and call content.

17

External Policy Servers

Diameter-based External Policy Servers

The Diameter base protocol (RFC 3588) is supported by the Oracle Communications Session Border Controller and is used for Resource and Admission Control Function (RACF) and Connectivity Location Function (CLF) applications.

Diameter Connection

The Oracle Communications Session Border Controller (SBC) supports Diameter (RFC 3588) connections to a Diameter server over TCP or SCTP. The base Diameter protocol runs on port 3868 for both TCP and SCTP. Diameter-based RACF and CLF are available from the media interfaces on the SBC.

The Diameter connection is always initiated from the SBC to the Diameter server. The SBC begins the connection by sending a Capabilities-Exchange-Request (CER) to the server, which replies with Capabilities-Exchange-Answer (CEA) message.

SCTP Support for the Rx Interface

You can configure the SBC to support Rx interface connections to external policy servers using SCTP transport as an alternative to TCP. Each server provides for this configuration and generates configuration errors when the server configuration is not appropriate for SCTP. All servers also use the global SCTP configuration within the **network-parameters** element, with the exception of **sctp-send-mode**, which you can configure within the **ext-policy-server** element. After transport is set up, diameter works the same as over TCP.

As is true for TCP, the SBC uses the first interface configured in the external policy server's realm to setup connections. For redundancy, however, you can configure SCTP multihoming and use multihoming targets provided to the SBC by the SCTP handshake with the peer. The SBC can access these targets via any interface in the external policy server's realm.

See a description of how SCTP works on the SBC in Stream Control Transfer Protocol Overview.

Additional Diameter Connection Compliance for the Rx Interface

When handling some Register and Message flows, the SBC default behavior does not include strict compliance with Diameter session teardown rules. Typically, the environment can proceed without issue, but the SBC provides an **ext-policy-server** option, called **diam-rx-strict-compliance**, that provides better compliance with Diameter session teardown rules.

Set the **diam-rx-strict-compliance** option on the applicable **ext-policy-server** to establish the following Diameter session behavior:

- For Register flows that do not establish a Diameter session with the PCRF due to a 3xxx, 4xxx or 5xxx error from the PCRF, the SBC does not send an STR to tear down the session when it receives a De-Register.

- For Message flows, when the SBC receives an ASR from PCRF, it stops the hold timer if it is running, then forwards the MESSAGE to the core, and sends an ASA with success.
- For unsuccessful Register flows that include an established Diameter session with the PCRF, the SBC sends an STR to tear down the session after the Register has failed due to, for example, responses from the core.
- If the SBC receives an S8HR Emergency Registration, with or without Authorization header, and either the Rx interface is not available or there is an error in AAA response sent by PCRF, the SBC replies with a 5xx response.
- If the SBC receives an S8HR Emergency Registration without an Authorization header and the EPC identities validation fails, the SBC sends a 403 error with the SIP reason header. If the SBC receives a REGISTER request with the authorization header, it sends a MIME XML body with a reason tag.
- For S8HR registrations and calls, the SBC adds the P-Visited-Network-ID header using the format "plmnIdPrefix.mncxxx.mccxxx.3gppnetwork.org".
- During an S8HR registration scenario, if the SBC receives a REGISTER request with the Authorization header and the next-hop is not configured, the SBC sends a 403 response if the EPC identities validation fails.
- When the SBC experiences a diameter transaction timeout from the PCRF, it does not issue an STR.
- Within emergency REGISTER call flows when S8HR is enabled and there are no EPC level identities cached, the SBC does not issue an STR simultaneously with a 403 error code if it receives a 3002 error code from the PCRF.

Syntax for this option may or may not include the plus (+) sign, but note that setting the option with the + sign retains all other options set on the element. Omitting the + sign replaces any existing options with the one you set.

```
ACMEPACKET(ext-policy-server)# options +diam-rx-strict-compliance
```

HA Support

The Oracle Communications Session Border Controller's high availability (HA) capabilities support CAC. When one Oracle Communications Session Border Controller in an HA configuration goes out of service, the MAC addresses are reassigned to a healthy Oracle Communications Session Border Controller. IP addresses follow the MAC addresses to provide a seamless switchover between HA nodes.

After an HA failover, the Diameter connection on the primary Oracle Communications Session Border Controller is either gracefully torn down, or times out depending on behavior of the PDP. The backup Oracle Communications Session Border Controller attempts to create a new Diameter connection with the PDP.

FQDN Support

You can configure FQDNs in the **address** parameter in the external policy server configuration element. These FQDNs must conform to RFC 1035. If the port parameter in external policy server configuration element is not zero, then it will be used to connect to the group of applicable external policy servers. If the port parameter is set to zero, then the port number returned in SRV RR from the DNS server will be used. For external policy servers using TCP transport, the Oracle Communications Session Border Controller (SBC) always appends the

_diameter._tcp. to request only TCP based Diameter servers to DNS queries triggered by FQDN configurations. For SCTP, the SBC appends _diameter._sctp.

There are differences in the way the SBC uses FQDNs between TCP and SCTP. For example, TCP uses FQDNs to establish server redundancy, whereas SCTP uses multihoming configurations. It is also important to note that the SBC only sends these DNS queries out to a DNS server configured on the first interface in the realm configuration.

The Stream Control Transfer Protocol Overview provides a thorough description of how this protocol works on the SBC.

IPv6 Support

An external policy server configuration element with an application mode parameter set to Rx may be configured with an IPv6 address in the address parameter (in addition to an IPv4 address or FQDN).

Diameter Heartbeat

Device-Watchdog-Request (DWR) and Device-Watchdog-Answer (DWA)messages are used to detect transport failures at the application layer between the Oracle Communications Session Border Controller communicating with a policy server via Diameter. The request/answer message pair forms a heartbeat mechanism that can alert the requesting side if the answering side is not reachable.

The Oracle Communications Session Border Controller always responds to a DWR by replying with a DWA message. In addition, the Oracle Communications Session Border Controller can be configured to initiate DWR messages toward a policy server or other Diameter-based network device.

You configure the **watchdog ka timer** with a timeout value that determines the number of seconds a DWA is expected in response to the Oracle Communications Session Border Controller sending a DWR.

If the Oracle Communications Session Border Controller fails to receive a DWA response from a Policy Server within the configured interval, then the connection towards that Policy Server is considered failed and torn down. The Oracle Communications Session Border Controller attempts to recreate the transport connection, followed by the recreating the Diameter connection by issuing a CEA toward the policy server.

Diameter Failures

During periods of application inactivity on the Diameter interface, Device-Watchdog-Request (DWR) and Answer (DWA) messages are exchanged between the client and server to provide an application-level heartbeat. DWRs may be sent toward the Oracle Communications Session Border Controller (SBC), which responds with a DWA message.

Prior to establishing a connection over TCP transport, the system tries to connect to a configured Diameter server using the default TCP retry interval of 10 seconds. This is also true if either side of the connection receives a TCP FIN or RST. The SBC labels a connection as failed when it does not receive a DWR from the Diameter server within the guard timer period.

If an operational Diameter connection over TCP transport fails, the SBC tries to re-open the TCP socket and Diameter connection to the Diameter server at 30 second intervals, and increases its retry interval to 5 minutes until a successful Diameter connection is made.

The SBC also supports SCTP transport for Rx interface connections. The SBC performs similar procedures to monitor diameter connections over SCTP, using SCTP's standard transport mechanisms, described in SCTP Monitoring Failure Detection and Recovery.

Application IDs and Modes

Diameter messages include an application ID to indicate the application and standards' body interface. The following table lists the different Application-IDs for the corresponding standards' and applications. Application IDs must be provisioned manually.

N/A	Standards Reference Point	Standards Reference Point	Standards Reference Point	Standards Reference Point
N/A	**RACF**	**RACF**	**RACF**	**CLF**
Reference Point/ Standards Body	Gq 3GPP R6 29.209	Rx 3GPP R15 29.214	Rq ETSI 283 026	e2 ETSI 283 035
Application-ID	16777222	16777229	16777222	16777231

You also set the application mode to specify the interface more precisely. Doing so avoids the potential for collision between interface types that can occur when you only configure the application identifier. By setting both the application mode and application identifier for the interface, you tell the Oracle Communications Session Border Controller the format for Diameter messages it sends and receives.

The following table describes the application mode settings.

Application Mode Type	Description
Rq	As the default mode for the interface, Rq is the Oracle Communications Session Border Controller's base RACF interface. Even when you leave the application mode set to none (default), the Oracle Communications Session Border Controller runs in Rq mode. The only exception to this rule is if you set the application identifier to 16777236 and leave the application mode set to none; in this instance, the interface runs in Rx mode.
Rx	The interface runs in Rx mode when you either: Set the application mode to Rx and the application identifier to 16777236 Leave the application mode set to none, and set the application identifier to 16777236
Gq	The interface runs in Gq mode when you set the application mode to Gq. **Note:** The application identifier 1677722 is no longer unique, but applies both to Rq and Gq interface modes.
e2	The interface runs in e2 mode, the base CLF interface, when you set the application mode to e2. Even if you leave the application mode set to none, the interface will run in e2 mode when the external policy server is configured as a CLF interface.
none	The interface runs in Rq mode when you do not configure an application identifier, or in Rx mode if you set the application identifier to 16777236.

Asynchronous SIP-Diameter Communication

The Oracle Communications Session Border Controller's Diameter-based external policy server support now offers an asynchronous mode in which the SBC does not wait for a Diameter Authorization-Authentication Answer (AAA) response to an Authorization-Authentication Request (AAR) before allowing the SIP 200 OK to proceed through the SBC.

One of the fundamental behaviors in the Oracle Communications Session Border Controller's Diameter model relies on the external policy server making an authorization decision which is then communicated to the SBC. Part of the call authorization sequence of events involves the SBC waiting for an external policy server's response before the SBC can completely set up, modify, or update the call. The long pause that the endpoints experience, while the SBC holds up SIP flows waiting for the external policy server's response, can lead to unnecessary call failure situations.

In some Diameter-based external policy server deployments, the media traverses a Cable Modem Termination System (CMTS) at the edge of the network; the CMTS gates may be established by a Policy Server to dynamically enable QoS from a UE toward another UE. If no gate is established then the media traverses the CMTS and is admitted to the network with a "best-effort" network path.

As QoS sessions might not be the most important priority to a network, Oracle now allows network operators the ability to decouple the call set up (signaling) from the request for bandwidth. The SBC's default external policy server model is the synchronous model, in which the SBC sends a policy server request based on a SIP or SDP trigger point, and the SIP signaling is held until a response is returned from the Policy Sever. In the asynchronous model the request that the SBC sends to the Policy Server flows in an asynchronous state with respect to SIP messaging; that is, the SBC allows the SIP session to proceed naturally, and does not pause for outstanding Policy Server answers to be received. The establishment of a SIP session is not affected by Policy Server answers, or answer timeouts, related to the SIP session. To enable the asynchronous model, the new parameter **asynchronous-mode** has been added to the **ext-policy-server** configuration element, with a default of **disabled** so as not to affect current default behavior.

> **Note:**
>
> Oracle Communications recommends that, for each SBC, the same model be used for all external policy server configuration instances. Failure to follow this guideline could result in complex interactions from a timing perspective which might lead to dropped or degraded calls.

SIP-Diameter Communication Configuration

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. **asynchronous-mode** — identifies whether to use the asynchronous mode of signaling on the external policy server interface rather than the default synchronous mode. Allowable values are **enabled** and **disabled**. The default is **disabled.**
4. Type **done** to save your configuration.

Serialized Diameter Messaging

After the Oracle Communications Session Border Controller sends a Diameter request, it will not send another Diameter request, such as a Session Termination Request (STR), with the same Session-ID Attribute Value Pair (AVP) until the original request receives a response or times out.

Access networks with complex policy server structures can allow non-sequential delivery of Diameter requests into a Cable Modem Termination System (CMTS), even if Diameter message delivery was correctly ordered on the TCP connection between the SBC and a lower-tiered external policy server.

The SBC now prevents the external policy server from receiving out-of-order messages at the application layer by serializing them. The SBC serializes Diameter messages to ensure that a Diameter request for one session-ID is not sent until an answer is received from the previous request for the same session-ID. The SBC applies this constraint while waiting for a Diameter response or when considering a Diameter request timeout (15 seconds). Serialized Diameter messaging is always enforced for Diameter-based external policy server communication.

Flow-Description AVP for Media Release

The Rx interface between the Oracle Communications Session Border Controller (SBC) and the Policy Server (PS) assumes that the media is always managed by the SBC and that the IP address and port number of one end of a service flow will always correspond to one present on the SBC. However, there are times when the media is released by the SBC, but a policy server request is still required. In these cases the flow descriptions should accurately represent the IP addresses of the two endpoints instead of that of the SBC. This feature lets the user configure the SBC to change the payload of the Flow-Description Attribute Value Pair (AVP) in the Diameter AAR messaging from the SBC to the PS, depending on whether the media is managed or released by the SBC.

In the case where the media is released, only incomplete flow information may be provided to the Policy Server because not all IP addresses and port numbers are known from the SDP offer. Media release is enabled on a per realm basis with the following settings in the **realm-config** configuration element:

```
mm-in-realm                    disabled
mm-in-network                  disabled
mm-in-system                   disabled
mm-same-ip                     disabled
msm-release                    enabled
```

When the realm is configured for external bandwidth management, the media layer checks if any of the configuration parameters for media release have been invoked. If none of the media release parameters are invoked (meaning that the SBC is managing the media), then the signaling application constructs the bandwidth request to the PS as it currently does. If the media layer detects that the realm is configured to possibly release media, then a few more operations are performed to correctly populate the bandwidth request to the PS. If the media has been released for the session, the signaling application inserts the IP port of the called endpoint into the bandwidth request instead of the IP port for the SBC. If the media has not yet been released, the media layer determines if the initial signal is an OFFER or an ANSWER. If it is an ANSWER the IP port in the bandwidth request will be that of the SBC because the SBC is managing the media for the session. If it is an OFFER, the signaling application inserts an empty IP port into the bandwidth request and sets a flag in the bandwidth request indicating unqualified flow information at this time. This occurs regardless of the value of the parameter **reserve-incomplete** in the **ext-policy-server** configuration element.

To enable this behavior on the Rx interface, the new parameter **media-release** has been added to the **ext-policy-server** configuration element.

Load Balancing Rx Servers

The Oracle Communications Session Border Controller (SBC) allows you to configure load balancing for DIAMETER Rx traffic across multiple Diameter Routing Agents (DRAs) using the **external-policy-server** configuration. When configured for TCP transport, this load balancing is available in addition to standard, DNS-based redundancy, where the SBC uses fully qualified domain names (FQDNs) to cycle through the multiple DRAs that DNS resolves to a single FQDN. For SCTP transport, the SBC simply substitutes the first address provided by a DNS lookup as the DRA connection address, and only uses **policy-group**s for load balancing.

The SBC performs load balancing via configured **policy-group**s, within which you configure the group's **policy-agent**s. You apply the **name** of a group to an **external-policy-server** element configured for Rx operations to complete the configuration.

You configure the load balancing strategy and recursion preferences within the **policy-group**, in addition to the agent list.

You configure each **policy-agent** for either TCP or SCTP transport, as well as the agent's address, port and realm. To utilize DNS, configure **address** parameter with FQDNs.

The SBC load balances between DRAs that are IN-SERVICE. The SBC marks a diameter agent as IN-SERVICE state if the transport socket is connected and the CER/CEA exchange is successful. The SBC does not send DIAMETER messages (other than CER) to a server that is not in the IN-SERVICE state. The SBC excludes a DRA from load balancing if it goes OUT-OF-SERVICE and includes it when it becomes IN-SERVICE again.

When configured, the SBC load balances all Rx traffic requests, with the exception of the following message types, which must be sent to each DRA individually:

- CER
- DWA and DWR
  - The SBC sends DWA answers to the DRA from which the DWR came.
  - The SBC sends DWRs to each DRA at the intervals configured by its **watchdog-ka-timer**.

The SBC must also send responses to the following requests directly to the DRA that issues them, and therefore does not load balance this traffic:

- DWR

- RAR
- ASR

This feature is RTC supported.

Rx Load Balancing Statistic Reporting

You can find statistics on load balanced RX traffic using:

- The **show policy-server** command displays cumulative counters for a group.
- The **show policy-server NAME/AGENT**" command displays specific group member counters.

Recursion

Recursion is de-coupled from load balancing. The SBC does not load balance recursive messages, sending them to the policy agents sequentially. Furthermore, the SBC performs recursion only on AAR and STR messages, not on CER and DWR messages.

> **Note:**
>
> Recursion, when enabled, takes precedence over the **str-retry** option.

Configuring Policy Groups and Policy Agents

To configure policy groups and policy agents on the Oracle Communications Session Border Controller (SBC):

1. Access the **policy-group** configuration

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# policy-group
ORACLE(policy-group)#
```

2. **group-name**—Enter a unique name for the policy group in Name format.
3. **description**—Optional. Enter descriptive information about the policy group.
4. **state**—Enable or disable the policy group. The default value is **enabled**. Valid values are:
   - enabled | disabled
5. **policy-agent**—Enter the policy-agent sub-element to configure 1 or more **policy-agent**s.
6. **policy-agent > name**—Enter a unique name for the **policy-agent** in Name format.
7. **policy-agent > description**—Optional. Enter descriptive information about the **policy-agent**.
8. **policy-agent > state**—Enable or disable the **policy-agent**. The default value is **enabled**. Valid values are:
   - enabled | disabled
9. **policy-agent > address**—Enter the address for the **policy-agent**. This address can be IPv4, IPv6 or an FQDN.
10. **policy-agent > port**—Enter the port for the **policy-agent**. The system ignores this parameter if the address parameter is set to an FQDN.

11. **policy-agent > realm**—Enter the realm for the **policy-agent**.
12. **policy-agent > watch-dog-ka-timer**—Enter the number of seconds to specify the expiry of this agent's watch dog timer.
13. **policy-agent > transport-protocol**—Enter the transport protocol for the **policy-agent**. The default value is **TCP**. Valid values are:
    - TCP (Default)
    - SCTP—SCTP is only valid over the Rx interface. This is confirmed by the **external-policy-server** configuration to which this **policy-group** is assigned.
14. **policy-agent > local-multi-home-addrs**—Applies to SCTP. Enter an IP address that is local to the SBC and can be used by this **policy-agent** as an alternate connection point. This address must by the same type, either IPv4 of IPv6 as that of the address parameter.
15. **policy-agent > remote-multi-home-addrs**—Applies to SCTP. Enter an IP addresses that can be used by this SBC as an alternate connection point. This address must by the same type, either IPv4 of IPv6 as that of the address parameter.
16. **policy-agent > sctp-send-mode**—Applies to SCTP. Specifies the SCTP delivery mode. The default value is **ordered**. Valid values are:
    - ordered | unordered
17. **strategy**—Enter the policy allocation strategy you want to use. The strategy you choose defines the order the SBC uses to try **policy-agent**s. The default and only value is **RoundRobin**:
    - **RoundRobin**—Selects each **policy-agent** in the order in which they are listed in the destination list, selecting each **policy-agent** in turn, one per session.
18. **max-recursions**—Enter an integer to specify the number of times the SBC can recurse through the agent list.
19. **stop-recurse**—Enter the list of SIP response codes that terminate recursion within the group. Upon receiving one of the specified response codes, such as 401 unauthorized, or upon generating one of the specified response codes internally, such as 408 timeout, the SBC returns a final diameter response code to the **policy-agent**s in the group and stops trying to route the message.

    Enter the response codes as a comma-separated list or as response code ranges.
20. **recursion-timeout**—Time in seconds that the SBC waits for max-recursions to finish before timing out. The default is 15 seconds.
21. Type **done** to save your configuration.

Assigning a Policy Group to a Policy Server

You can configure the SBC to load balance Rx interface traffic using **policy-group** configurations. The **ext-policy-server** element allows you to set its **address** parameter to a **policy-group**'s **name** parameter to use a group's load balancing configuration. You also configure the **policy-group**'s **policy-agent** sub-elements to define the external servers with which you load balance this traffic.

Example syntax for targeting a **policy-group** with the **ext-policy-server address** parameter:

```
ext-policy-server
        name                                    policyGroup1
        state                                   enabled
        operation-type                          bandwidth-mgmt
```

```
        protocol                                DIAMETER
        address                                 PAG:myGroupName
...
```

As with all Rx traffic, you can configure the SBC to support this traffic over SCTP as an alternative to TCP.

IPv6 Support

The Oracle Communications Session Border Controller supports Diameter-based CLF and RACF external policy servers in both IPv4 and IPv6 networks. There are three areas of enhanced functionality where the Oracle Communications Session Border Controller's Diameter external policy server offerings have changed.

- AVP support of IPv6 addresses encoded in UTF-8 format
- Extra bandwidth allocation in policed flows to compensate for longer addresses
- Framed-IPv6-Prefix AVP (97) support

IPv6 Addresses in UTF-8 Format

When necessary, the Oracle Communications Session Border Controllerchecks that IPv6 addresses are formatted correctly in UTF-8 when they are inserted into relevant AVPs. The applicable AVPs are

- Flow-Description AVP (507)
- Subscription-Id-Data AVP (444)
- Destination-Realm AVP (283) only if applicable

Framed-IPv6-Prefix AVP

The Oracle Communications Session Border Controller supports the Framed-IPv6-Prefix AVP (97) as defined in RFC4005.

The Diameter interface on the Oracle Communications Session Border Controller substitutes this AVP for the Framed-IP-Address AVP (8) in a Diameter message when carrying IPv6 addresses in the AVP.

The IP Address is encoded as Octet-String. Although the IPv6 address is a 128 bit number, it will not fall on a 4 byte boundary due to the formatting of this particular AVP. Additional whitespace will be added per RFC 3588 to the end of the octet-string to pad the ending of the AVP.

Bandwidth Allocation Compensation for IPv6

Transporting IPv6 packets requires extra bandwidth because of their larger packet size. This needs to be taken into account when allocating bandwidth and policing media. In order to reserve the necessary bandwidth for signaling messages, the **standard pkt rate** parameter has been added to the **media profile** configuration element.

The Oracle Communications Session Border Controller needs a baseline media packet size for bandwidth requests. It first checks if the SDP includes a **ptime** value. That value will be used as the baseline if present in the request. If this value is not included in the SDP, the **standard**

pkt rate parameter is used as the baseline. The chosen value is then multiplied times 20 (the difference between an IPv4 and IPv6 packet).

This value is sent to the external policy server in the bandwidth request when sending media into an IPv6 realm. From that point it is used when allocating bandwidth or media policing.

Bandwidth Requests while Transcoding

The Oracle Communications Session Border Controller can evaluate transcoding and IPv4 to IPv6 call scenarios and mitigate between end stations and policy servers to request appropriate bandwidth.

Two behaviors better assure proper bandwidth requests:

- AAR bandwidth requests for transcoded calls—The Oracle Communications Session Border Controller uses the SDP sent or received on the same realm as the external policy server to calculate the bandwidth information it presents in an AAR. This ensures that the Oracle Communications Session Border Controller receives bandwidth allocation information that is applicable to the leg that the policy server is serving.
- AAR bandwidth requests for dual stack calls—If the Oracle Communications Session Border Controller receives a call that switches between IPv4 and IPv6, the bandwidth that it presents in the AAR is based on the address space used by the call on the same realm as the external policy server. This ensures that the Oracle Communications Session Border Controller receives bandwidth allocation information that is applicable to the leg that the policy server is serving.

Diameter: RACF

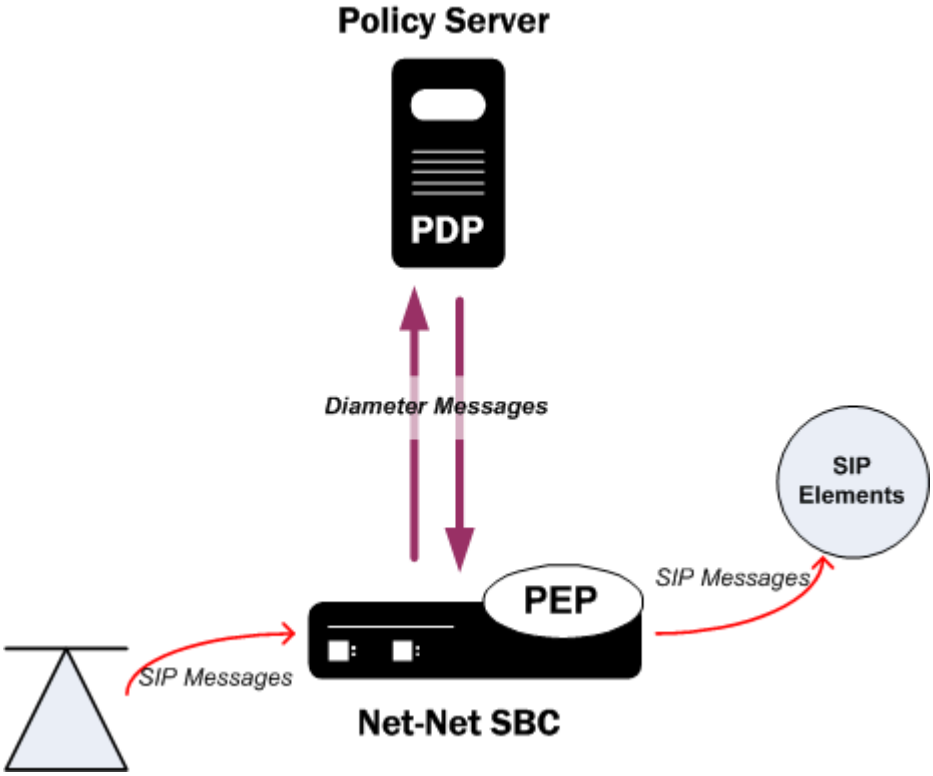
As the Oracle Communications Session Border Controller proxies and forwards calls, caller media information is known before the INVITE reaches the callee. The Oracle Communications Session Border Controller, acting as a P-CSCF, requests a specific amount of bandwidth for a call, and the RACF can reserve this amount of bandwidth for a call before the called phone rings. A call's required bandwidth can also be reserved by network devices along the path from the caller to callee if the QoS admission criteria is pushed from the RACF to other edge nodes (PEPs) such as routers, along this path to the callee.

Implementation Features

Bandwidth-based CAC is performed according to the following typical scenario. When the Oracle Communications Session Border Controller, known as the Policy Enforcement Point (PEP), receives a SIP INVITE, it sends a Diameter Authentication Authorization Request (AAR) message to the Policy Decision Point (PDP) or Resource and Admission Control Function (RACF). The Oracle Communications Session Border Controller does not forward the INVITE to its destination at this point.

The AAR message includes call identification information and the SDP-based bandwidth requirements for the call. The RACF responds with a Diameter Authentication Authorization Answer (AAA) message to either the install or remove the call. An install command directs the Oracle Communications Session Border Controller to forward the INVITE to the next SIP device. A remove command directs the Oracle Communications Session Border Controller send a SIP 503 Service Unavailable message sent back to the UA and reject the call.

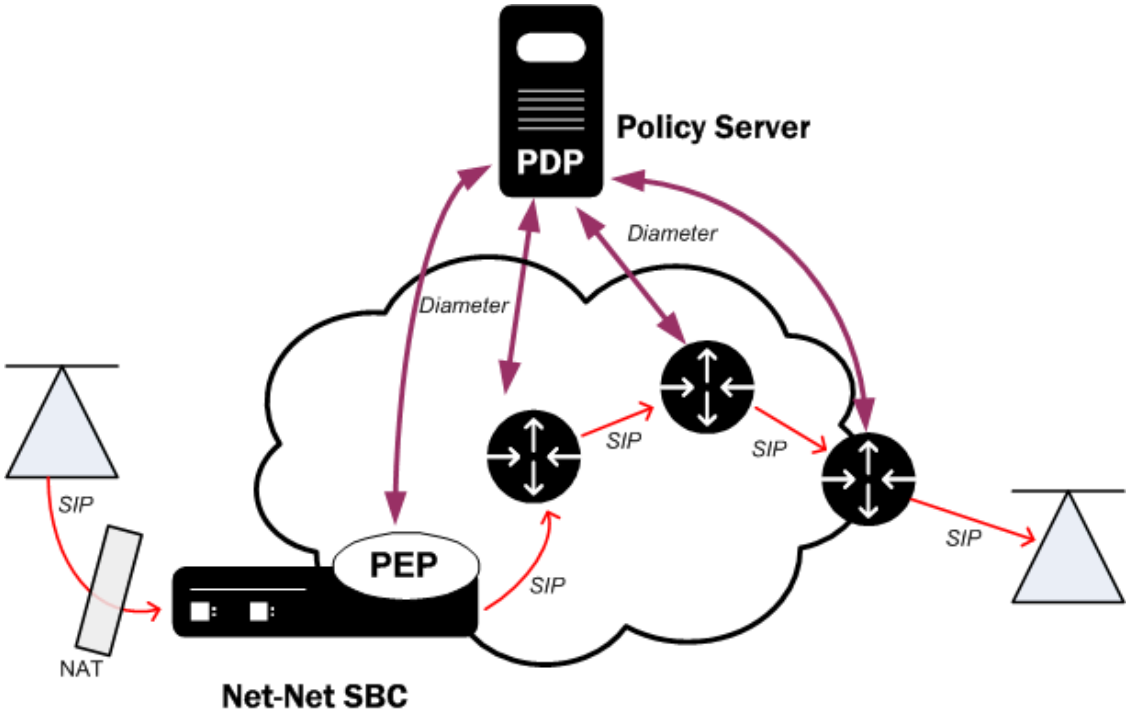
When the RACF is unreachable, incoming calls are rejected by default with a 503 message, as bandwidth can not be reserved. It is possible to configure the Oracle Communications Session Border Controller to allow all calls when the RACF is unreachable if this is the desired behavior.



The Oracle Communications Session Border Controller can be configured so that both sides of a call, based on realm, are subject to bandwidth enforcement. Each flow is treated as a unique call/event, because from a media and signaling perspective, they are distinct. As the Oracle Communications Session Border Controller functions as one side of a call, its IP address is inserted into the AAR message regardless of whether it is the calling or called party. This allows for the Diameter install or remove decision to be made before the Oracle Communications Session Border Controller receives the 200 OK response, and before ringing the far-end phone. Only one external policy server can be used within a single realm.

When a call ends, either with the expected SIP BYE or CANCEL methods, or due to other error conditions, the Oracle Communications Session Border Controller alerts the RACF by sending it a Diameter Session Termination Request (STR) message. All ended calls must be deleted from the RACF in order to accurately track used and available bandwidth.

The RACF can apply its hosted policies for calls originating at SIP UAs located behind NATs. This is a standard part of the Oracle Communications Session Border Controller's ability to provide seamless HNT.



Bandwidth Negotiation

Because the decision whether to admit or reject a call is made before the INVITE is forwarded to the called party, some information is not available to the PDP at the initial request. The final IP Address, UDP port number, that transport the RTP flow, and the codec used are not known by the Oracle Communications Session Border Controller until the called party responds with its own SDP information (either in the 180 or 200 response).

The Oracle Communications Session Border Controller examines the Session Description Protocol (SDP) value in the body of the SIP INVITE to determine what codecs are available for the call. If the INVITE specifies more than one codec, the Oracle Communications Session Border Controller bases its request to the RACF on the most bandwidth-hungry codec to ensure that all bandwidth requests will succeed or fail on the first attempt.

> **Note:**
>
> The amount of bandwidth requested depends on the configured media profiles.

If the call is admitted, and when the called party returns finalized SDP information, the Oracle Communications Session Border Controller modifies the original reservation with the chosen codec's bandwidth requirements. This ensures the RACF has current and accurate information with which to make policy decisions.

Session Lifetime

When receiving a successful Diameter response message for bandwidth from the RACF, a session lifetime timer may be included in the message. If included, this timer states how long the session can last. If the session continues past 3/4 of session lifetime, the Oracle Communications Session Border Controller sends another bandwidth request for that session to ultimately refresh the lifetime timer. If the RACF grants this bandwidth request, the Oracle

Communications Session Border Controller continues to allow the session to proceed uninterrupted. If a lifetime timer for a session is not returned to the Oracle Communications Session Border Controller by the RACF, the Oracle Communications Session Border Controller assumes the session can last forever and never issues a refresh in this manner.

Opening for RTCP Flows

When the Oracle Communications Session Border Controller functions as the AF (i.e., A-SBC or P-CSCF), you can configure it to explicitly open RTCP ports, just as it does for RTP. Without explicitly opening these ports, the Oracle Communications Session Border Controller relies on possibly unreliable PDN-GWs and BRAs to open RTCP ports and it sends only RTP information in AARs.

In external bandwidth managements configurations where the application mode is Rx and the **include-rtcp-in-request** parameters is **enabled**, the Oracle Communications Session Border Controller ensures RTCP ports are opened. It sends AAR requests to the policy server (PCRF) that contain both RTP and RTCP, opening the gates (ports) for both RTP and RTCP flows. RTCP information in the AAR is the number of the RTP port plus one (RTP port + 1 = RTCP ports) for all sessions. Flow information for RTCP is part of a different Media-Sub-component AVP as the RTP, but under the same Media-Component-Description AVP. RTCP flow information will also include the Flow-Usage AVP with RTCP (1). The RTCP port is set to 0 when the RTP ports is also unknown and therefore set to 0.

RACF-only AVPs

Diameter AAR Query Post SDP Exchange

The Oracle Communications Session Border Controller supports sending the Authentication-Authorize-Request (AAR) query upon SDP answer instead of the SDP offer. This change can useful in WiMax environments where mobile phones go idle and become semi-detached from their base stations and from the WiMax access controller (WAC). In such a case, the WAC receives an AAR from the idle user but, because it cannot determine that user's base station, rejects the request.

You enable this behavior by setting the **reserve-incomplete** parameter to **orig-realm-only**.

The Proxy Bit

When a signaling protocol receives an event request, the Oracle Communications Session Border Controller must ensure that the external policy server on the other end has enough bandwidth to maintain the requested call. The SDP information from the signaling message is stripped and encoded into the Diameter Band Request to be forwarded onto the external policy server. This feature is used with the Gq and other Diameter based interfaces.

The proxy bit allows the Oracle Communications Session Border Controller to tell the external policy server whether it wants the main server to handle the Diameter message, or if it is okay to proxy it to another server on the network. A parameter in the **ext-policy-server** configuration element called **allow-srv-proxy** has been developed. When this parameter is enabled, the proxy bit is set and the external policy server must process this Diameter request. When the parameter is disabled, the Oracle Communications Session Border Controller gives the external policy server permission to proxy the request along.

If you do not use this feature, this external policy server either handles the Diameter message on its own or proxies it to another server, depending on how much traffic it is handling at the

time. This is done without any input from the Oracle Communications Session Border Controller.

Experimental-Result-Code AVP: RACF

The Diameter RACF interface takes special actions based upon what AVPs are present inside the Experimental-Result AVP in the User-Data-Answer (UDA) message. If the Experimental-Result-Code AVP found within an Experimental-Result AVP has any value that is not considered successful, per RFC 3588, the response will be handled as non-successful response and a 503 error code will be issued back to the endpoint. If the value is a successful one, the normal call flow will proceed.

Transport-Class AVP

When the Oracle Communications Session Border Controller, running as a Diameter-based RACF, receives a SIP INVITE triggering external bandwidth management, the Oracle Communications Session Border Controller performs SDP stripping and—through internal processes—selects an external bandwidth manager to use. If the options parameter in the selected external bandwidth manager is set to transport-class, the Oracle Communications Session Border Controller's Diameter RACF interface will issue authentication authorization requests (AARs) with the transport class AVP. The Oracle Communications Session Border Controller does not insert the transport-class AVP messages when the option is not configured.

The transport-class AVP will:

- Be identified with the AVP code 311
- Always have the vendor (V) bit set in the AVP flags
- Never have the mandatory (M) bit set in the AVP flags
- Have the Vendor-Id field set to 13019 (a value specified by ETSI TISPAN)
- Be formatted as an unsigned integer
- Reside in the Media-Component AVP, a grouped AVP

In addition, the transport-class AVP's payload field will be a 32-bit unsigned integer corresponding to a specific media type. The Oracle Communications Session Border Controller learns the specific media type from the m-line of the SDP it received. The following table shows how the Oracle Communications Session Border Controller evaluates the SDP's m= lines and concludes a default service type classification.

Service Class Classification	SDP evaluation	Default transport-class value
video	At least 1 m=video line	1
audio	No m= video At least 1m=audio	0
application	No m=video, No m=audio At least 1 m=application	3
data	No m=video, No m=audio, No m=application At least 1 m=data	2
control	No m=video, No m=audio, No m=application, No m=data At least 1 m=control	4

Service Class Classification	SDP evaluation	Default transport-class value
image	No m=video, No m=audio, No m=application, No m=data, No m=control At least 1 m=image	10
text	No m=video, No m=audio, No m=application, No m=data, No m=control, No m=image At least 1 m=text	5
message	No m=video, No m=audio, No m=application, No m=data, No m=control, No m= image, No m=text At least 1 m=message	6

After the service class classification has been chosen, the Oracle Communications Session Border Controller inserts the corresponding default transport class value in the transport-class AVP to send to the RACF.

Overriding Transport- Class AVP Value

You can override the Transport class AVP value sent to the RACF in a Diameter message by configuring the **service class options** parameter. Custom service class option values are configured as a <service-class-option>=<user-entered-value> value pair. For example, to configure the Oracle Communications Session Border Controller to send sending the value 80 instead of the value 8 for a message service class classification, you would configure **message=80** in the service class options parameter.

Transport-class AVP Configuration

You configure the Oracle Communications Session Border Controller to send the Transport-Class AVP in the external bandwidth manager's **options** parameter.

To set the transport-class AVP support for an external bandwidth manager:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **ext-policy-server** and press Enter.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. **options**—Set the options parameter by typing options, a Space, the option name **transport-class** with a plus sign in front of it. Then press Enter.

```
ORACLE(ext-policy-server)# options +transport-class
```

If you type **options** and then the option value without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. **service-class-options**—Set this options parameter by typing **service-class-options**, a <space>, the service-class type, an equal sign, your custom value. Then press Enter. Enter multiple service class/value pairs separated by a comma. For example:

```
ORACLE(ext-policy-server)# service-class-options message=80,text=70
```

6. Save and activate your configuration.

Service-Info-Status AVP

When the Oracle Communications Session Border Controller (SBC), running as a Diameter-based RACF, receives a SIP INVITE triggering external bandwidth management, the SBC uses the **Service-Info-Status** AVP to indicate the status of the service information that the SBC provides to the external policy server. If the **Service-Info-Status** AVP is not provided in the AA request, the policy servers assumes the service information is final.

The initial context for the SBC's support of this AVP is to support the reservation of resources over diameter, supporting call flows for Mobile Originating (MO), Mobile Terminating (MT) and non-mobile scenarios:

- Invite with SDP offer payload (MO)
- UE sends invite without offer, and an SDP answer is sent in PRACK (MO)
- 183 Response with SDP answer payload from UE (MT)
- UE sending SDP offer payload in 183 response (MT)
- UE in origination realm sends re-invite, SDP is renegotiated with AAR
- UE in origination realm sends update resulting in SDP exchange
- Scenarios where SDP is inserted by SBC

When triggered, the SBC sends AAR messages to the external policy server over Diameter Rx interface. The supported protocol specification is Policy and Charging Control over Rx reference point 3GPP TS 29.214. The **Service-Info-Status** AVP (AVP code 527) carries an enumerated value, as follows:

- **PRELIMINARY SERVICE INFORMATION** (1) —This value indicates that the service information provided to the external policy server is preliminary and needs to be further negotiated between the two endpoints.
- **FINAL SERVICE INFORMATION** (0) — This value indicates that the service has been fully negotiated between the two endpoints. The service information provided is the result of that negotiation.

> **Note:**
>
> Using the **Service-Info-Status** to discriminate between the status of service information is limited to cases where **optimize-AAR** is disabled and **reserve-incomplete= original-realm-only**.

Rx-Request-Type AVP

When the Oracle Communications Session Border Controller (SBC), running as a Diameter-based RACF, supports external bandwidth management, the SBC uses this AVP to specify the status of the request.

The SBC can use this AVP in conjunction with the service-info-status AVP to specify whether the referenced SDP is fully negotiated and will be used for the call.

When triggered, the SBC sends AAR messages to the external policy server over Diameter Rx interface. The supported protocol specification is Policy and Charging Control over Rx reference point 3GPP TS 29.214.

The **Rx-Request-Type** AVP (AVP code 533) carries an enumerated value, as follows:

- **INITIAL_REQUEST** (0) — The SBC is initiating the Rx session for the first time.
- **UPDATE_REQUEST** (1) — The SBC is updating information related to SDP parameters of an existing Rx session.

> **Note:**
>
> The value **PCSCF_RESTORATION** is not supported within this AVP.

SIP-Forking-Indication AVP (523)

When handling access VoLTE sessions with multiple early dialogs, the Oracle Communications Session Border Controller (SBC), acting as A-SBC or P-CSCF, includes the **SIP-Forking-Indication** AVP in the Rx request sent to the PCRF. This occurs when the SBC receives several responses (provisional or not) with different To-Tag identifiers and different SDP.

The **SIP-Forking-Indication** AVP (AVP code 523) is of type Enumerated, and indicates whether or not multiple SIP dialogs are related to one Diameter session:

- **SINGLE_DIALOGUE** (0) - This value indicates that the Diameter session relates to a single SIP dialog. This is the default value applicable if the AVP is omitted.
- **SEVERAL_DIALOGUES** (1) - This value indicates that the Diameter session relates to several SIP dialog.

Considering the usual Access VoLTE call flow at the A-SBC level, the SBC behaves as follows:

- When receiving a first 183 to establish a first early dialog (CALL_ID_1, From_Tag_1 and To_Tag_1), the SBC does not include the **SIP-Forking-Indication** AVP in the Rx request it sends to the PCRF
- When receiving subsequent 183 to establish another early dialog (CALL_ID_1, From_Tag_1 and To_Tag_N), the SBC includes the **SIP-Forking-Indication** AVP with the value SEVERAL_DIALOGUES in the Rx request it sends to the PCRF.
- On receiving the first final SIP response, the SBC sends the AAR without the **SIP-Forking-Indication** AVP and includes the service information derived from the SDP corresponding to the dialogue of the final response.
- In a multi-dialog scenario, the SBC sends the **SIP-Forking-Indication** AVP with the value **SEVERAL_DIALOGUES** in any PRACK or UPDATE requests that are involved in subsequent SDP exchanges that target further service provisioning.

RACF and CLF AVPs

Frame-IP-Address AVP

The Diameter CLF and RACF interfaces can send a Frame-IP-Address AVP. You can configure the value to appear in either an ascii string (e.g., 192.168.10.1) or an octet string (e.g.,0xC0A80A01) with the **framed ip addr encoding** parameter.

1637 - Diameter Destination Realm AVP

As of S-C6.2.0, the Destination Realm AVP's value does not contain the realm of the incoming SIP message. Now, it contains the realm where the Policy Server resides as learned from the Origin-Realm AVP received in a CEA message from the Policy Server.

The Oracle Communications Session Border Controller can be configured with an option, **include-gua** , to retain the previous behavior of sending an incoming SIP message's realm to a policy server. This is accomplished by sending the Globally Unique AVP in the AAR message to the policy server, by adding an option parameter to the external policy server configuration.

The following table summarizes the effect of provisioning the external policy server with the Globally Unique AVP option on each Diameter interface, as configured.

Diameter Interface	No include-gua option Configured (default)	Add include-gua option
Rq	AAR sends Globally Unique Address AVP	AAR sends Globally Unique Address AVP
Rx	AAR does not send Globally Unique Address AVP	AAR will contain Globally Unique Address AVP
Gq	AAR does not send Globally Unique Address AVP	AAR will contain Globally Unique Address AVP
E2	AAR sends Globally Unique Address AVP	AAR sends Globally Unique Address AVP

Legacy Destination-Realm AVP Behavior

The Diameter CLF and RACF interfaces can change the format of the payload string in the Destination-Realm AVP for any Diameter message it originates and sends to an external server. The payload field for this AVP can be constructed in any the following formats:

Format	Description
<user>@<realm>	<user>—IP address of the endpoint initiating the call with the Oracle Communications Session Border Controller <realm>—Name of the realm on which the Oracle Communications Session Border Controllerreceived the INVITE from a user
<user>	<user>—IP address of the endpoint initiating the call with the Oracle Communications Session Border Controller
<realm>	<realm>—Name of the realm on which the Oracle Communications Session Border Controller received the INVITE from a user

When either the Diameter CLF or RACF interface sends any message with the Destination-Realm AVP, it determines from the external policy server configuration how to construct the payload string for this AVP.

You can set the format to use in the **dest-realm-format** parameter in the external policy server configuration. The parameter can be set to any value in the table above and defaults to <user>@<realm>. By treating the format this way, the policy server and the Oracle Communications Session Border Controller can easily communicate this value; if sent to the policy server in any AVP, the policy server can simply return the full value.

Origin-Host AVP

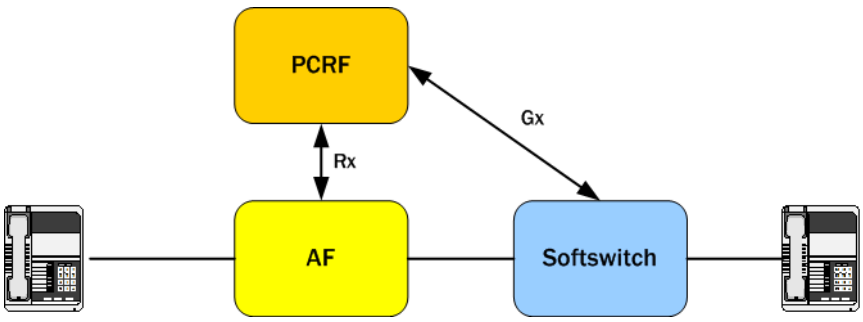
The Diameter CLF and RACF interfaces can change the suffix for Origin-Realm and Origin-Host AVPs that have a payload string constructed as a domain name.

You can set the suffix you want appended to payload strings using the **domain-name-suffix** parameter in the external policy server configuration. This parameter can be set to any string (default is .com), and the Oracle Communications Session Border Controller automatically adds a dot (.) to the front of this entry if you do not include one. The policy server and the Oracle Communications Session Border Controllercan easily communicate this value; if sent to the policy server in any AVP, the policy server can simply return the full value.

Wildcard Transport Protocol

The Oracle Communications Session Border Controller external bandwidth management solution provides for an Rx interface that supports the Flow-Description AVP (507). Rather than use a numerical value, this Flow-Description AVP uses an IP filter rule with the keyword “ip.” The ip keyword means any transport protocol matches the Flow-Description AVP when issuing AARs to the PCRF. Before it forwards a Gx RAR messages to the softswitch, the PCRF decodes the audio codec into the correct speed and classification. The PCRF passes the Oracle Communications Session Border Controllers Flow-Description AVP to the softswitch untouched. But not all softswitches accommodate the ip keyword, resulting in rejected requests.

Figure 17-1 Interfaces Supporting Diameter Bandwidth Management



When you enable the **wildcard-transport-protocol** parameter, however, you can essentially format the Flow-Description AVP to suit your network.

For sessions that need to allocate media and have applicable external bandwidth management associations, the Oracle Communications Session Border Controller’s Diameter interface checks for the necessary bandwidth. The Diameter interface, with an Rx or Rq application mode, constructs an AAR with Flow-Description AVP of ip when the **wildcard-transport-protocol** parameter is **enabled**. The flow description would look like this:

```
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit out ip from 168.192.24.20 49500 to 168.192.24.0 8000
```

```
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit in ip from 168.192.24.0 8000 to 168.192.24.20 49500
```

With the **wildcard-transport-protocol** set to **disabled**, the Oracle Communications Session Border Controller does not use the ip wildcard keyword. Instead, it uses the specific media stream transport protocol in the Flow-Description AVP—and only falls by to the ip keyword when the transport protocol is unknown. The flow description with this parameter disabled would look like this:

```
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit out 17 from 168.192.24.20 49500 to 168.192.24.0 8000
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit in 17 from 168.192.24.0 8000 to 168.192.24.20 49500
```

New Configurations and Upgrading

To comply with 2GPP TS 29.213, **wildcard-transport-protocol** parameter is **disabled** by default in new configurations. So if the transport protocol is known, the Oracle Communications Session Border Controller uses it in the Flow-Description AVP.

To maintain default behavior for existing configurations, the Oracle Communications Session Border Controller performs a check at the time of upgrade to set this parameter to **enabled**. This setting means the Oracle Communications Session Border Controller does use the ip keyword in the Flow-Description AVP.

Configuring Diameter-based RACF

In the following configuration examples, we assume that your baseline configuration passes SIP traffic, with the Oracle Communications Session Border Controller (SBC) in the role of an Access SBC. In this example, you perform realm configuration and external bandwidth manager configuration. You also configure media profiles with the bandwidth parameters you reserve at the RACF.

Diameter Support Realm Configuration

To configure the realm configuration for Diameter support in a CAC scenario:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure.

```
ORACLE(realm-config)# select 1
ORACLE(realm-config)#
```

5. **mm-in-realm**—Set this parameter to enabled so that calls from devices in the same realm have their media flow through the Oracle Communications Session Border Controller to be subject to CAC. The default value is **disabled**. The valid values are:
   - enabled | disabled
6. **mm-in-network**—Set this parameter to **enabled** so that the Oracle Communications Session Border Controller will steer all media traveling between two endpoints located in different realms, but within the same network. If this field is set to disabled, then each endpoint will send its media directly to the other endpoint located in a different realm, but within the same network. The default value is **enabled**. The valid values are:
   - enabled | disabled
7. **ext-bw-manager**—Enter the name of the external bandwidth manager configuration instance to be used for external CAC for this Realm.
8. Save your work using the ACLI **done** command.

External Bandwidth Manager Configuration

To configure the external bandwidth manager:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the media-related configurations.

```
ORACLE(configure)# media-manager
```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. **name**—Enter the name for this external bandwidth manager instance. This parameter is used to identify the PDP that will be used in each Realm configuration.
5. **state**—Set the state of this ext-policy-server configuration to **enabled** to run this CAC. The default value is **enabled**. The valid values are:
   - enabled | disabled
6. **operation-type**—Enter **bandwidth-mgmt** for this external policy server configuration element to perform RACF/External Policy Server functions. The default value is **disabled**. The valid values are:
7. **protocol**—Enter **Diameter** to support Diameter-based CAC. The default value is **C-SOAP**.
8. **address**—Enter the IP address or FQDN of an external policy server, or enter the name of a policy-group preceded by the **PAG:** prefix. IP addresses can by IPv4 or IPv6.

9. **port**—Enter the port number the diameter connection connects to on the RACF. The system ignores this parameter if the address parameter is set to a policy-group or an FQDN. The valid range is:
   - Minimum—0
   - Maximum—65535
10. **realm**—Enter the name of the Realm in which this Oracle Communications Session Border Controller defines the RACF to exist. This is NOT necessarily the Realm where the Oracle Communications Session Border Controller performs admission control. The system ignores this parameter if the address parameter is set to a policy-group, with the exception that it is used to populate all Origin-Realm and Origin-Host AVPs in diameter messages generated by traffic from the **policy-group**'s **policy-agent**s.
11. **transport-protocol**—Enter the transport protocol used to connect to this external policy server.
    - TCP (Default)
    - SCTP
12. **local-multi-home-addrs**—Applies to SCTP. Enter an IP address that is local to the SBC and can be used by this external policy server as an alternate connection point. This address must be the same type as the address parameter, either IPv4 or IPv6
13. **remote-multi-home-addrs**—Applies to SCTP. Enter an IP address that can be used by this SBC as an alternate connection point. This address must be the same type as the address parameter, either IPv4 or IPv6.
14. **sctp-send-mode**—Applies to SCTP. Specifies the SCTP delivery mode..
    - ordered (Default)
    - unordered
15. **permit-conn-down**—Enter **enabled** if this external policy server configuration can permit new calls into the network when the policy server connection is down. The default value is **disabled**. The valid values are:
16. **product-name**—Enter text string that describes the vendor-assigned name for the RACF. This parameter is required.
17. **application-mode**—Enter the type of interface you want to use. Your choices are: **Rq**, **Rx**, **Gq**, **e2**, and **none**.
18. **application-id**—Enter a numeric application ID that describes the interface used to communicate with the RACF. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
19. **framed-ip-addr-encoding**—Enter the format of the Frame-IP-Address (AVP 8) value in Diameter messages. The default value is **octet-string**. The valid values are:
    - **ascii-string**—Example: 192.168.10.1
    - **octet-string**—Example: 0xC0A80A01
20. **dest-realm-format**—Enter the format you want to use for the Destination-Realm AVP. The default value is **user_with_realm**. The valid values are:
    - user_with_realm | user_only | realm_only
21. **domain-name-suffix**—Enter the suffix you want to use for Origin-Realm and Origin-Host AVPs that have a payload string constructed as a domain name Your value can be any

string, to which the system will prepend with a dot if you do not include one. The default value is **.com**.

22. **allow-srv-proxy**—Set to enabled in order to include the proxy bit in the header. The presence of the proxy bit allows the Oracle Communications Session Border Controller to tell the external policy server whether it wants the main server to handle the Diameter message, or if it is okay to proxy it to another server on the network (disabled). The default is enabled. The valid values are:
    - enabled | disabled
23. **wildcard-trans-protocol**—Set this parameter from **enabled** if you want to use transport protocol wildcarding for Rx/Rq Flow-Description AVP (507) generation. Enabled sends a flow description of ip. Set this parameter to **disabled** if you want to use the specific media stream transport protocol.
24. **reserve-incomplete**—Set this parameter to **enabled** when communicating with a PDP via Diameter. The parameter allows the Oracle Communications Session Border Controller to make admission requests before learning all the details of the flows and devices (e.g., not knowing the final UDP port numbers for the RTP media streams until after the RTP has begun). The default value is **enabled**. The valid values are:
    - **enabled** (default)—This mode supports the usual behavior when the AAR is sent upon SDP offer as well as SDP answer. This mode ensures backwards compatibility.
    - **orig-realm-only**—This mode allows calls originating from a realm with a policy server associated with it to send the AAR upon SDP offer. However, calls terminating at a realm with a policy server associated with it send the AAR post SDP exchange.
    - **disabled**—This mode allows no bandwidth reservation for incomplete flows.
25. **include-rtcp-in-request**—Change this parameter from **disabled** (default), to **enabled** so the Oracle Communications Session Border Controller will include RTCP information in AARs. RTCP information is the number of the RTP port plus one (RTP port + 1 = RTCP ports) for all sessions.
26. **trans-expires**—Set the amount of time, in seconds, that the Oracle Communications Session Border Controller waits before performing an expiration if a Diameter base protocol transaction does not occur. The default value is 15 seconds. Valid values range between 1 and 15.
27. Save your work using the ACLI **done** command.

Media Profile Configuration

Values for the following parameters can be found in the PacketCable™ Audio/Video Codecs Specification PKT-SP-CODEC-I06-050812 document.

To configure the media profile configuration for Diameter support in a CAC scenario:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session router path.

```
ORACLE(configure)# session-router
```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# media-profile
ORACLE(media-profile)#
```

4. Type **select** and the number of the pre-configured media profile you want to configure.

```
ORACLE(media-profile)# select 1
ORACLE(media-profile)#
```

5. **req-bandwidth**—Enter the required bandwidth in Kbps for the selected media profile. This is the bandwidth that the Oracle Communications Session Border Controller will request from the policy server. The default value is zero (**0**). The valid values are:
   - Minimum—0
   - Maximum—4294967295
6. **standard-pkt-rate**—Enter the value to use for the standard packet rate for this codec when sending a request to the RACF for a bandwidth request.
7. Save your work using the ACLI **done** command.

Additional Diameter Compliance for the Rx Interface

When handling some Register and Message flows, the SBC default behavior does not include strict compliance with Diameter session teardown rules. Typically, the environment can proceed without issue, but the SBC provides an **ext-policy-server** option, called **diam-rx-strict-compliance**, that provides better compliance with Diameter session teardown rules.

Set the **diam-rx-strict-compliance** option on the applicable **ext-policy-server** to establish the following Diameter session behavior:

- For Register flows that do not establish a Diameter session with the PCRF due to a 3xxx, 4xxx or 5xxx error from the PCRF, the SBC does not send an STR to tear down the session when it receives a De-Register.
- For Message flows, when the SBC receives an ASR from PCRF, it stops the hold timer if it is running, then forwards the MESSAGE to the core, and sends an ASA with success.
- For unsuccessful Register flows that include an established Diameter session with the PCRF, the SBC sends an STR to tear down the session after the Register has failed due to, for example, responses from the core.
- If the SBC receives an S8HR Emergency Registration, with or without Authorization header, and either the Rx interface is not available or there is an error in AAA response sent by PCRF, the SBC replies with a 5xx response. (ACMECSBC-37498)
- If the SBC receives an S8HR Emergency Registration without an Authorization header and the EPC identities validation fails, the SBC sends a 403 error with the SIP reason header. If the SBC receives a REGISTER request with the authorization header, it sends a MIME XML body with a reason tag. (ACMECSBC-37497)
- For S8HR registrations and calls, the SBC adds the P-Visited-Network-ID header using the format "plmnIdPrefix.mncxxx.mccxxx.3gppnetwork.org". (ACMECSBC-37431)
- During an S8HR registration scenario, if the SBC receives a REGISTER request with the Authorization header and the next-hop is not configured, the SBC sends a 403 response if the EPC identities validation fails. (ACMECSBC-37459)

- Within a register call flow wherein the SBC receives an AAA with a 3002 error code from the PCRF after the diameter transaction has timed out, the SBC does not send an STR to the PCRF.
- Within emergency REGISTER call flows when S8HR is enabled and there are no EPC level identities cached, the SBC does not issue an STR simultaneously with a 403 error code if it receives a 3002 error code from the PCRF.

Syntax for this option may or may not include the plus (+) sign, but note that setting the option with the + sign retains all other options set on the element. Omitting the + sign replaces any existing options with the one you set.

```
ACMEPACKET(ext-policy-server)# options +diam-rx-strict-compliance
```

Configuring the Rx Interface for SCTP

You configure the Rx interface for SCTP transport from the **ext-policy-server** element. See the SCTP Overview and configuration sections in this document's *System Configuration* chapter for information about SCTP operation and the global settings within the **network-parameters** element that apply to all SCTP operation.

1. From superuser mode, use the following command sequence to access sip-port configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

2. Set the **operation-type** to **bandwidth-mgmt** for this external policy server configuration element to perform RACF/External Policy Server functions.

   ```
   ORACLE(ext-policy-server)#  operation-type  bandwidth-mgmt
   ```

3. Set the **application-id** to **16777236** for the Rx interface. (See the guidelines in Application ID and Modes.)

   ```
   ORACLE(ext-policy-server)#  application-id 16777236
   ```

4. Set the **application-mode** to **Rx**. (See the guidelines in Application ID and Modes.)

   ```
   ORACLE(ext-policy-server)#  application-mode Rx
   ```

5. Use the **address** parameter to set the IPv4, IPv6 or FQDN address of the policy server.

   ```
   ORACLE(ext-policy-server)# address 172.16.10.76
   ```

6. Retain the default value, 5060 (the well-known SIP port) for the **port** parameter.

   ```
   ORACLE(ext-policy-server)# port 5060
   ```

7. Use the **transport-protocol** parameter to set the layer 4 protocol to SCTP.

   ```
   ORACLE(ext-policy-server)# transport-protocol sctp
   ```

8. Use the **remote-multi-homed-addrs** parameter to specify a remote secondary address of the SCTP endpoint.

   This address must be of the same type (IPv4 or IPv6) as that specified by the **address** parameter, unless the **address** is an FQDN.

```
ORACLE(ext-policy-server)#remote-multi-homed-addrs 182.16.10.76
```

9. Use the **local-multi-homed-addrs** parameter to specify a local secondary address of the SCTP endpoint.

   This address must be of the same type (IPv4 or IPv6) as that specified by the **address** parameter, unless the **address** is an FQDN. Like the address parameter, this address identifies an SBC network interfaces.

```
ORACLE(ext-policy-server)# local-multi-homed-addrs 162.16.10.76
```

10. Use **done**, **exit**, and **verify-config** to complete configuration of this SCTP-based SIP port.

```
ORACLE(ext-policy-server)# done
ORACLE(media-manager)# exit
ORACLE(configure)# exit
ORACLE# verify-config
----------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

CAC Debugging

A new argument has been added to the show command for viewing CAC statistics. From the user prompt, type **show ext-band-mgr**.

```
ORACLE# show ext-band-mgr
10:11:38-194
EBM Status                  -- Period -- -------- Lifetime --------
                  Active    High   Total       Total  PerMax    High
Client Trans           0       0       0           0       0       0
Server Trans           0       0       0           0       0       0
Sockets                1       1       1           1       1       1
Connections            0       0       0           0       0       0
                            ---- Lifetime ----
                    Recent      Total  PerMax
Reserve                  0          0       0
Modify                   0          0       0
Commit                   0          0       0
Remove                   0          0       0
EBM Requests             0          0       0
EBM Installs             0          0       0
EBM Errors               0          0       0
EBM Rejects              0          0       0
EBM Expires              0          0       0
EBMD Errors              0          0       0
```

Retrieve the CAC statistics in the log.embd file.

2127 - Configurable Subscription ID Types

Subscriber Information AVP

Certain policy servers rely on having the user's URI information available as means to identify the endpoint/subscriber. In addition to conveying the L3 IP address of a user, the Oracle Communications Session Border Controller supports RFC 4006: The Subscription-Id AVP. It identifies the end user's subscription and is used in 3GPP Rx reference point. This feature can be enabled regardless of the selected application mode of the external policy server.

You can enable the Oracle Communications Session Border Controller to include the Subscription-ID-Type based on the received message's contact header, or you can set the Subscription ID type on a per-realm basis.

Subscription-ID AVP

The Subscription-Id AVP (AVP Code 443) includes a Subscription-Id-Data AVP that holds the identifier and a Subscription-Id-Type AVP that defines the identifier type. The external policy server configuration element is configured with an option to enable sending the Subscription-Id AVP to the policy server in an AA-Request (AAR) message.

Subscription-Id-Type

The Oracle Communications Session Border Controller can send a Subscription-Id-Types AVP to an external policy server:

Value	Name	Description
0	END_USER_E164	Identifier is in international E.164 format (e.g., MSISDN), according to the ITU-T E.164 numbering plan.
1	END_USER_IMSI	Identifier is in international IMSI format, according to ITU-T E.212 numbering plan as defined in [E212] and [CE212].
2	END_USER_SIP_URI	Identifier is in the form of a SIP/SIPS URI.

To send the Subscription-ID-Types AVP in an AAR, you must add the include-sub-info option in the external policy server configuration element. If this option is enabled, and you do not configure a **subscription-id-type** in the realm-config, the contact header of the received message is sent in the AAR.

You can also set the Subscription-ID-Type on a per-realm basis by configuring the parameter **subscription-id-type** in the realm-config element to determine the value of the Subscription ID in the AAR.

The default value for **subscription-id-type** is END_USER_NONE. If the value of subscription-id-type is set to END_USER_NONE, and the external policy server option is enabled to send the Subscription ID AVP, then the Oracle Communications Session Border Controller relies on the contact header of the received message.

If the value for **subscription-id-type** is set to one format, such as END_USER_IMSI, but the contact header is set to another format, such as END_USER_SIP_URI, the value of **subscription-id-type** is used to send in the AAR message.

Subscription-Id-Data

When applicable, the SBC can send a Subscription-Id-Data AVP (444) to an external policy server. This AVP is contained within the grouped Subscription-Id AVP (443) and carries the user's identifier. You can configure the SBC to refine this data so it gets this information from the SBC and uses your configured value for the **subscription-id-type** parameter to determine which user identifier it sends.

You must enable the **include-sub-info** option in an applicable **ext-policy-server** to send the grouped Subscription-Id AVP, which includes the Subscription-Id-Data AVP, in an AAR towards the PCRF.

In addition to the **include-sub-info** option and the **subscription-id-type** parameter, you may need to enable the **preferred-sub-id-type** option in an **ext-policy-server** to establish all of the behavior you need for your deployment. Use of the **preferred-sub-id-type** option is only relevant to registered users.

When enabled, the **preferred-sub-id-type** option further refines the value of the Subscription-id-Data AVPs for registered users before the SBC sends the Subscription-id AVP towards the PCRF in the AAR. Specifically, when you enable this option, the SBC fetches the user identity from the its cache and determines which ID type it sends in the Subscription-id-Data AVP to be the same type presented in the Subscription-Id-Type (450) and specified by your **subscription-id-type** configuration in the AAR message.

For example, if you set **subscription-id-type** to END_USER_IMSI and enable **preferred-sub-id-type**, the SBC retrieves the IMSI value from the subscriber registration cache, formats it based on the and sends it as the Subscription-Id-data in the grouped AVP within the AAR.

The cumulative effects of these configurations are:

- Enabling the **include-sub-info** option causes the SBC to send the Subscription-ID-Types AVP in an AARs.
- Configuring the **subscription-id-type** parameter causes the SBC to include your value as the value of the Subscription-ID-Types AVP in AARs.
- Enabling the **preferred-sub-id-type** option causes the SBC to retrieve the user identify from its cache that is the same type you configured in the applicable **subscription-id-type**, thereby specifying and sending the correct value in the Subscription-Id-Data AVP.

Finally, if the SBC does not have the subscriber registration in its cache, it sets the Subscription-Id-Type and Subscription-Id-Data AVP values using the contact header or the Request URI in the request line of the received message.

Important caveats on the use of this feature include:

- The SBC only caches an IMSI when it is a 14 or 15 digit number.
- If the incoming data does not correspond to your **subscription-id-type** value, the SBC sets the Subscription-Id-Type/Data based on the incoming Contact URI/R-URI as a Sip-Uri or Tel-Uri.
- If you configure the **subscription-id-type** value to END_USER_NONE, the SBC sets the Subscription-Id-Type/Data based on the incoming Contact URI/R-URI as a Sip-Uri or Tel-Uri.
- The SBC uses this behavior when you enable the **preferred-sub-id-type** option in the **ext-policy-server** applied to either the ingress or the egress realm.
- The SBC sends subscriber data towards the PCRF based on the realm to which you have applied the **ext-policy-server**.

- If applied to the access realm, the SBC sends the calling (west) side subscriber data to the PCRF.
- If applied to the core realm, the SBC sends the called (east) side subscriber data to the PCRF.

Subscription ID AVP in AA-Request Message Configuration

To configure the Oracle Communications Session Border Controller to send Subscription ID AVP in AA-Request Messages:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **ext-policy-server** and press Enter.

```
ORACLE(session-router)# ext-policy-server
ORACLE(ext-policy-server)#
```

If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI select command) the external policy server that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **include-sub-info** with a plus sign in front of it, and then press Enter.

```
ORACLE(ext-policy-server)# options +include-sub-info
```

If you type the option without the plus sign, you overwrite any previously configured options. To append the new options to an element's options list, prepend the new option with a plus sign as shown in the example.

5. **options**—If needed, set the options parameter by typing **options**, a Space, the option-name **preferred-sub-id-type** with a plus sign in front of it, and then press Enter.

```
ORACLE(ext-policy-server)# options +preferred-sub-id-type
```

If you type the option without the plus sign, you overwrite any previously configured options. To append the new options to an element's options list, prepend the new option with a plus sign as shown in the example.

6. Save and activate your configuration.

Subscription ID AVPs in AA-Request Message Configuration

To set the Subscription ID AVPs in AA-Request Messages:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-config** and press Enter.

```
ORACLE(session-router)# realm-config
ORACLE(realm-config)#
```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI select command) the realm that you want to edit.

4. **subscription-id-type**—Set the Subscription ID Type included in the AA-Request.
   - END_USER_E164
   - END_USER_IMSI
   - END_USER_SIP_URI
   - END_USER_NONE
5. Save and activate your configuration.

Specific Action AVP support

When acting as a P-CSCF, Oracle Communications Session Border Controller sends the Specific-Action AVP to the PCRF in an AAR message to indicate the subscription types it supports.

The Oracle Communications Session Border Controller can be configured to subscribe to one or more of the following subscription types:

- LOSS OF BEARER
- RECOVERY OF BEARER
- RELEASE OF BEARER
- OUT OF CREDIT
- SUCCESSFUL RESOURCES ALLOCATION
- FAILED RESOURCES ALLOCATION

When no subscription types are configured, the Oracle Communications Session Border Controller does not include the Specific-Action AVP in its AAR.

Specific Action AVP subscription is configured in the **Specific action subscription** parameter located in the External policy server configuration element.

ACLI Examples - specific-action-subscription

1. Navigate to the ext-policy-server configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# ext-policy-server
ACMEPACKET(ext-policy-server)#
```

2. Select an existing external policy server configuration element.

```
ACMEPACKET(ext-policy-server)# select
name:
1: extpol1

selection: 1
ACMEPACKET(ext-policy-server)#
```

3. specific-action-subscription - Configure 1 or more specific actions. When configuring 2 or more specific actions, enclose them in quotation marks, with the values separated by spaces. The following are valid specific actions: loss-of-bearer, recovery-of-bearer, release-of-bearer, out-of-credit, successful-resources-allocation, failed-resources-allocation

```
ACMEPACKET(ext-policy-server)# specific-action-subscription "loss-of-
bearer recovery-of-bearer"
```

4. Type done to save your work.
5. Save and activate your configuration to begin using this feature.

Diameter STR Timeouts

When a call ends, the Oracle Communications Session Border Controller alerts the RACF by sending it a Diameter Session Termination Request (STR) message. You can enable the Oracle Communications Session Border Controller to resend STR messages at the application layer if the STR messages time out. A new option **STR-retry=x** has been created that allows you to configure the number of times the STR messages are resent.

Diameter STR Timeouts Configuration

To configure Diameter request timeouts:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. Set the options parameter by typing **options**, a Space, the option name **STR-retry=x** with a plus sign in front of it. Then press Enter.

```
ORACLE(ext-policy-server)# options +STR-retry=x
```

   If you type options and then the option value without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Gq Interface Features

The Oracle Communications Session Border Controller can run the Gq interface over a Diameter connection and act as a P-CSCF (AF) communicating with a PDF. The application ID field must be set to 16777222 to run the Gq reference point.

Rx Interface Features

The Oracle Communications Session Border Controller can run the Rx interface over a Diameter connection and act as a P-CSCF communicating with a PCRF. The application ID field must be set to 16777236 to run the Rx reference point. When running in this mode, the Oracle Communications Session Border Controller will send two AVPs, in addition to other information:

- The Codec-Data AVP is then included for non-priority calls. This AVP is one of several that together comprise a Group AVP structure.
- The Reservation-Priority AVP is included for priority calls. This AVP will be the main AVP within the AAR message.

Non-Priority Call Handling

When a SIP signaling event triggers external bandwidth management use, the Oracle Communications Session Border Controller removes all SDP information from the signaling message that was the trigger. The Oracle Communications Session Border Controller repackages this bandwidth information so that it can form a Bandwidth Request and decide on an external bandwidth manager to which it should be sent. If the appropriate external bandwidth manager is configured for Rx interface use, then Oracle Communications Session Border Controller then reformats the SDP information to construct a Codec-Data AVP.

If the external bandwidth manager that receives the request ignores the SDP information, then it does not include the Codec-Data AVP in the AAR.

For calls that do not require special treatment, the Codec-Data AVP is required to have the:

- AVP code 599
- 3GPP vendor identification number (10415)

- V (vendor) bit set in the AVP
- M (mandatory) bit set when sending this AVP
- Type octet string

In addition, the Codec-Data AVP appears as described in the following table.

AVP section/line	Requirement
Line 1	Must specify the direction of the flow by including the ASCII “uplink” or downlink: uplink—Identifies the SDP as having come from the UE and sent to the network downlink—Identifies the SDP as having come from the network and sent to the UE
Line 2	Must specify whether the offer or answer codec is at issue by including the ASCII “offer” (from an SDP offer according to RFC 3264) or answer (from an answer according to RFC 3264)
Remainder of the AVP	Must include lines found in the signaling SDP, formatted in ASCII and separated by new-line characters; the first line of this section must be the m line, followed by any “a” or “b” lines related to that m line

Priority Call Handling

The Oracle Communications Session Border Controller determines that a call is priority call when it matches a defined network management control (NMC) priority rule. No other scenario triggers the priority call handing treatment described in this section.

When a SIP signaling event triggers external bandwidth management use for a priority call, the Oracle Communications Session Border Controller sends the Reservation-Priority AVP in the AAR message. The Reservation-Priority AVP is required to:

- Use the ETSI Vendor identification number (13019)
- Have the V (vendor) bit set in the AVP
- Not to have the M (mandatory) bit set when sending this AVP
- Be of type enumeration
- Set to PRIORITY-SEVEN (7)

Rx bearer plane event

The Rx reference point is used to exchange application level session information between the PCRF and the AF which is the Oracle P-CSCF/SBC. The PCRF exchanges the Policy and Charging Control (PCC) rules with the PCEF (Policy and Charging Enforcement Function) and QoS rules with the BBERF (Bearer Binding and Event Reporting Function). The role of the former is typically performed by a 3GPP/GGSN, 3GPP-R9/PGW, 3GPP2/PDSN, WiFi/Max/ASN-GW. The role of the latter may be a standalone element or typically it’s incorporated within the system acting as the PCEF. The Oracle P-CSCF/SBC offers applications that require policy and charging control of traffic plane resources (e.g. UMTS PS domain/GPRS domain resources). Currently support for Rx within the Oracle Communications Session Border Controller supports policy, the present requirement will enhance Rx to incorporate traffic plane event notifications north bound. The PCRF receives session and media related information from the Oracle Communications Session Border Controller currently and with this enhancement the PCRF will report traffic plane events to the Oracle Communications Session Border Controller.

BN-Bearer-0040: When not all the service data flows within the AF session are affected, the PCRF will send an RAR command that includes the deactivated IP flows encoded in the Flows AVP and the cause encoded in the Specific-Action AVP.

BN-Bearer-0050: The Oracle SBC/P-CSCF shall acknowledge the receipt of this RAR command by sending an RAA. It shall then proceed to release the entire session at the SIP signaling layer and proceed to send the corresponding Diameter STR command to the PCRF and receive the STA.

AA-Request (AAR) Command

The following Specific-Action notifications will be added to the AA-Request command:

- INDICATION_OF_LOSS_OF_BEARER (2)
- INDICATION_OF_RECOVERY_OF_BEARER (3)
- INDICATION_OF_RELEASE_OF_BEARER (4)
- INDICATION_OF_OUT_OF_CREDIT (7)
- INDICATION_OF_SUCCESSFUL_RESOURCES_ALLOCATION (8)
- INDICATION_OF_FAILED_RESOURCES_ALLOCATION (9)

Re-Auth-Request (RAR) Command Handling

The Flows AVP (510) and Abort-Cause AVP (500) will be ignored if they are received in the RAR command.

The Specific-Action AVP (513) will be handled as follows:

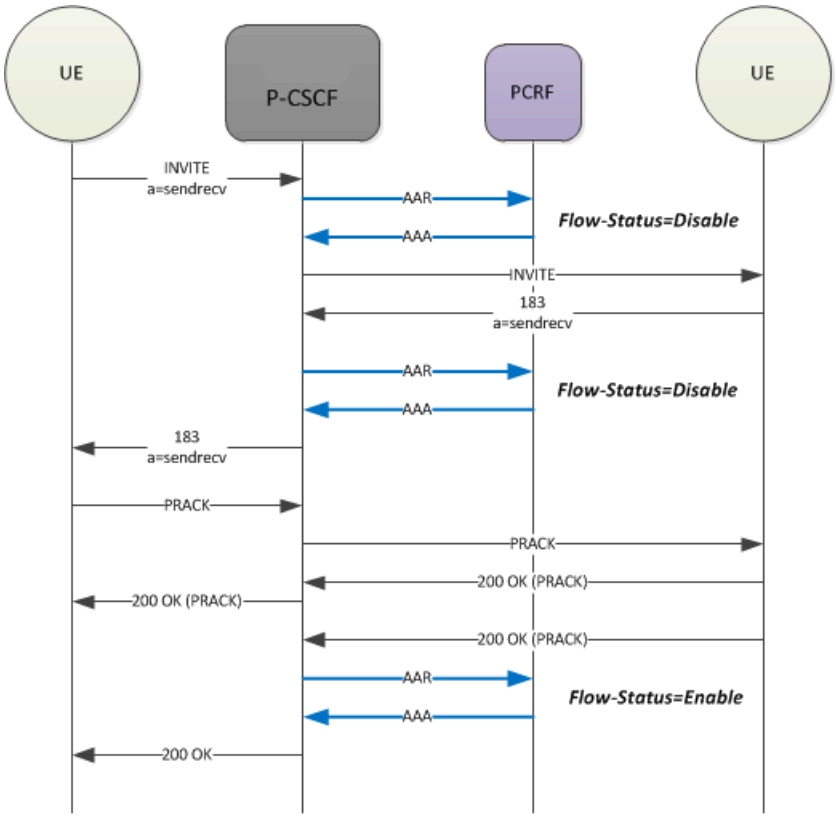
- Notification: INDICATION_OF_LOSS_OF_BEARER (Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_RELEASE_OF_BEARER (Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_OUT_OF_CREDIT(Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_FAILED_RESOURCES_ALLOCATION (Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_SUCCESSFUL_RESOURCES_ALLOCATION (Action: Send RAA).
- Notification: INDICATION_OF_RECOVERY_OF_BEARER (Action: Send RAA)

2836 - Early Media Suppression for Rx

The Oracle Communications Session Border Controller supports early media suppression where it does not allow media to flow between the endpoints it connects until the session is fully established as noted by receiving a 200 OK. Early media supression is configured by setting the **early-media-allow** parameter on a realm, realm group, or session agent to **none**.

The Oracle Communications Session Border Controller also relays the state of early media suppression in the Flow-Status AVP in an AAR message to the PCRF as the call is being set

up. When the call's media is still in a suppressed state, the Flow-Status AVP is set to Disabled. The following image illustrates when the Flow-Status changes from disable to enable.



Media-Component-Number and Flow-Number AVP Values

The values assigned to the Media-Component-Number and Flow-Number AVPs are not derived in compliance with 3GPP standards. Version S-CZ7.2.0 provides the capability to derive compliant values.

AA-Request (AAR) messages are sent, via the Rx Interface, by an Access Function (AF) to a Policy and Charging Rules Function (PCRF) in order to provide it with the session information. Among other fields, these messages contain a Media-Component-Number AVP and a Flow-Number AVP, used to identify specific media sessions, and the individual IP flows associated with these sessions.

The values assigned to the Media-Component-Number and Flow-Number AVPs are not derived in compliance with 3GPP standards.

Release S-CZ7.2.0 provides a new option that enables compliance with 3GPP requirements. Note that this support is limited to the Rx Interface

Media-Component-Number AVP

The Media-Component-Number AVP (AVP code 518) is of type Unsigned32; it contains the ordinal number of the media component (the SDP m= line) as specified in 3GPP TS 29.214. When this AVP refers to AF signalling, it contains the value 0. The Media-Component-Number AVP provides an index to the grouped Media-Component-Description AVP (AVP code 517).

3GPP compliance requires that:

1. Number should start from 1
2. It should contain the ordinal number of the position of the m= line in the SDP.
3. When this AVP refers to AF Signalling, this is indicated by using the value 0.
4. The ordinal number of a media component shall not be changed when the session description information is modified.

Flow-Number AVP

The Flow-Number AVP (AVP code 509) is of type Unsigned32; it contains the ordinal of the IP flows created to support a specific SDP m= line as specified in 3GPP TS 29.214. The Flow-Number AVP provides an index to the grouped Media-Sub-Component AVP (AVP code 519).

3GPP compliance requires that:

1. Number should start from 1
2. It should contain the ordinal number of the IP flow(s) within the m= line assigned in the order of increasing downlink destination port numbers, if downlink destination port numbers are available.
3. It should contain the ordinal number of the IP flow(s) within the m= line assigned in the order of increasing uplink destination port numbers, if no downlink destination port numbers are available.
4. The ordinal number assigned should not be changed when the session description information is modified.

Flow Examples

The following example illustrates the derivation of flow identifiers from SDP as specified by 3GPP TS 29.214 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Policy and Charging Control over Rx reference point (Version 9.3.0).

A flow identifier consists of two digits in the form #,# with the digit identifying a specific SDP m= line (defining a media session), and the second digit identifying a specific IP flow (for example, an RTP stream) supporting the media session. 3GPP standards require that the Media-Component-Number AVP be populated with the first digit if the flow identifier and the Flow-Number AVP be polulated with the second digit.

A UE, as the offerer, sends a SDP session description (relevant sections shown below) to an application server.

```
v=0
o=ecsreid 3262464865 3262464868 IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
s=MM01
i=One unidirectional audio media and one unidirectional video media and one
bidirectional application
media
t=3262377600 3262809600
m=video 50230 RTP/AVP 31                          Ordinal 1 (3 media flows, 1
RTP & 2 RTCP)
c=IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
a=recvonly
m=audio 50330 RTP/AVP 0                           Ordinal 2 (3 media flows, 1
RTP & 2 RTCP)
c=IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
```

```
a=sendonly
m=application 50430 udp wb                          Ordinal 3 (2 media flows,
up & down links)
c=IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
a=sendrecv
```

The application returns (relevant sections):

```
v=0
o=ecsreid 3262464865 3262464868 IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
s=MM01
i=One unidirectional audio media and one unidirectional video media and one
bidirectional application
media
t=3262377600 3262809600
m=video 51372 RTP/AVP 31                            Ordinal 1
c=IN IP6 2001:0646:000A:03A7:02D0:59FF:FE40:2014
a=sendonly
m=audio 49170 RTP/AVP 0                             Ordinal 2
c=IN IP6 2001:0646:000A:03A7:02D0:59FF:FE40:2014
a=recvonly
m=application 32416 udp wb                          Ordinal 3
c=IN IP6 2001:0646:000A:03A7:0250:DAFF:FE0E:C6F2
a=sendrecv
```

From this offer–answer exchange of SDP parameters the UE and the application server each creates a list of flow identifiers as shown below.

Order of m= line	Type of IP Flows	Destination IP Address/Port Number of Flows	Flow Identifier
1	RTP (Video) DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50230	<1,1>
1	RTCP DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50231	<1,2>
1	RTCP UL	2001:0646:000A:03A7:02D0:59FF:FE40:2014 / 51373	<1,2>
2	RTP (Audio) UL	2001:0646:000A:03A7:02D0:59FF:FE40:2014 / 49170	<2,1>
2	RTCP DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50331	<2,2>
2	RTCP UL	2001:0646:000A:03A7:02D0:59FF:FE40:2014 / 49171	<2,2>
3	UDP (application) DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50430	<3,1>
3	UDP (application) UL	2001:0646:000A:03A7:0250:DAFF:FE0E:C6F2 / 32416	<3,1>

Media-Component AVP 3GPP Compliance Configuration

Use this procedure to enable compliance with 3GPP requirements. Compliance is enabled with the **format-flow-id** option available in **ext-policy-server** configuration mode.

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. **format-flow-id**—Add this option to enable 3GPP compliance.

```
ACMEPACKET(ext-policy-server)# options +format-flow-id
```

4. Type **done** to save your configuration.

Rx Interface Reason Header Usage

The Oracle Communications Session Border Controller has increased capability to map network events to a configurable SIP Reason header. The extended capability provides for the mapping of specific disconnect events on the Rx Interface.

Release S-CZ7.2.0, and later releases expands the usage of the Reason header and its associated parameters to include mapping of disconnect events on the Rx interface between an SBC (functioning in the P-CSCF role) and a Policy and Charging Rules Function (PCRF). With the expanded capability enabled, contents of the Reason header are based on either the contents of the Specific-Action AVP in Re-Authorization Request (RAR) messages issued by the PCRF, or the contents of the Abort-Cause AVP in Abort Session Request (ASR) messages issued by the PCRF.

To use this feature, set the **session-router**, **sip-interface**, **rx-sip-reason-mapping** parameter to enabled.

Specific-Action AVP

The Specific-Action AVP (AVP code 513) is of type Enumerated.

Within a PCRF initiated Re-Authorization Request, the Specific-Action AVP determines the type of the action.

The following reported disconnected events can be mapped to SIP Reason headers using the corresponding value to configure in a **local-response-map-entry**, **local-error**.

Specific-Action AVP Value	local-error configuration
2 -- INDICATION_OF_LOSS_OF_BEARER	rx-rar-loss-of-bearer
4 -- INDICATION_OF_RELEASE_OF_BEARER	rx-rar-release-of-bearer
7 -- INDICATION_OF_OUT_OF_CREDIT	rx-rar-out-of-credit
9 -- INDICATION_OF_FAILED_RESOURCES_ALLOCATION	rx-rar-failed-resources-allocation

Abort-Action AVP

The Specific-Action AVP (AVP code 500) is of type Enumerated.

Within a PCRF initiated Abort Session Request (ASR), the Abort-Cause AVP identifies the cause of the ASR.

The following reported disconnected events can be mapped to SIP Reason headers using the corresponding value to configure in a **local-response-map-entry**, **local-error**.

Abort-Cause AVP Value	local-error configuration
0 -- BEARER_RELEASED	rx-asr-bearer-released
1 -- INSUFFICIENT_SERVER_SERVICES	rx-asr-insufficient-server-resources
2 -- INSUFFICIENT_BEARER_SERVICES	rx-asr-insufficient-bearer-resources
3 -- PS_TO_CS_HANDOVER	rx-asr-ps-to-cs-handover
4 -- SPONSORED_DATA_CONNECTIVITY_DISALLOWED	rx-asr-sponsored-data-connectivity-disallowed

Message Flows

Mapped headers will appear in either a BYE or a CANCEL message from the P-CSCF. As shown in the following illustrations, a BYE message is issued when the disconnect event occurs after establishing the SIP session. A CANCEL message is issued when the disconnect event occurs before establishing the SIP session.

RAR Loss-of-Bearer After Session Establishment

If mapping is not enabled, the Reason header defaults to SIP;cause=503;text="Service Unavailable”.

Network Provided Location Information

Network provided location information (NPLI) is a service commonly supported within the IMS network architecture. NPLI is most commonly provided as a geographic identifier which specifies a location either in terms of geodetic coordinates (latitude and longitude), for example 42°25'33 N 71°18'16 W; or in terms of a recognized civic identifier, for example Lincoln, Massachusetts.

NPLI Implementation

The Oracle Communications Session Border Controller (SBC) provides for an explicit subscription to NPLI changes resulting in the dynamic update of NPLI.

This NPLI implementation triggered by an originating mobile device typically involves a message exchange similar to the following.

1. Upon receiving the initial INVITE from the mobile handset, the P-CSCF sends an AAR command to the PCRF via the Rx Interface. This AAR contains a Required-Access-Info AVP that requests user location reports as they become available to the PCRF.
2. The P-CSCF buffers the INVITE pending the receipt of an AAA command from the PCRF. The AAA contains a 3GPP-User-Location-Info AVP along with a RAT-Type AVP. Depending upon network design, the AAA may also contain a proprietary MSISDN AVP, although this is not usually the case.

> **Note:**
>
> If you have set the **location-optimization-on-aar** option on the applicable **ext-policy-server** element, the SBC still receives the AAA, but it then waits for a RAR, from which it gets the 3GPP-user-location.

3. Upon receiving the AAA, the P-CSCF, using values contained in the 3GPP-User-Location-Info attribute, and the RAT-Type attribute, adds an extended PANI header to the buffered INVITE. If an MSISDN is included within the AAA, the P-CSCF also adds a P-Subscription-MSISDN header to the buffered INVITE.

> **Note:**
>
> If you have set the **location-optimization-on-aar** option on the applicable **ext-policy-server** element, the SBC still receives the AAA, but it then waits for a RAR, from which it gets the 3GPP-user-location.

4. The P-CSCF caches received attribute values, and forwards the altered INVITE to the IMS core.
5. At any point during the call (from initial session establishment to session termination) the PCRF can send a Re-Auth Request (RAR) that contains updated NPLI conveyed by the 3GPP-User-Location-Info, RAT-Type, IP-CAN-Type, or MSISDN AVPs.
6. After receiving the RAR, the P-CSCF updates the cached attribute values.
7. All subsequent re-INVITES and UPDATES contain the PANI and P-Subscription-MSISDN headers populated with the most recent cache values.

You configure global NPLI parameters in the **sip-config** or in profiles of those same NPLI parameters as an **npli-profile** element and apply them to a **sip-interface** to establish more granular control of NPLI management. The configuration at the **sip-interface** takes precedence. These configurations are explained in sections below.

Configuration Conflict

The **npli-upon-register** component of an **npli-profile** conflicts with the **provision-signaling-flow** configuration when they overlap. Exercise caution when applying a **npli-profile** to ensure

it does not conflict with any **provision-signaling-flow** configuration that you need. The SBC provides a configuration verification error when it detects this conflict.

Handling User-Endpoint-Provided Location Information

Handling of user-endpoint-provided location information is user configurable. By default, this information is not included in the PANI header constructed by the P-CSCF. This default state can be over-ridden, in which case the PANI header will contain both user-endpoint-provided location information and NPLI.

In the case where a user-endpoint PANI header contains an **np** parameter, the location information contained in the header is ignored, and is not included in the PANI constructed by the P-CSCF.

The following sample PANI headers illustrate the inclusion of both user-endpoint-provided and network-provided location information.

In this case, the UE -provided location information and NPLI matched.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601,
3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601; network-provided
```

In this case, the cell-IDs did not match.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601,
3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207abcde7A7777; network-provided
```

In this case, neither the network access type, nor the cell type matched.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601,
3GPP-GERAN; cgi-3gpp=26207FFFDF1B9601; network-provided
```

In this case, NPLI was not available.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=
```

Selective INVITE Holding for NPLI

When configured to use external policy servers, the Oracle Communications Session Border Controller allows the user to manage INVITE forwarding behavior based on preferences for inclusion of NPLI information. This forwarding behavior differs, depending on whether or not the call is an emergency call. The system also refers to external policy server, sip-interface and sip-config configuration to further specify when it forwards applicable INVITEs.

By default, the Oracle Communications Session Border Controller holds non-emergency calls until an AAR/AAA transaction is complete, but does not wait for this exchange before forwarding emergency calls. These AAR/AAA transactions may or may not provide NPLI information. If not, the system may forward the INVITE using location information that comes from sources other than the network. As the call proceeds, the system may also correct and/or update location information with NPLI from subsequent SDP and Re-Authentication Request (RAR) transactions.

The system uses the parameters below to specify location information in addition to or as alternatives to NPLI:

- When enabled, the sip-config's **include-ue-loc-info** instructs the system to use location information provided by the endpoint as an additional PANI location header. In this case, the system always waits for the AAA before forwarding the INVITE.
- The sip-interface's **default-location-str** parameter allows the user to configure a default location string, which the system can use if, for example, it is not provided by any other means.
  When configured and when the AAA has a RAT type of 'WLAN', the **default-location-str-VoWifi**, within an applicable sip-interface's **npli-profile**, takes precedence over the **default-location-str** value.

To further manage this INVITE holding, the user configures sip-config timers to wait for AAR/AAA transactions, including:

- **hold-invite-calls-for-loc-info**
- **hold-emergency-calls-for-loc-info**

When set, these timers modify system behavior as follows:

- For non-emergency INVITEs, the system sends AAR requests and waits for the AAA response.
  - If NPLI is in the AAA, The system forward the INVITE using that NPLI.
  - If the AAA does not include NPLI, the system continues to holds the INVITE waiting for a RAR.
  - If the **hold-invite-calls-for-loc-info** timer expires and the system has still not received location information, it forwards the INVITE with PANI headers populated with either or both the default location string value and the location provided by the UE.
  - If the system later receives a RAR with NPLI, it sends subsequent messages using that information.
- For emergency INVITEs, the system sends AAR requests and waits for the AAA response.
  - If NPLI is in the AAA, The system forward the INVITE using that NPLI.
  - If the AAA does not include NPLI, the system continues to holds the INVITE waiting for a RAR.
  - If the **hold-emergency-calls-for-loc-info** timer expires and the system has still not received location information, it forwards the INVITE with PANI headers populated with either or both the default location string value and the location provided by the UE.
  - If the system later receives a RAR with NPLI, it sends subsequent messages using that information.

There are two situations that cause the system to ignore these timers:

- If the PCRF configuration has **reserve-incomplete=disabled**, the system skips the AAR request. The system ignores the timer and sends the INVITE out with the default location string, if configured.
- If the initial INVITE arrives with no SDP, the system does not send out an AAR. The system ignores the timer and sends the INVITE out with the default location string, if configured.

> **Note:**
>
> The **hold-invite-calls-for-aaa** option is no longer operational.

Optimizing Location Information Handling

The Oracle Communications Session Border Controller (SBC) can perform procedures to identify and manage location over the Rx interface using AAR/AAA transactions with the **ext-policy-server**. The standard behavior is to issue the AAR after the 200 OK for every register and other SIP messages. The standard SBC behavior also includes sending the Required-Access-Info AVP and a Specific-Action-AVP with a value of ACCESS_NETWORK_INFO_REPORT for every AAR. You can configure the SBC to limit (optimize) the number of times it sends these AVPs and set timers to control when it sends responses to the initial INVITE are sent. This configuration limits the number of times the SBC sends a PANI to the core to once per session and changes the hold timer to start on receipt of the AAA instead of the INVITE.

The SBC issues AARs for a variety of reasons, and includes the AVPs mentioned above when you set the applicable **ext-policy-server**, **specificActionSubscription** parameter to **access-network-info-report**. To implement this optimization, you enable the **location-optimization-on-aar** option on the applicable **ext-policy-server**.

```
ORACLE(ext-policy-server)# +options location-optimization-on-aar
```

You can configure this option in conjunction with the **start-hold-timer-event** in the **sip-config** to refine when the system starts waiting for the expiry of the **hold-emergency-calls-for-loc-info**, **hold-invite-calls-for-loc-info** or **msg-hold-for-loc-info** timer, depending on the traffic type. During this hold, the system waits for NPLI information to arrive in the RAR. By default, the system begins this timer after it sends out an AAR. You can configure the system to start this timer after it receives the AAA.

> **Note:**
>
> If you turn on the option without changing the **start-hold-timer-event** parameter, the system provides a configuration verification notice telling you to set the **start-hold-timer-event** parameter to **aaa**. This option conflicts with the AAR setting.

This behavior affects SIP calls only, not impacting REGISTER or SMS sessions. It also requires the **reserve-incomplete** on the applicable **ext-policy-server** be set to **orig-realm-only**.

Operational detail includes:

- When the SBC generates its first AAR for session initiation/establishment and inserts NPLI AVP's, depending on whether you have set the **specific-action-subscription** to **access-network-info-report**. This subscribes the to Network Provided Location Information (NPLI) changes.
- Whenever the AAA includes the RAT-type as WLAN, the SBC uses the value configured in **default-location-string-VoWifi** from the applicable **npli-profile**. If **default-location-string-VoWifi** is empty, the **default-location-string** in the **realm-config**.
- When this feature is enabled, the SBC feature check for NetLocAccess AVP of incoming AAA. If the value of this AVP is zero, the SBC does not start the hold timer even if it is configured and the NPLI feature is enabled.
- Although the SBC stores the NPLI location when it comes in the AAA, if another location comes in a RAR, the SBC overwrites the saved location with the latest location received in the RAR.

- If the UE is from a trusted realm, the SBC sends a PANI header to UE once.
- The SBC implements a timer to determine when to issue these messages, depending on specific configuration and circumstances:
  - For terminating calls and when **reserve-incomplete** is set to **orig-realm-only**, and NPLI is enabled, the SBC holds the first a-sdp it finds, which could be provided by a 18x SIP reply with SDP or a 200 OK to the Invite, until the NPLI information is provided through a RAR or the timer expires.
  - For terminating calls and when **reserve-incomplete** is set to **disabled** (AAR on SDP answer), the SBC holds the first 18x response with SDP until NPLI is provided through a RAR or the timer expires.

Related Configuration

There are three timers related to NPLI optimization that you can configure on each applicable **sip-interface** or **realm-config**. The **sip-interface** configuration takes precedence:

- hold-emergency-calls-for-loc-info
- hold-invite-calls-for-loc-info
- msg-hold-for-loc-info

If you have enabled **asynchronous-mode** on the applicable **ext-policy-server**, the SBC does not hold the INVITE even if hold timers are set, effectively disabling the feature.

The value configured for the **sip-interface**, **npli-upon-register** parameter takes precedence over the value configured in **sip-config**.

Rules for Issuing PANI

This table displays the matrix of different configuration and traffic combinations that result in the SBC issuing UE and Network PANI.

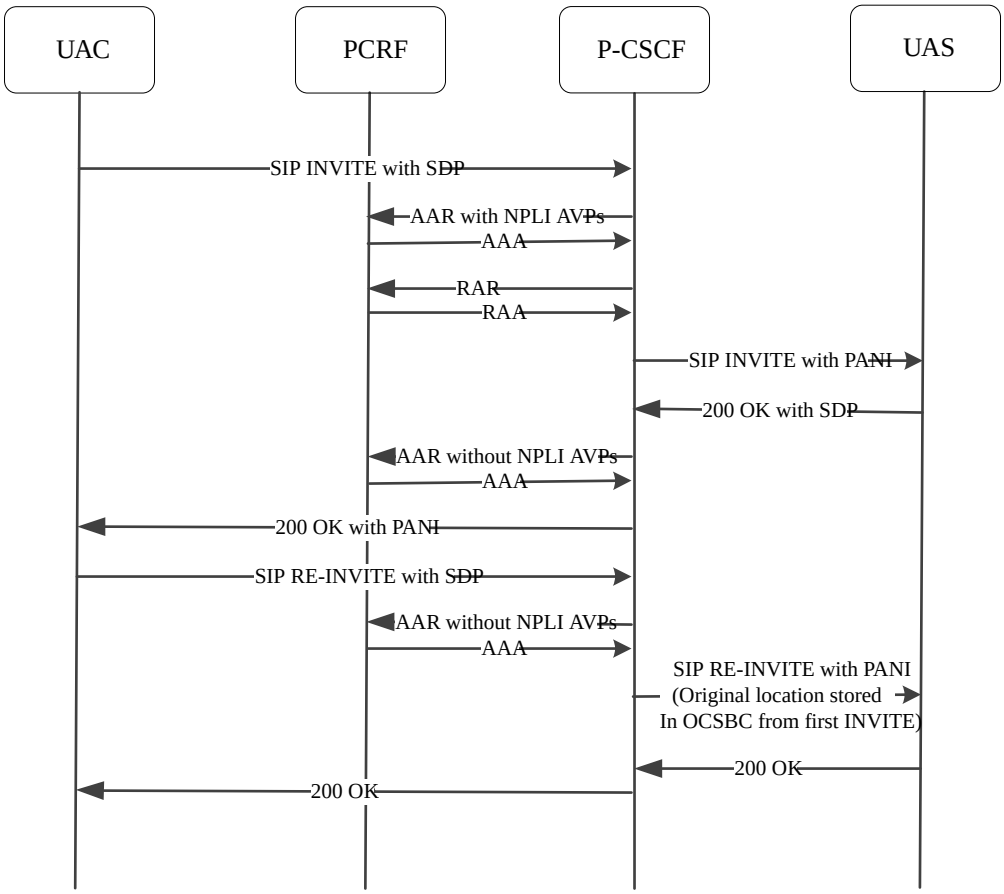
add-ue-location-in-pani	allow-pani-for-trusted-only	Is endpoint Trusted	Req/Res has UE added PANI	UE PANI has network-provided string	RxDip Done	Got Network Provided PANI from PCRF	Manipulated PANI (np - network-provided)
Enabled	Enabled	No	Yes	Yes	Yes	Yes	No PANI sent to UE
Enabled	Enabled	Yes	Yes	Yes	Yes	Yes	UE-PANI (w/o np string),Network-PANI;network-provided
Enabled	Disabled	No	Yes	Yes	Yes	Yes	UE-PANI (w/o np string),Network-PANI;network-provided
Disabled	Enabled	Yes	Yes	Yes	Yes	Yes	Network-PANI;network-provided
Disabled	Enabled	No	Yes	Yes	Yes	Yes	No PANI sent to UE

add-ue-location-in-pani	allow-pani-for-trusted-only	Is endpoint Trusted	Req/Res has UE added PANI	UE PANI has network-provided string	RxDip Done	Got Network Provided PANI from PCRF	Manipulated PANI (np - network-provided)
Disabled	Disabled	No	Yes	Yes	Yes	Yes	Network-PANI;network-provided
Enabled	Enabled	Yes	Yes	Yes	Yes	No	UE-PANI(w/o np string),default-PANI string(w/o np string)
Enabled	Enabled	No	Yes	Yes	Yes	No	No PANI sent to UE
Enabled	Enabled	Yes	No	No	Yes	No	Default-PANI string (w/o np string)
Enabled	Enabled	Yes	Yes	Yes	No	No	UE-PANI(w/o np string) to core
Enabled	Enabled	Yes	No	No	No	No	No PANI sent

Example Call Flows

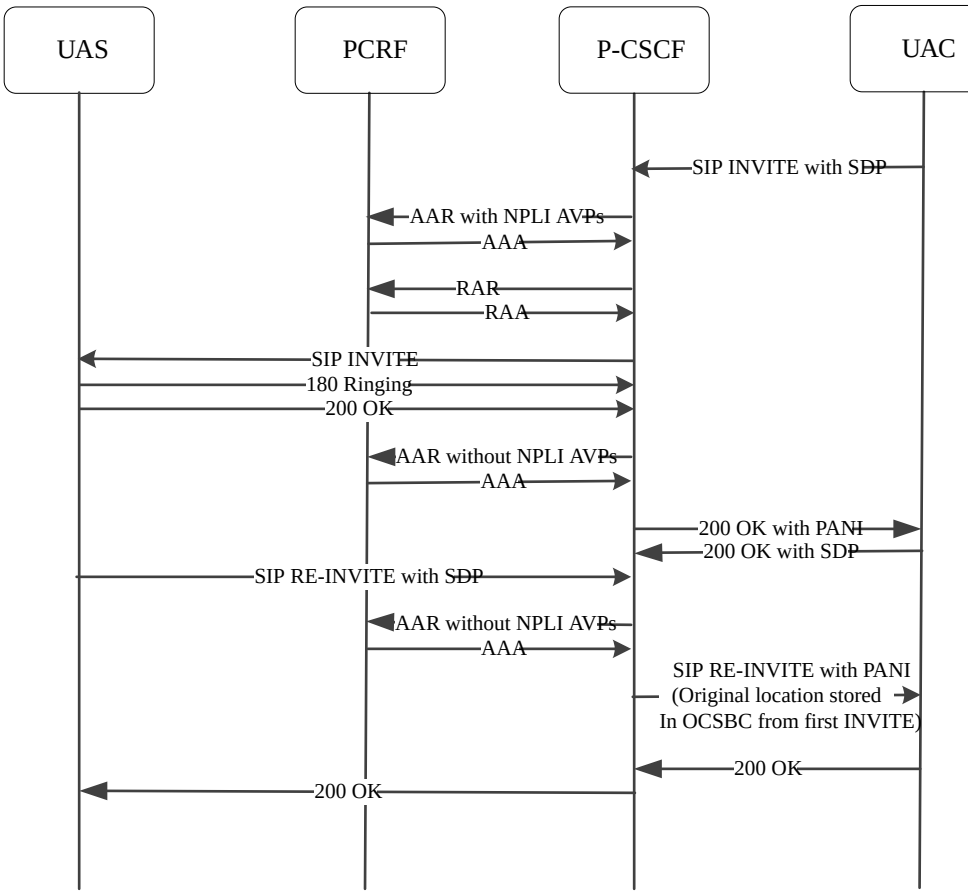
When the SBC receives an INVITE for a mobile originating call, which originates from inside the core, it sends an AAR with the NPLI AVPs (and other AVPs) to the PCRF. When it receives the AAA, the SBC fetches and stores the IP-CAN and RAT type. The SBC then waits for the RAR, which usually includes the location information to be used for constructing the PANI header in the INVITE to the UAS. When the SBC receives the 200OK with the answer SDP, it sends another AAR to the PCRF. This time, however, the AAR does not contain the NPLI AVPs. Similarly, when it receives a Re-Invite later, the SBC sends the AAR to the PCRF without the NPLI AVPs.

Note that the SBC sends the NPLIs AVP to the PRCF only once in this SIP session.



When the SBC receives an INVITE for a mobile terminating call, which originates from outside the core, it sends an AAR with the NPLI AVPs (and other AVPs) to PCRF. When it receives the AAA, the again SBC fetches and stores the IP-CAN and RAT type. When the RAR comes, the location is stored. When the SBC receives the 200OK with the answer SDP, it sends an AAR to the PCRF. This time, however, the AAR does not contain the NPLI AVPs. The SBC uses the stored location information for the PANI towards the core. Similarly when the SBC later receives a Re-Invite, it sends the AAR to the PCRF without the NPLI AVPs.

Again, the SBC sends the NPLIs AVP to the PRCF only once in this SIP session.


UAS
PCRF
P-CSCF
UAC
SIP INVITE with SDP
AAR with NPLI AVPs
AAA
RAR
RAA
SIP INVITE
180 Ringing
200 OK
AAR without NPLI AVPs
AAA
200 OK with PANI
200 OK with SDP
SIP RE-INVITE with SDP
AAR without NPLI AVPs
AAA
SIP RE-INVITE with PANI
(Original location stored
In OCSBC from first INVITE)
200 OK
200 OK

NPLI Emergency Call Processing

Currently, when the P-CSCF sends AAR requests to the PCRF for an emergency call, the P-CSCF does not wait for the response (AAA) to forward the INVITE to the core. Instead, it forwards the INVITE as soon as it sends the AAR to the PCRF. Even if the subsequent AAA reports a failure response, the P-CSCF ignores the failure and allows emergency call processing.

Recent requirements issued by Bundesnetzagentur (BNetzA), the German telecom regulator, affect the processing of emergency calls originated by mobile handsets. The German requirements mandate that INVITES for establishment of emergency calls, be buffered, for some specified period of time, pending receipt of NPLI from the PCRF. If the PCRF fails to provide NPLI within that time period, the P-CSCF forwards the emergency INVITE to the network core with a PANI header populated with both the default location string value (if configured) and with any UE-provided location information (if available, and if configured to do so). If location information is unavailable, the P-CSCF forwards the INVITE without a PANI header.

See the NPLI Emergency Call Configuration procedure to configure a hold-timer that specifies the time (in seconds) that emergency INVITES are buffered while waiting for NPLI from the PCRF. Such a timer ensures compliance with BNetzA regulations. Users who are not subject to BNetzA authority can choose to implement a minimum timer value, which provides the network some period of time to report possibly more accurate location information.

Handling NPLI for Unregistered Emergency Calls

Despite the absence of a SIP contact within the context of an unregistered user, the SBC manages network provided location information (NPLI) within call flows for unregistered emergency calls using the same controls it uses for registered flows. There is no additional configuration required to support unregistered calls. Common behavior for registered and unregistered emergency calls include the SBC managing NPLI AVP's in AARs and using the same timing controls to fine tune this management.

When configured, the SBC creates an AAR to be sent to the PCRF to initiate and establish the session, inserts NPLI AVPs and subscribes the SBC to NPLI changes. The SBC holds the INVITE after sending the AAR, even after the AAA comes from the PCRF.

As is true for registered emergency calls, configuration required for this feature includes:

- Include the **access-network-info-report** in the **specific-action-subscription** parameter in the applicable **ext-policy- server**.
- Enable the **location-optimization-on-aar** option on the applicable **ext-policy-server**.
- Disable **optimize-aar** on the applicable **ext-policy-server**.

Additional configuration that affects this feature include:

- Configuring the **hold-emergency-calls-for-loc-info** parameter to a value greater than zero.
- Configuring the **cache-loc-info-expire** parameter to zero. If not set to zero, the SBC adds the default PANI when this location cache timer expires.
  You can configure **hold-emergency-calls-for-loc-info** and **cache-loc-info-expire** in an **npli-profile** applied to the applicable **sip-interface** and/or the **sip-config**. The values in the **npli-profile** take precedence.

When receiving an unregistered emergency call, the SBC generates an AAR with NPLI AVPs and sends it to the PCRF for session initiation/establishment. At this point, the SBC holds the

INVITE waiting for a RAR, even if it receives an AAA. When the RAR arrives, the SBC stops the hold timer and constructs its PANI header for inclusion within the subsequent INVITE as follows:

- If the AAA/RAR includes location information, the system used that to frame the PANI.
- If the incoming AAA has the RAT-type set to WLAN the SBC includes the value configured in **default-vowifi-string** in the **npli-profile** in the PANI.
  Having determined it is using the **default-vowifi-string**, the system ignores any timers and forwards the INVITE immediately.
- If either of the following are true, the SBC immediately populates its PANI header with your configured **default-location-string** or any UE-provided location, and forwards the SIP request without waiting for hold timer expiry:
  - The incoming AAA from the PCRF has the NetLoc-Access-Support AVP set to 0
  - The RAR from the PCRF includes the INDICATION_OF_ACCESS_NETWORK_INFO_REPORTING_FAILURE (specific-action AVP)
- If the SBC does not receive location information from the PCRF and you have configured a **default-location-string**, it uses the **default-location-string** in the PANI.

> **Note:**
>
> If you have configured the default location string in both the **sip-interface** and the **realm**, the **realm** configuration takes precedence.

- If the SBC does not receive location information and there is no **default-location-string**, it does not include network-provided PANI.
- The SBC includes any UE-provided in addition to network-provided location information if the UE is trusted, meaning you have enabled the **trust-me** parameter on the applicable **session-agent**.
  If the UE is not trusted, the SBC does not include UE-provided location information.

Feature Interactions with S8HR

If an S8HR unregistered emergency call arrives at an SBC with the NPLI feature enabled, the SBC initiates an Rx exchange requesting EPC-level identities (MSISDN, IMEI, IMSI) along with NPLI AVP's in an AAR. It sends this to the PCRF and waits for the AAA. When the SBC receives the AAA, it validates EPC-level identities and waits for the RAR. When the SBC receives the RAR, it constructs the PANI header and inserts it, along with the P-Asserted-Identity, into the INVITE, and sends the INVITE to the UAS.

CDRs

You can configure the SBC to generate an INTERIM Call Detail Record (CDR) for an unregistered Emergency MO call. This configuration would include CDR generation and NPLI management. When the SBC then receives an unregistered emergency INVITE, the SBC retrieves policy information, including NPLI, from the PCRF during the AAR/AAA and RAR/RAA sequences over the Rx interface.

The SBC includes any NPLI provided by the PCRF in the CDRs, and creates the interim record after the egress INVITE.

High Availability

This feature is supported by high availability (HA) for:

- MO Call
- MT Call
- Registered emergency call
- Unregistered emergency call

> **Note:**
>
> This feature does not support HA for REGISTER and SMS.

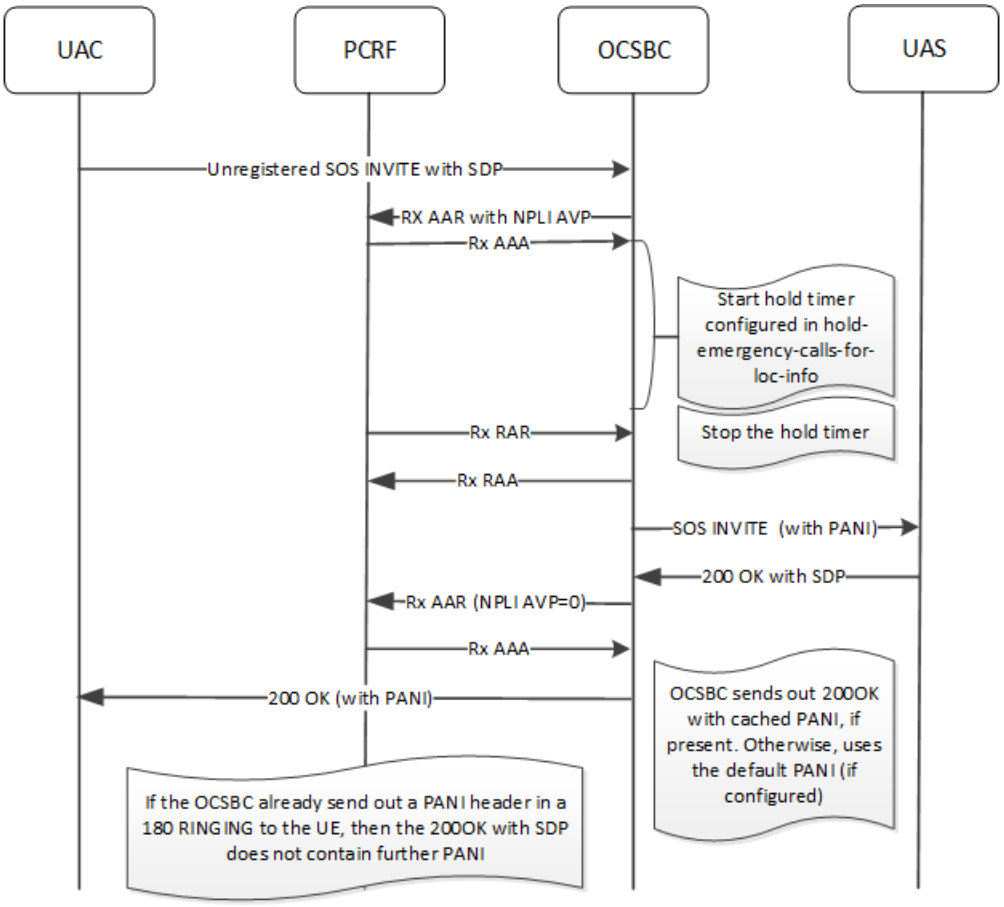
This feature does not require any additional configuration parameter to support HA because the SBC replicates the required configuration on the standby.

When a call is established at the ACTIVE SBC and a switchover occurs, the newly ACTIVE SBC continues to perform NPLI management. If it receives a re-INVITE for this call, the newly active SBC does not send NPLI AVP's in the AAR initiated for re-INVITE. Also the newly active SBC does not send the network provided PANI header while forwarding the re-INVITE. Additionally, the SBC does not hold the re-INVITE (to wait for RAR). Instead, it sends the re-INVITE to the core immediately after receiving the AAA.

Flows Handling NPLI for Unregistered Emergency Calls

The diagram below shows an example flow for an unregistered emergency MO call. In this scenario, you have enabled the NPLI feature and set the **hold-emergency-calls-for-loc-info** parameter.

1. Because you have configured the NPLI management feature, the SBC holds the unregistered emergency INVITE, waiting for the RAR.
2. When the RAR comes with the 3GPP-User-Location-INFO AVP, the SBC constructs the PANI header in unregistered emergency INVITE and sends towards UAS. (Recall that, if the RAR comes without location information, the SBC uses your default PANI, if configured.)
3. When the SBC receives the 18x/200OK, it sends the AAR to the PCRF again. This time, however, it does not contain NPLI AVPs, and the SBC holds the 18x/200OK until it receives the AAA.
4. As soon as AAA comes, the SBC uses the cached PANI or the default PANI to insert a PANI header in the 18x/200OK and sends it to the UE. (Recall that, if you have not configured a default PANI, the SBC does not send a PANI header to the UE.)



Note:

if the SBC has already send out the PANI header in the 18x to UE, then it does not send a PANI header within any subsequent 18x/200OK with SDP. Regardless, receiving any 18x/200OK with SDP continues to trigger the AAR, and the wait for the AAA.

Contrast this flow above with the one below, which depicts an unregistered emergency MO call. This time, you have not configured the SBC with the NPLI management feature, but you have configured the **hold-emergency-calls-for-loc-info**.

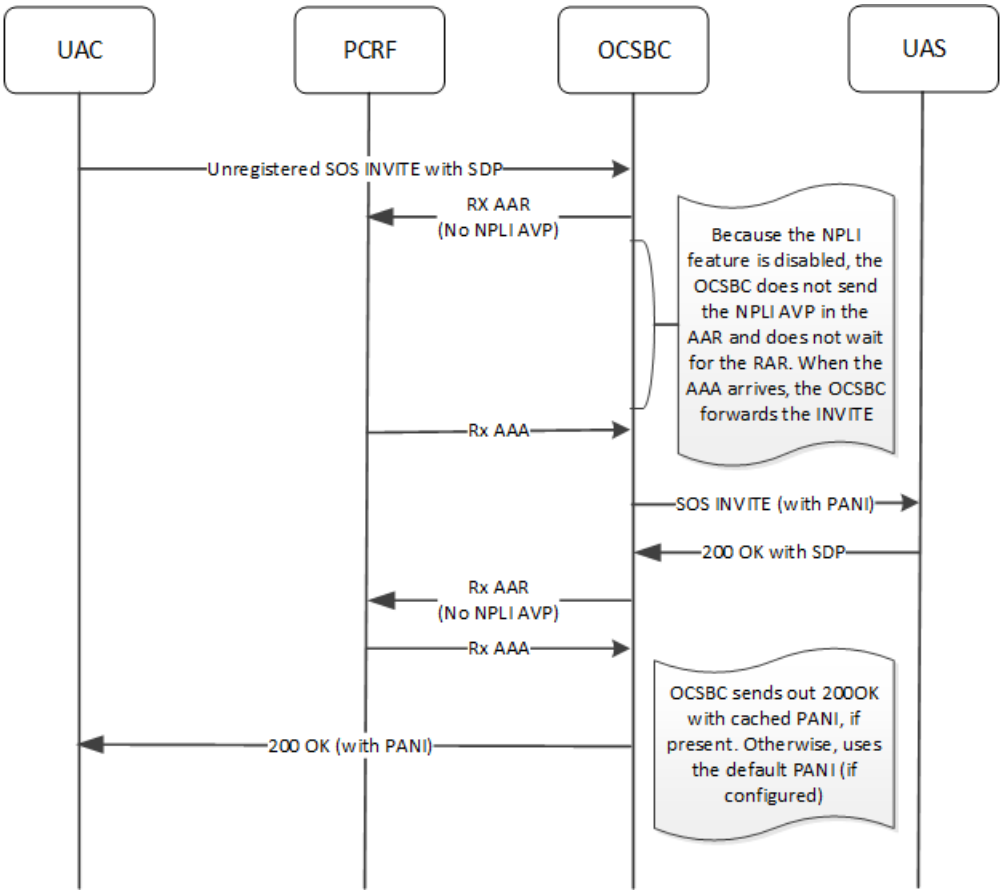
1. Upon receiving the unregistered emergency INVITE, the SBC sends an AAR without NPLI AVP's and waits for AAA to come. The SBC does not wait for a RAR.
2. When the AAA arrives without location information, the SBC sends the INVITE to the UAS with the default PANI, if configured.

Note:

If the SBC does not receive an AAA, it drops the call and sends a 503 error to the UAC.

3. As soon as it receives the AAA, the SBC forwards the INVITE without PANI.
4. The call proceeds normally without any PANI.

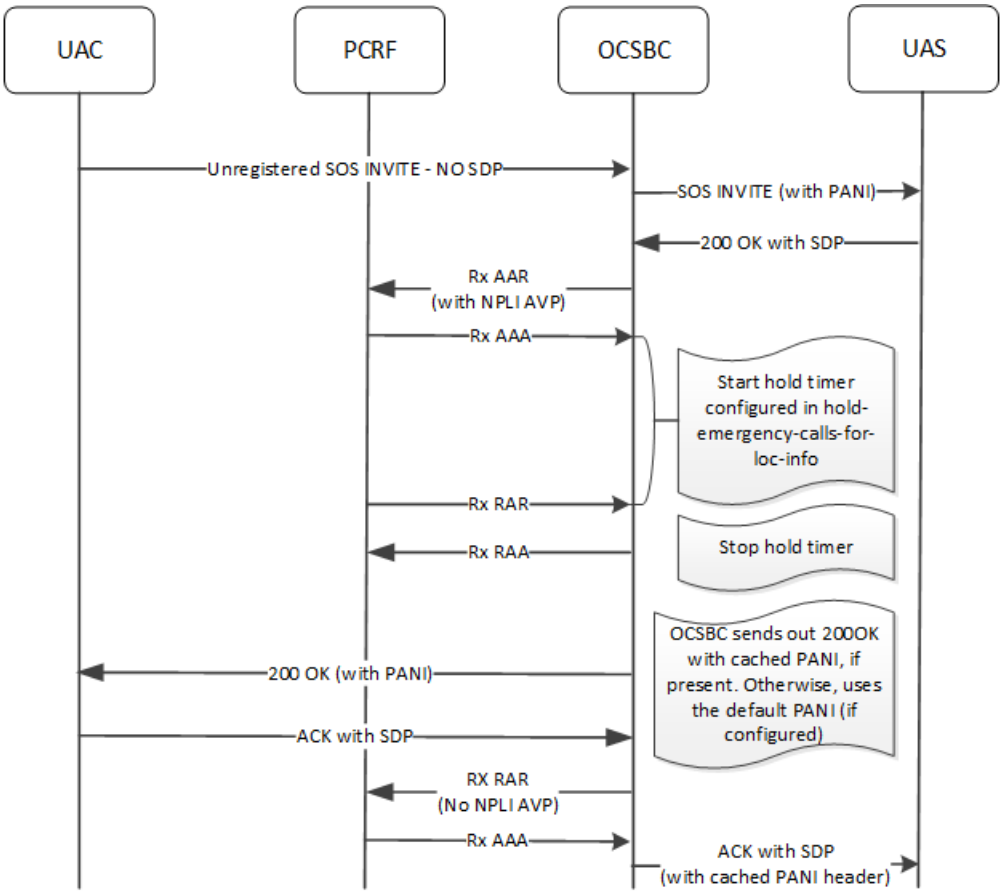
5. Any subsequent Rx dips do not include the NPLI AVP.



The next flow presents a successful unregistered emergency MO call with an Offer-less Invite.

1. Upon receiving the unregistered emergency INVITE, the SBC does not send out an AAR because the INVITE has no SDP. Instead, it simply forwards the INVITE without any PANI.
2. Upon receiving a 200 OK with an SDP Offer, the SBC now generates an AAR with NPLI AVP's, and sends it to the PCRF.
3. The SBC receives the AAA, it starts its hold timer and waits for the RAR, which includes NPLI.
4. Upon receiving the RAR, the SBC constructs the PANI, adds it to the 200 OK, and sends it to the UAC.
5. The SBC saves this NPLI in its cache, keeping the NPLI valid for the duration of your **cache-loc-info-expire** configuration.
6. When the ACK arrives, the SBC retrieves the stored NPLI and, ultimately, sends the PANI to the core/ target UAS.

> **Note:**
>
> If you configure **cache-loc-info-expire** to zero, the SBC retains the NPLI for the duration of the session. This is recommended for using this PANI feature for unregistered emergency flows.



NPLI Support for 5G NR

The Oracle Communications Session Border Controller (SBC) provides 5G NR Location support, which enables interoperability with 5G core elements in 5G systems. This allows the SBC to operate as an Application Function and in its role as a P-CSCF in the 5G core. Many carriers deploy a separate 5GS with the N26 interface in parallel with their older architectures. This newer architecture continues to use the Rx diameter interface to interwork with the PCF/PCRF and route diameter messages via the DRA. As an A-SBC, the SBC supports this architecture, further enabling operation within 5G. The SBC achieves this support through the **ext-policy-server**, **specific-action-subscription**, **access-network-info-report** configuration, which enables it to support additional 5G objects and behaviors, including objects that provide location information. No additional configuration is required to support NR location support for 5G. The SBC can also include the resulting 5G location fields within all CDR types.

The SBC supports 5G NPLI within the following call flows:

- 5G-NR Registration
- Registered MO and MT Calls
- SMS from Registered User
- Registered Emergency Calls
- Emergency Call from Registered S8HR roaming user

5G elements supported by the SBC to construct the 5G NPLI include:

- 5G access-type/access-class in the SIP PANI Header on the Gm and Mw interfaces

- 5G-NR specific values in 3GPP-User-Location-Info on the Rx interface, per 3GPP TS 29.212 v16.4.0 and 3GPP TS 29.061 v16.2.0, including:
  - 3GPP-User-Location-Info AVP with the following Geographical Location Types, decoded per 3GPP TS 23.003 v16.5.0 and 3GPP TS 38.413 v16.4.0:
    * 135—NCGI
    * 136—5GS TAI
    * 137—5GS TAI and NCGI
  - RAT-Type AVP value 3GPP-NR (1006)
  - IP-CAN type AVP value 3GPP-5GS (8)

5G NPLI Operation

When the SBC receives a SIP message with a PANI header that has its access-class set to 3GPP-NR, 3GPP-NR-FDD or 3GPP-NR-TDD, it recognizes and treats the header as a 5G supported PANI header. Behavior, timing and header population of the SBC refer to the same configurations it uses to handle PANI prior to 5G. When working from information presented over diameter, the SBC continues to build the PANI based on information received over the Rx interface.Having received 5G location information from the PCRF, the SBC processes and populates the outgoing PANI header with the appropriate information, based on configuration. The provider can then use this location information for purposes including Charging, Emergency Call Routing, and so forth.

As soon as it receives the initial SIP message, the SBC sends a request for location information to the PCF/PCRF by sending an AAR with the Required-Access-Info AVP set to the value LOCATION(0) and the Specific-Action AVP set to the value ACCESS_NETWORK_INFO_REPORT(12). After performing this Rx dip, the SBC builds the 5G NPLI string for the SIP PANI header using 5G values received within the RAR from the PCF/PCRF in the following AVPs:

- RAT-Type—Builds the access-class using the RAT type received in the RAT-Type AVP
- IP-CAN-Type—3GPP-5GS
- 3GPP-User-Location-Info—The SBC builds the "utran-cell-id-3gpp" parameter for SIP PANI header (for 5G) is encoded as per Section 7.2A.4.3, points 22/22A of 3GPP TS 24.229 v16.4.0. It is a concatenation of:
  - MCC (3 decimal digits)
  - MNC (2 or 3 decimal digits depending on MCC value)
  - Tracking Area Code (6 hexadecimal digits) as described in 3GPP TS 23.003
  - The NR Cell Identity (NCI) (9 hexadecimal digits)

> **Note:**
>
> Be sure to configure the applicable **default-location-string** with a valid 5G value. The SBC does not validate this string.

When creating a SIP PANI header for 5G, the SBC follows the 3GPP-specific extended syntax of the PANI header field per 3GPP TS 24.229 v16.8.0, RFC 7315 and RFC 7913. The SBC restricts its changes to the header to the following fields:

```
<access-class>;<access-info>=<location>;<np>
```

- access-class = “3GPP-NR”—Set based on RAT type received in Diameter RAT-Type AVP
- access-info = utran-cell-id-3gpp—If the access-class is “3GPP-NR” and the 3GPP-User-Location-Info contains “137 – 5GS TAI and NCGI”, then the SBC sets the utran-cell-id-3gpp to "MCC+MNC+TAC+NCI"
- np = “network-provided” (if applicable)

5G NPLI Call Flows

As stated, there are no changes to existing call flows when using 5G NPLI. The difference is in the construction of the NPLI because the SBC receives 5G-specific values in Rx AVPs, parsing and storing them appropriately so it can construct the 5G location string (NPLI). This section provides some call flow examples to display operation when 5G specific values are received in the AVPs.

For all of these call flows, the following configuration is common:

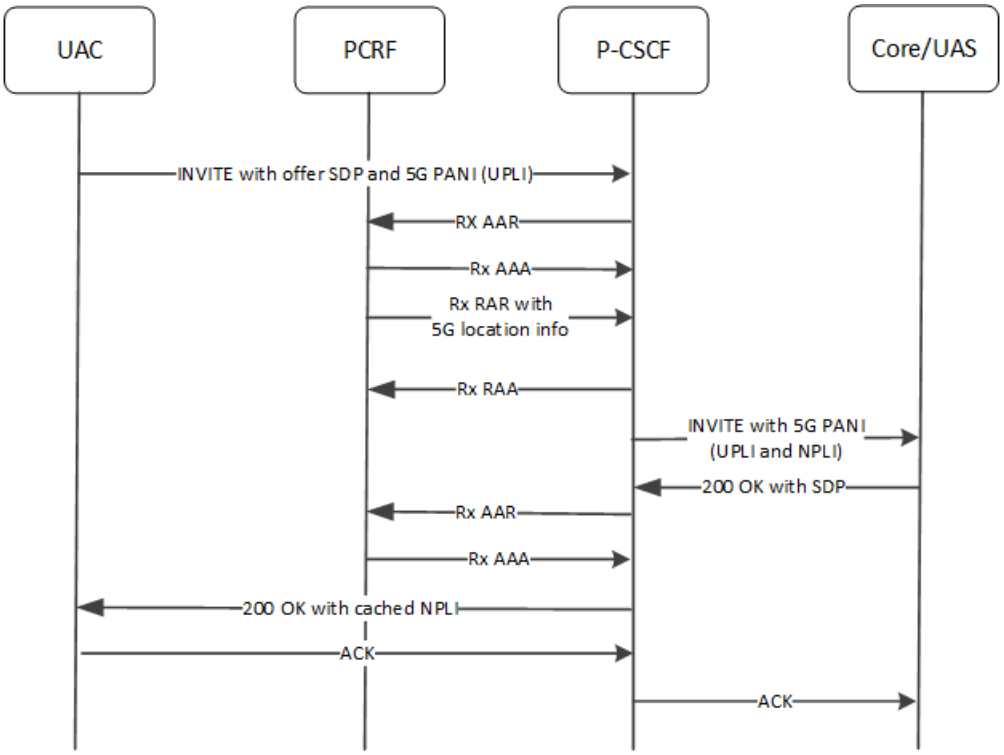
- Set **hold-invite-calls-for-loc-info** to a non-zero value and **start-hold-timer-event** to AAA. This causes the SBC to start the hold timer after receiving the AAA and wait for RAR from PCF/PCRF.
- Configured the **location-optimization-on-aar** feature.
- Configured the **access-network-info-report** in the **specific-action-subscription**.

5G PANI in MO with Calling Endpoint Trusted

For this call flow, the calling UE is registered, trusted and you have enabled **add-ue-location-in-pani**.

The procedure depicted in the image below includes:

1. The UAC sends an INVITE with an offer SDP to the SBC with the 5G PANI Header (UPLI).
2. The SBC sends an AAR (INITIAL_REQUEST) to the PCF/PCRF, requesting location Information, and waits for the RAR.
3. The PCF/PCRF sends 5G supported location information in the 3GPP-User-Location-Info AVP of the RAR.
4. The SBC constructs the NPLI from RAR and sends the INVITE out with a 5G PANI header (UPLI + NPLI).
5. When it receives a 200 OK with answer SDP, the SBC sends an AAR (UPDATE_REQUEST) to the PCF/PCRF requesting location information again. However, it does not wait for the RAR.
6. The SBC uses the cached NPLI (generated during initial INVITE) and adds it as the PANI header in the 200 OK.
7. Because the UAC (access) is trusted, the SBC also adds the 5G NPLI towards the UAC.



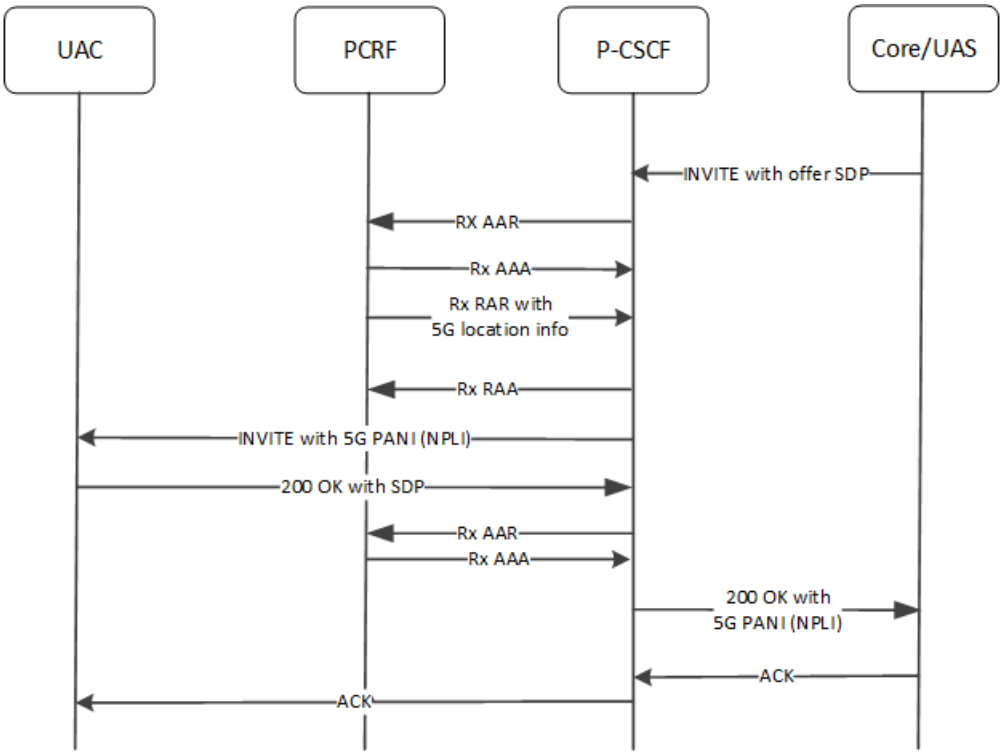
5G PANI in MT Call

For this call flow, the called UE is registered, trusted, and the caller is on the core side. The procedure depicted in the image below includes:

1. The SBC receives offer an SDP from the core and sends the AAR (INITIAL_REQUEST) to the PCF/PCRF requesting location Information.
2. When the SBC receives the AAA, it starts the hold timer and waits for the RAR.
3. The PCF/PCRF sends 5G specific location information in the 3GPP-User-Location-Info AVP within the RAR.
4. The SBC constructs the NPLI from the RAR and sends the INVITE (towards access) with a 5G PANI header (NPLI).

> **Note:**
>
> The access UE is trusted.

5. When the SBC receives the 200 OK with the answer SDP, it sends an AAR (UPDATE_REQUEST) to the PCF/PCRF.
6. The SBC uses the cached NPLI, generated during the initial INVITE, and adds it as the PANI header in the 200 OK towards the Core for that session.



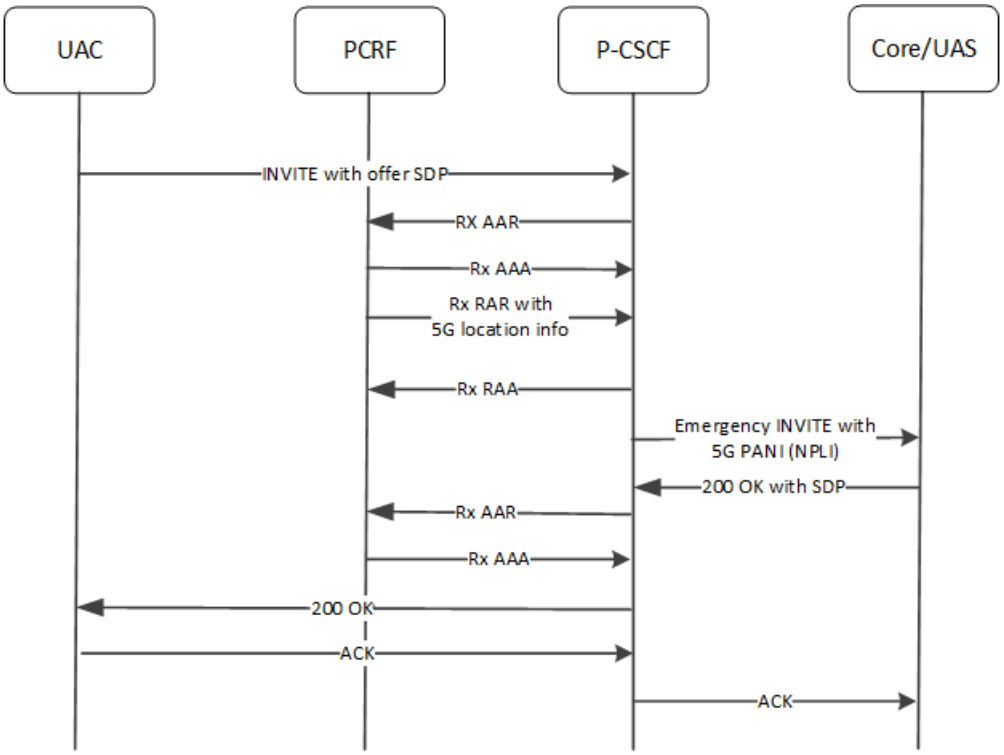
Registered Emergency Calls

SBC behavior (including the configuration parameters) for adding PANI header for Emergency calls shall remain the same as in previous release. Key detail to keep in mind includes:

- You can configure the **hold-emergency-calls-for-loc-info** to enable and set the timing (in seconds) for the buffering of emergency INVITES, while waiting for location information from the PCRF.
- If the PCRF fails to provide NPLI within that time period, the P-CSCF (SBC) forwards the emergency INVITE to the core network with a PANI header populated from the default location string value (if configured).
- If location information is unavailable, the P-CSCF forwards the INVITE without a PANI header.

For this call flow, the UAC is registered, but not trusted. In addition, you have enabled **allow-pani-for-trusted-only**. The procedure depicted in the image below includes:

1. The SBC receives Emergency INVITE from UAC (Registered UE) and it sends AAR (INITIAL_REQUEST) to PCF/PCRF requesting location Information.
2. When the The SBC receives the AAA, it starts the hold timer and waits for the RAR.
3. The PCF/PCRF sends 5G specific location information in the 3GPP-User-Location-Info AVP within the RAR.
4. The SBC constructs NPLI from the RAR.
5. The SBC sends the INVITE out with the 5G PANI header (NPLI).
6. When the SBC receives the 200 OK with the answer SDP, it sends an AAR (UPDATE_REQUEST) to the PCF/PCRF.
7. The SBC forwards the 200 OK to UE with no NPLI header because it is untrusted.



Key AVP Support

3GPP-User-Location-Info AVP

The 3GPP-User-Location-Info AVP (type 22) is defined in section 16.4.7.2, Coding 3GPP Vendor-Specific RADIUS attributes within the 3GPP TS29.061, Interworking between the Public Land Mobile Network (PLMN) specification. This AVP supports packet based services and Packet Data Networks (PDN).

3GPP-User-Location-Info contains the location type, for example a cell global identifier (CGI), and actual location data for the specified cell.

The SBC supports this attribute. It occurs in AAA messages received by the P-CSCF.

The AVP consists of four fields:

- The 1-octet 3GPP-type field contains the value 22.
- The 1-octet 3GPP-length field contains the attribute length in octets.
- The 1-octet Geographic Location Type field identifies the equipment type whose location is being provided.
- The variable length Geographic Location field contains the geographic coordinates of the equipment specified in the Geographic Location Type field.

This AVP contains the location type, for example a cell global identifier (CGI), and the actual location data for the specified cell.

```
0 = CGI
1 = SAI
2 = RAI
```

```
128 = TAI
129 = ECGI
130 = TAI and ECGI
135 = NCGI
136 = 5GS TAI
137 - 5GS TAI and NCGI
```

Other values are reserved for future use.

IP-CAN-Type AVP

The IP-CAN-Type AVP (type 1027) specifies the type of Connectivity Access Network to which the user is connected. It occurs in AAA and RAR messages received by the P-CSCF.

The following values are defined.

```
0 = 3GPP-GPRS
    indicates 3GPP General Packet Radio Service (GPRS) access and is further
detailed by the RAT-Type AVP that contains an applicable value (not EUTRAN).
1 = DOCSIS
    indicates Data Over Cable Service Interface Specification (DOCSIS) access.
2 = xDSL
    indicates Data Over Cable Service Interface Specification (DOCSIS) access.
3 = WiMAX
    indicates Worldwide Interoperability for Microwave Access (WiMAX -- IEEE
802.16).
4 = 3GPP2
    indicates CDMA2000 access as specified in 3GPP2 X.S0011 and is further
detailed by the RAT-Type AVP.
5 = 3GPP-EPS
    indicates 3GPP Evolved Packet System (EPS) access and is further detailed
by the RAT-Type AVP.
6 = Non-3GPP-EPS
    indicates Evolved Packet System (EPS) based on non-3GPP access technology
and is further detailed by the RAT-Type AVP.
8 = 3GPP-5GS
    indicates 5G and is further detailed by the RAT-Type AVP.
```

MSISDN AVP

The Vodafone proprietary MSISDN AVP is supported. This attribute can be found in AAR and RAR messages received by the P-CSCF. It is constructed as follows.

```
<Name>MSISDN-VF:21274</Name>
<AvpName>MSISDN-VF</AvpName>
<AvpCode>3701</AvpCode>
<VendorId>21274</VendorId>
<MandatoryFlag>false</MandatoryFlag>
<ProtectFlag>false</ProtectFlag>
<MayencryptFlag>false</MayencryptFlag>
<VendorSpecificFlag>true</VendorSpecificFlag>
<AvpType>utf8String</AvpType>
```

RAT-Type AVP

The RAT-Type AVP (attribute type 1032) specifies the radio access technology type. This is present in AAA and RAR messages received from PCRF. For purposes of NPLI, the following values are supported:

```
   0 - WLAN
   1 - Virtual (indicates that RAT is unknown)
1000 – Universal Terrestrial Radio Access Network (UTRAN)
1001 – GSM Edge Radio Access Network (GERAN)
1002 - GAN (Generic Access Network)
1003 - HSPA_EVOLUTION (Evolved High Speed Packet Access)
1004 – Evolved UMTS Radio Access Network (EUTRAN)
1006 – 5G NR
2000 - CDMA2000_1X (core CDMA2000 wireless air interface)
2001 - HRPD (High Rate Packet Data)
2002 - UMB (Ultra Mobile Broadband)
2003 - EHRPD (Enhanced High Rate Packet Data)
```

In the event that a received RAT-Type attribute contains an unsupported type, an error condition is declared and the default PANI is used.

NPLI Required-Access-Info AVP

The Required-Access-Info AVP (type 536) conveys a subscription request for NPLI. It occurs in AAR messages originated by the P-CSCF.

The following values are defined.

```
0 - subscribe to NPLI
1 - subscribe to user timezone information
```

NPLI Specific-Action AVP

The Specific-Action AVP (type 513) works in conjunction with the Required-Access-Info AVP (type 536) to convey an NPLI subscription request to the PCRF. This attribute occurs in AAR messages originated by the P-CSCF.

The following value is used to request NPLI:

```
12 – ACCESS_NETWORK_INFO_REPORT
```

Key Header Support

Extended PANI SIP Header

The Oracle Communications Session Border Controller provides an extended PANI header as defined in 3GPPTS 24.229, IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3.

Extended PANI header syntax is as shown below.

```
P-Access-Network-Info  = "P-Access-Network-Info" HCOLON
                                 access-net-spec *(COMMA access-net-spec)
access-net-spec                = (access-type / access-class) *(SEMI access-
info)
access-type                    = "IEEE-802.11" / "IEEE-802.11a" /
"IEEE-802.11b" / "IEEE-802.11g" /
                                 "IEEE-802.11n" / "3GPP-GERAN" / "3GPP-UTRAN-
FDD" / "3GPP-UTRAN-TDD" /
                                 "3GPP-E-UTRAN-FDD" / "3GPP-E-UTRAN-TDD" /
"ADSL" / "ADSL2" / "ADSL2+" /
                                 "RADSL" / "SDSL" / "HDSL" / "HDSL2" /
"G.SHDSL" / "VDSL" / "IDSL" /
                                 "3GPP2-1X" / "3GPP2-1X-Femto" / "3GPP2-1X-
HRPD" / "3GPP2-UMB" /
                                 "DOCSIS" / "IEEE-802.3" / "IEEE-802.3a" /
"IEEE-802.3e" / "IEEE-802.3i" /
                                 "IEEE-802.3j" / "IEEE-802.3u" /
"IEEE-802.3ab"/ "IEEE-802.3ae" /
                                 "IEEE-802.3ah" / "IEEE-802.3ak" /
"IEEE-802.3aq" / "IEEE-802.3an" /
                                 "IEEE-802.3y" / "IEEE-802.3z" / GPON/
XGPON1 / "GSTN"/ "DVB-RCS2"
                                 "3GPP-NR-FDD" / "3GPP-NR-TDD" / token
access-class                   = "3GPP-GERAN" / "3GPP-UTRAN" / "3GPP-E-
UTRAN" / "3GPP-WLAN" / "3GPP-GAN" /
                                 "3GPP-HSPA" / "3GPP2" / "3GPP-NR" / token
np                             = "network-provided"
access-info                    = cgi-3gpp / utran-cell-id-3gpp / dsl-
location / i-wlan-node-id / ci-3gpp2 /
                                 ci-3gpp2-femto / eth-location / fiber-
location / np/ gstn-location /
                                 local-time-zone / dvb-rcs2-node-id /
extension-access-info
extension-access-info          = generic-param
cgi-3gpp                       = "cgi-3gpp" EQUAL (token / quoted-string)
utran-cell-id-3gpp             = "utran-cell-id-3gpp" EQUAL (token / quoted-
string)
i-wlan-node-id                 = "i-wlan-node-id" EQUAL (token / quoted-
string)
dsl-location                   = "dsl-location" EQUAL (token / quoted-string)
eth-location                   = "eth-location" EQUAL (token / quoted-string)
fiber-location                 = "fiber-location" EQUAL (token / quoted-
string)
ci-3gpp2                       = "ci-3gpp2" EQUAL (token / quoted-string)
ci-3gpp2-femto                 = "ci-3gpp2-femto" EQUAL (token / quoted-
string)
gstn-location                  = "gstn-location" EQUAL (token / quoted-string)
dvb-rcs2-node-id               = "dvb-rcs2-node-id" EQUAL quoted-string
local-time-zone                = "local-time-zone" EQUAL (token / quoted-
string)
```

P-Subscription-MSISDN SIP Header

The Proprietary P-Subscription-MSISDN SIP header is supported.

Its syntax is as follows.

```
P-Subscription-MSISDN = "P-Subscription-MSISDN" HCOLON
                                    MSISDN

        MSISDN        =             msidsn value (string)
```

P-Access-Network-Info SIP Header

The P-Access-Network-Info SIP header is defined in section 4.4 of RFC 3455, *Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)*. This header contains information on the access network that the user handset is using to get IP connectivity, and is typically ignored by intermediate proxies between the SBC that adds the header and the SIP proxy that is providing services. The proxy providing services can inspect the header and make use of the information contained there to provide appropriate services, depending on the value of the header. Before proxying the request onwards, this proxy strips the header from the message.

The P-Access-Network-Info header contains an access-type field identifying the radio access technology as received in the RAT-Type attribute.

- 3GPP-UTRAN — when the RAT- Type AVP value is 1000
- 3GPP-GERAN — when the RAT- Type AVP value is 1001
- 3GPP-E-UTRAN — when the RAT- Type AVP value is 1004

The access-type field is followed by the access-info field which identifies the mobile phone cell that provides network access. The location of this cell provides the means to geographically locate connected mobile handsets.

- utran-cell-id-3gpp — when the RAT- Type AVP value is 1000
- cgi-3gpp — when the RAT- Type AVP value is 1001
- utran-cell-id-3gpp — when the RAT- Type AVP value is 1004

The access-info field is followed by an EQUAL sign and a CellID/CGI (Cell Global Identifier) field that contains an octet string which provides the location value contained in the 3GPP-User-Location-Info AVP. The location value can be either an CGI or an Extended CGI (ECGI).

- CGI field — when the RAT- Type AVP value is 1000
- CGI field — when the RAT- Type AVP value is 1001
- ECGI field— when the RAT- Type AVP value is 1004

Examples of P-Access-Network-Info headers are as follows:

If the RAT-Type is UTRAN:

P-Access-Network-Info: 3GPP-UTRAN;utran-cell-id-3gpp=C359A3913B20E

If the RAT-Type is GERAN:

P-Access-Network-Info: 3GPP-GERAN;cgi-3gpp=62F8100005C599

If the RAT-Type is EUTRAN:

P-Access-Network-Info: 3GPP-E-UTRAN;utran-cell-id-3gpp=C359A3913B20E

NPLI Configuration

Enabling NPLI Notifications

The P-CSCF sends the PCRF AAR messages containing the Specific-Action and Required-Access-Info AVPs to indicate a desired notification/subscription type. Use the following procedure to request NPLI notifications.

1. Move to ext-policy-server configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# ext-policy-server
ACMEPACKET(ext-policy-server)#
```

2. Use the **select** command to identify the target PCRF.

```
ACMEPACKET(ext-policy-server)# select
name:
...
...
...
4. PCRF_10A
...
...
...
selection: 4

ACMEPACKET(ext-policy-server)#
```

3. Use the **specific-action-subscription** parameter to specify one or more specific actions. When designating 2 or more actions, enclose them in quotation marks, with the values separated by spaces. Use **access-network-info-report** to enable NPLI support.

```
ACMEPACKET(ext-policy-server)# specific-action-subscription
access-network-info-report
ACMEPACKET(ext-policy-server)#
```

4. Use **done**, **exit** and **verify-config** to complete configuration.

Configuring Default PANI Contents

Users can configure default contents of the P-Access-Network-Info header in the event of any failure during the NPLI transfer. By default, such failure results in the generation of an empty header. In some circumstances a more informative header may be of benefit to the user.

Default PANI contents can be configured at either the SIP interface level, or at the realm level. If values are configured at both levels, the realm value takes precedence. A value configured at the sip-interface level takes precedence over a default value (empty string) specified at the realm level.

Use the following procedure to configure default PANI contents.

1. Navigate to either the sip-interface or realm configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sip-interface
ACMEPACKET(sip-interface)#
```

or

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# realm-config
ACMEPACKET(realm-config)#
```

2. Use the **select** command to specify the target SIP interface or realm.
3. Use the **default-location-string** parameter to specify the default contents of the P-Access-Network-Info SIP header in the event that NPLI is not available.

```
ACMEPACKET(sip-interface)# default-location-sting "Location services
temporarily unavailable."
ACMEPACKET(sip-interface)#
```

or

```
ACMEPACKET(realm-config)# default-location-sting "Location services not
unavailable."
ACMEPACKET(realm-config)#
```

4. Use **done**, **exit**, and **verify-config** to complete configuration.

Bear in mind also the **default-location-srt-VoWifi** parameter if there is an **npli-profile** assigned to the **sip-interface**. If configured, this value takes precedence over the **default-location-str** parameter that may be configured in both the **realm-config** and **sip-interface**.

Configuring the NPLI Profile

You configure profiles of the same NPLI parameters presented in the **sip-config** and apply them to a **sip-interface** to establish more granular control of NPLI management. The configuration at the **sip-interface** takes precedence.

1. Navigate to either the sip-interface or realm configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)#npli-profile
ACMEPACKET(npli-profile)#
```

2. Type in a name for your profile, or use the **select** command to select an existing profile to edit.

```
ACMEPACKET(npli-profile)# name my-npli-profile
```

3. Enable this parameter to add UE Location string in the PANI header when the location is available.

```
ACMEPACKET(npli-profile)# add-ue-location-in-pani enable
```

4. Specify, in seconds, this timer to hold emergency calls until the system receives location information from the PCRF.

```
ACMEPACKET(npli-profile)# hold-emergency-calls-for-loc-info 200
```

5. Specify, in seconds, this timer to wait for the location information from the PCRF in a RAR over the Rx interface, assuming it is greater than 0, and reserve-incomplete is enabled.

```
ACMEPACKET(npli-profile)# hold-invite-calls-for-loc-info 200
```

6. Specify, in seconds, the maximum number of seconds after which the system drops network location information for the NPLI for the Short Message feature, unless the keep-cached-loc-info-after-timeout parameter is enabled.

```
ACMEPACKET(npli-profile)# cache-loc-info-expire 200
```

7. Specify, in seconds, the maximum number of seconds that the system holds MESSAGEs for location information for the NPLI for the Short Message feature.

```
ACMEPACKET(npli-profile)# msg-hold-for-loc-info 20
```

8. Enable this parameter to capture Network Provided Location Information during the Registration process.

```
ACMEPACKET(npli-profile)# npli-upon-register enable
```

9. Enable this parameter to allow the PANI header only for trusted domains.

```
ACMEPACKET(npli-profile)# allow-pani-for-trusted-only enable
```

10. Specify the default contents of the P-Access-Network-Info header in the event of any failure during the NPLI transfer. By default, such failure results in the generation of an empty header. In some circumstances a more informative header may be of benefit to the user. In contrast to the other parameters in this profile, this parameter takes precedence over the **sip-interface**, **default-location-str parameter**, as opposed to the global parameters in the **sip-config**.

```
ACMEPACKET(npli-profile)# default-location-string-VoWifi MyLocation
```

11. Use **done**, **exit**, **exit** and **verify-config** to complete configuration.
12. Apply your profile to the applicable **sip-interface**.

User-Endpoint-Provided Location Information Configuration

Use the following procedure to include user-endpoint-provided location information in the PANI constructed by the P-CSCF. This capability is disabled by default.

1. Move to sip-config configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sip-config
ACMEPACKET(sip-config)#
```

2. **add-ue-location-in-pani**—Use this parameter to enable the inclusion of user-endpoint-provided location information in PANI headers.

   With the parameter enabled, the PANI header contains both user-endpoint-provided location information in addition to NPLI.

   By default, **add-ue-location-in-pani** is disabled.

```
ACMEPACKET(sip-config)# add-ue-location-in-pani enable
```

3. Use **done**, **exit** and **verify-config** to complete configuration.

NPLI Emergency Call Configuration

Use this procedure to buffer emergency INVITES pending the receipt of NPLI from the PCRF. This procedure is required for users subject to German telecommunications regulations. For other users, the procedure is optional.

1. Navigate to sip-config configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sip-config
ACMEPACKET(sip-config)#
```

2. **hold-emergency-calls-for-loc-info**—Use this parameter to both enable the buffering of emergency INVITES (thus complying with German regulations), and to specify the time (in seconds) that INVITES are buffered.

   By default **hold-emergency-calls-for-loc-info** is set to the value 0, which disables the buffering of emergency INVITEs.

   Set **hold-emergency-calls-for-loc-info** to a non-zero value, within the boundaries acceptable to the German regulator, to enable the timer.

```
ACMEPACKET(sip-config)# hold-emergency-calls-for-loc-info 5
ACMEPACKET(sip-config)#
```

3. Use **done**, **exit** and **verify-config** to complete this configuration

Caveats

Within environments that provide NPLI support, but do not have the **location-optimization-on-aar** option set, the caveats below are applicable:

- **reserve-incomplete** (ext-policy-srv mode) — set to enabled
- **optimize-aar** (ext-policy-srv mode) — set to disable

Rf Interface Features

Node-Functionality AVP Support

The Oracle Communications Session Border Controller sends the Node-Functionality (862) AVP in all Rf ACR messages.

The Node-Functionality AVP indicates the function that the message's source plays in the network. The CDF/CGF function that collects the ACR messages can use the information in the Node Functionality AVP for billing or analysis purposes.

In an IMS network, theOracle Communications Session Border Controller may perform the following functions: P-CSCF, E-CSCF, IBCF, BGCF (when configured as an Oracle Communications Session Router). In fact, the system might perform different roles simultaneously, so that on a call-by-call basis, the value in the Node-Functionality might change.

To accurately reflect multiple, simultaneous functions that the Oracle Communications Session Border Controller performs, the value inserted into the Node-Functionality AVP may be defined per realm. The node functionality value for a call's ACR is taken from the configuration in the ingress realm. Each realm may only be marked with a single Node Functionality value.

The system can still be configured with a single, global Node Functionality value. This is done in the SIP config's node functionality parameter. When configured, all system-generated ACRs include this value. However, if the node functionality parameter is also configured in a realm config, the ingress realm's node functionality value supersedes the global value.

The node-functionality in the **realm-config** may be configured with an empty string (). This indicates that this realm should revert to the global node-functionality value.

Node Functionality AVP Configuration

To configure a global Node Functionality AVP value:

1. Navigate to the **sip-config** configuration element.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# sip-config
   ACMEPACKET(sip-config)#
   ```

2. Type **select** to begin configuring this object.

   ```
   ACMEPACKET(sip-config)# select
   ACMEPACKET(sip-config)#
   ```

3. **node-functionality** — Enter a global value to insert into the Node-Functionality AVP when the system sends ACRs over the Rf interface to an appropriate destination. The default is P-CSCF. Valid values are:
   - P-CSCF
   - BGCF
   - IBCF
   - E-CSCF

4. Save and activate your configuration.

Node Functionality Per Realm Configuration

To configure a Node Functionality value per realm:

1. Navigate to the **realm-config** configuration element.

```
ACMEPACKET(session-router)# exit
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# realm-config
ACMEPACKET(realm-config)#
```

2. Select the ingress realm where calls generate ACR's whose node functionality AVP will be marked as configured.

```
ACMEPACKET(realm-config)# select
identifier:
1: realm01    left-left:0      0.0.0.0
selection: 1
ACMEPACKET(realm-config)#
```

3. **node-functionality** — Enter the value to insert into the Node-Functionality AVP when the system sends ACRs for calls that enter the system from this realm. The default is empty which uses the global node functionality value configured in the sip-config configuration element. Valid values are:
   - P-CSCF
   - BGCF
   - IBCF
   - E-CSCF
4. Save and activate your configuration.

AAR Message Optimization

Currently, upon receiving an INVITE with a Proxy-Authorization header, the P-CSCF sends an Authorization-Authentication Request (AAR) message when the values of the **optimize-aar** and the **reserve-incomplete** parameters in the **ext-policy-server** configuration element are set to **enabled**. If the INVITE does not contain a Proxy-Authorization header then an AAR message is not sent. However, for mobile VoLTE, because IMS AKA is used for security there is no need to authenticate requests, so all INVITE messages are sent without Proxy-Authorization headers. This enhancement allows the P-CSCF to generate an AAR message when the INVITE is identified as coming from an IMS AKA user.

When the Oracle Communications Session Border Controller acts as a P-CSCF, the Diameter Rx interface updates bandwidth and addressing changes during a session. Many transactions between the P-CSCF and a PCRF server trigger a new SDP offer resulting in the transmission of an P-CSCF-initiated AA-Request (AAR) sent to the PCRF for the purpose of updating bearer parameters. However, not all transactions need to be reported to the PCRF, for example transactions that do no carry any bandwidth or addressing changes. In such instances, the issuance of an AAR is unnecessary and wasteful.

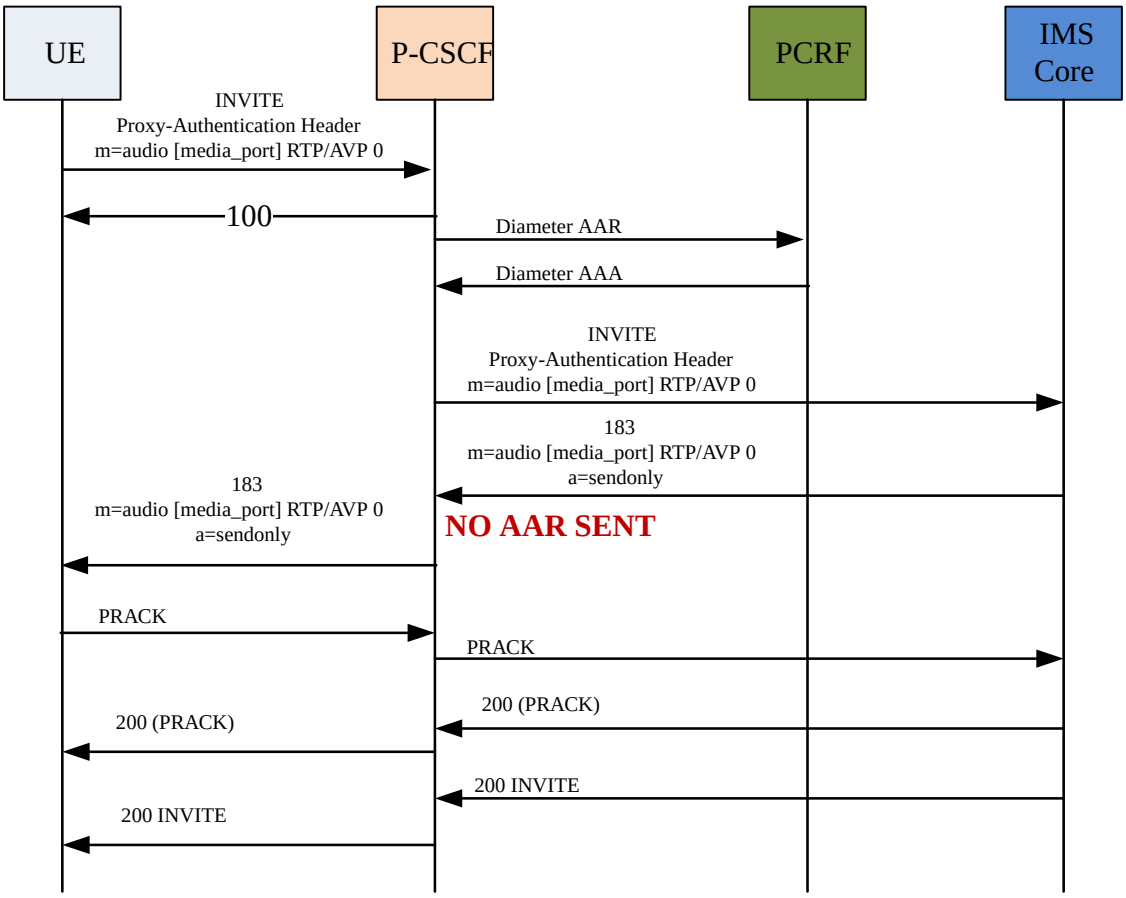
AAR message optimization (the suppression of unnecessary AARs) is activated by setting the **optimize-aar** parameter in the **ext-policy-server** configuration element to **enabled**.

If optimization is enabled, AARs are suppressed when:

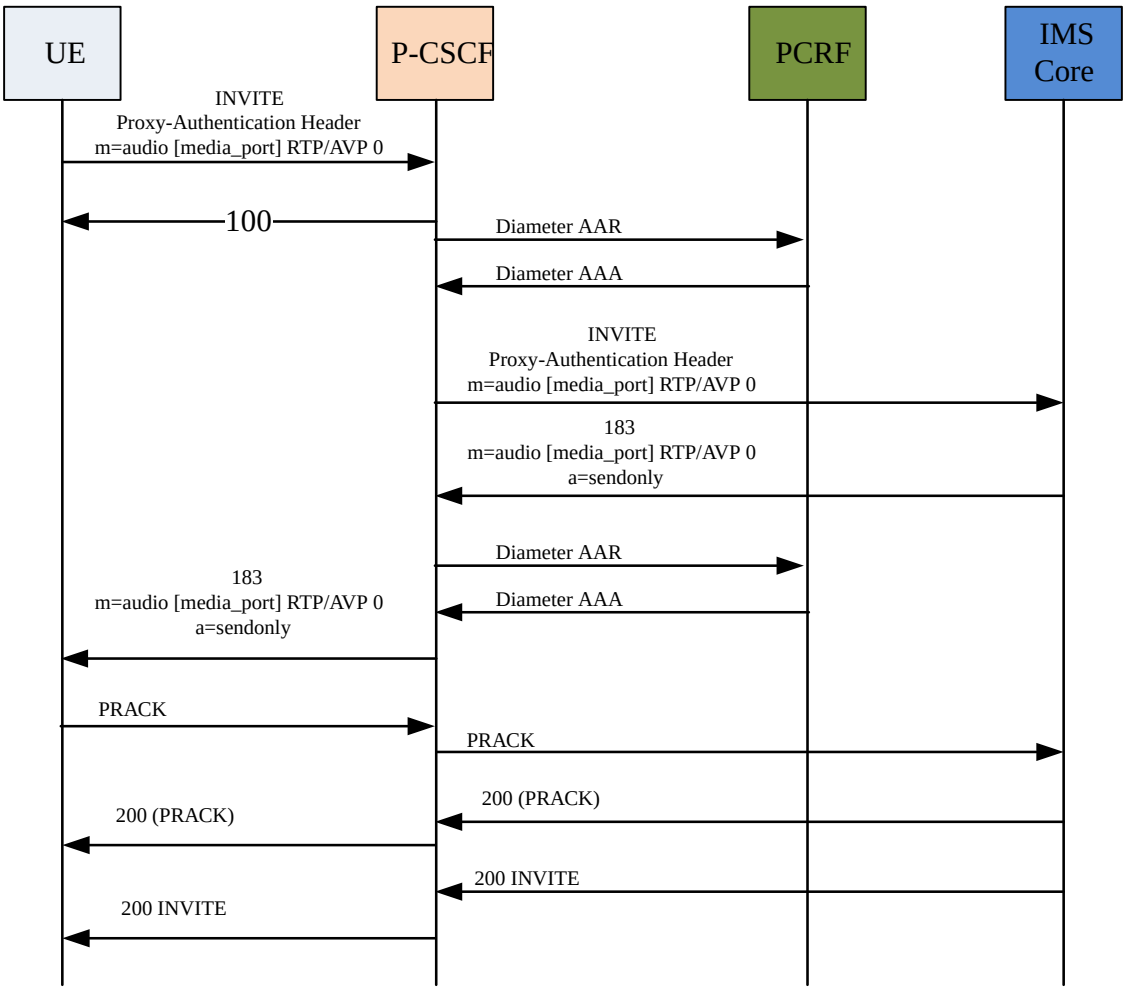
- the originating INVITE does not contain a proxy-authorization header, except for INVITE messages coming from IMS AKA users
- the codec & bandwidth provided in the m-line of an SDP response, and media IP address and port have not changed since the last SDP message
- In the MTC scenario, AAR is suppressed on reception of the originating non-IMS-AKA INVITE with no NPLI configured

AAR Optimization for Supplementary Services

In some call flows, a reINVITE or UPDATE request or response is sent to the P-CSCF and neither the bandwidth nor codec has changed. Such reINVITEs or UPDATES may include SDP offers/answers with new a=sendonly, a=recvonly, or a=inactive. These SDP parameters are directly related to the provision of Supplementary Services such as Call Hold. By default the Oracle Communications Session Border Controller does not send an AAR to the PCRF for these changes.



The Oracle Communications Session Border Controller can be configured to allow the generation of an AAR in these cases by adding the **optimized-aar=supplementary-service** option to the **ext-policy-server** configuration element. (You may also configure the option as **optimized-aar= supplementry** resulting in identical behavior.) The following call flow reflects this optimized behavior.



AAR Message Configuration

To optimize the use of AAR messages:

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. **optimize-aar**—Set this parameter to **enabled** to optimize the use of AAR messages.
4. **reserve-incomplete**—Set this parameter to **enabled** in conjunction with **optimize-aar** set to enabled for the system to not send an AAR to the PCRF if the Proxy-authorization header is absent from the INVITE.

5. Type **done** to save your configuration.

AAR Optimization with supplementary-service Configuration

This configuration sets the Oracle Communications Session Border Controller to send AAR messages to the PCRF when in a reINVITE or UPDATEs the SDP offer/answer contains changes to the a=sendrecv, a=recvonly, a=sendonly, a=inactive lines lines.

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. **options**—Set the options parameter by typing **options**, a space, **+optimized-aar=supplementary-service** to generate an AAR when a reINVITE or UPDATE contains SDP that changes an a=sendonly |recvonly | inactive parameter.

```
ORACLE(ext-policy-server)# options +optimized-aar=supplementary-service
```

> **Note:**
>
> You may also configure the option as **optimized-aar=supplementary** resulting in identical behavior.

If this option is not configured and **optimize-aar** is enabled, changes to these SDP parameters will not generate an AAR.

4. Type **done** to save your configuration.

AF-Application-Identifier AVP Generation

By default, the SBC populates the AF-Application-Identifier AVP with the hostname parameter of the external policy server object that receive's the AAR for the call. Applicable signaling may provide the SBC with the AF-Application-Identifier AVP at the Media-Component-Description level. When provided, the SBC processes and forwards further signaling based on the AF-Application Identifier provided within the Media-Component-Description AVP.

The SBC can populate the AF-Application-Identifier AVP with the IMS Communication Service Identifier (ICSI). 3GPP TS 24.173 defines the ICSI format as:

urn:urn-xxx:3gppservice.ims.icsi.mmtel

An ICSI appears in one of the following three SIP headers in an INVITE:

- P-Assured-Service
- P-Preferred-Service
- Accept-Contact

When this feature is configured, the AF-Application-Identifier AVP is inserted into an AAR and sent to the PCRF when the SBC (as P-CSCF) receives an INVITE. To configure the feature, navigate to the **sip-config** configuration element and set the **options** parameter by typing **options**, a space, and then **+get-icsi-from-sip**, as shown in the following example:

```
ORACLE(sip-config)# options +get-icsi-from-sip
```

When the SBC has an ICSI value from the access UE and the core, it uses the value received from the core. ICSI values received from the network supercede a configured default ICSI value. When the INVITE is received from the access side, ICSI values are taken from the access side until values are received from the core side. If no valid ICSI value is received in a message, and the SBC is not configured with a default-icsi value, the external policy server object's hostname is used for the ICSI value sent to the PCRF. If an INVITE does not contain SDP, no AAR is generated.

AF-Application-Identifier AVP generation occurs for access and core initiated calls.

AVP Generation on Initial INVITE from UE

The Oracle Communications Session Border Controller receives the UE's INVITE. When configured to send an AAR on receipt of the INVITE, the AF-Application-Identifier AVP is populated with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service
- Accept-Contact

If none of the above three headers are present, the default value of AF-Application-Identifier shall be used.

If the Oracle Communications Session Border Controller is not configured to send an AAR on receipt of an INVITE, the value from the 3 headers will be cached for later AF-Application-Identifier generation.

AVP Generation on response to Initial INVITE from UE

The Oracle Communications Session Border Controller receives the response (1xx or 200 OK) to the original INVITE. When configured to send an AAR on receipt of the response, the AF-Application-Identifier AVP is populated with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service
- P-Preferred-Service
- Accept-Contact header is not expected

Values obtained in this step supercede any value set from the initial INVITE step. If none of the three headers are present, the value determined in the INITIAL invite step is used for the AF-Application-Identifier AVP contents.

AVP generation on reINVITE from UE

The Oracle Communications Session Border Controller can receive a re-INVITEs from the core. The Oracle Communications Session Border Controller populates the AF-Application-Identifier AVP with one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service
- Accept-Contact

If none of the three headers are present, the default AF-Application-Identifier value is used.

Responses to the Re-Invites will be treated in the similar manner.

Examples

Example 1:

Received from access UE	Received from core	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P2”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 2:

Received from access UE	Received from core	Sent to PCRF
INVITE with no SDP	N/A	No AAR generated
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P2”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 3:

Received from access UE	Received from core	Sent to PCRF
INVITE with no SDP	N/A	No AAR generated

Received from access UE	Received from core	Sent to PCRF
N/A	200 OK with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains default AF-A-I value.
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P3”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 4:

Received from access UE	Received from core	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with no SDP	AAR with AF-A-I AVP contains “P1”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 5:

Received from access UE	Received from core	Sent to PCRF
INVITE with no SDP	N/A	No AAR generated
N/A	183 with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains ext-policy-server’s “hostname”
PRACK with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with no SDP	No AAR generated
N/A	200 OK (INVITE) with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 6:

Received from access UE	Received from core	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”

Received from access UE	Received from core	Sent to PCRF
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
N/A	re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	AAR with AF-A-I AVP contains “P3”
200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	N/A	AAR with AF-A-I AVP contains “P3”

AVP Generation on Initial INVITE from core

The Oracle Communications Session Border Controller receives the original INVITE. When configured to send an AAR on receipt of the INVITE, the AF-Application-Identifier AVP is populated with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service
- P-Preferred-Service
- Accept-Contact

If none of the above three headers are present, the default value of AF-Application-Identifier shall be used.

If the Oracle Communications Session Border Controller is not configured to send an AAR on receipt of an INVITE, the value from the 3 headers will be cached for later AF-Application-Identifier generation.

AVP Generation on response to initial INVITE from core

The Oracle Communications Session Border Controller receives the response (1xx or 200 OK) to the original INVITE. When configured to send an AAR on receipt of the response, if none of the three headers (P-Assured-Service, P-Preferred-Service, or Accept-Contact) were present in the original INVITE, the Oracle Communications Session Border Controller populates the AF-Application-Identifier AVP with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service
- Accept-Contact header is not expected

If none of the three headers are present, the value determined in the INITIAL invite step is used for the AF-Application-Identifier AVP contents.

AVP generation on reINVITE from core

The Oracle Communications Session Border Controller can receive re-INVITEs from the core. The Oracle Communications Session Border Controller populates the AF-Application-Identifier AVP with one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service

- Accept-Contact

If none of the three headers are present, the default AF-Application-Identifier value is used.

Responses to the Re-Invites will be treated in the similar manner.

Example 1:

Received from core	Received from access UE	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P1”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P3”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 2:

Received from core	Received from access UE	Sent to PCRF
INVITE with no SDP	N/A	No AAR generated
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P3”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 3:

Received from core	Received from access UE	Sent to PCRF
INVITE with no SDP	N/A	No AAR generated
N/A	200 OK with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains ext-policy-server hostname.
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P3”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 4:

Received from core	Received from access UE	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with no SDP	AAR with AF-A-I AVP contains “P1”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	N/A	AAR with AF-A-I AVP contains “P3”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 5:

Received from core	Received from access UE	Sent to PCRF
INVITE with no SDP	N/A	No AAR generated
N/A	183 with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains ext-policy-server’s “hostname”
PRACK with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with no SDP	No AAR generated
N/A	200 OK (INVITE) with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P1”

Example 6:

Received from core	Received from access UE	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”	N/A	AAR with AF-A-I AVP contains “P1”
N/A	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P1”
N/A	re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	AAR with AF-A-I AVP contains “P1”
200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	N/A	AAR with AF-A-I AVP contains “P4”

Diameter: CLF

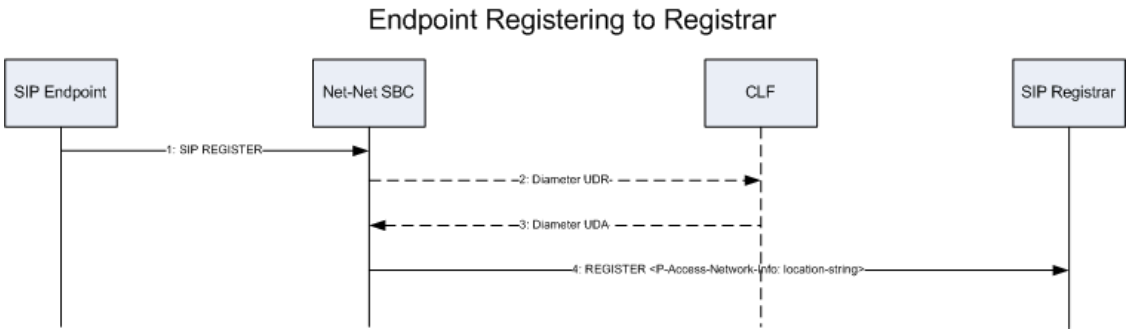
The Oracle Communications Session Border Controller supports the e2 interface over a Diameter connection acting as a P-CSCF communicating with a CLF. The application ID field must be set to 16777231 to run the e2 reference point.

A Connectivity Location Function (CLF) maintains mappings between endpoints with dynamically assigned IP addresses and their physical location. The Oracle Communications Session Border Controller, acting as a P-CSCF, is the intermediary device between a registering endpoint and a CLF. The CLF thus validates and tags a registering endpoint, and the Oracle Communications Session Border Controller applies the CLF's actions.

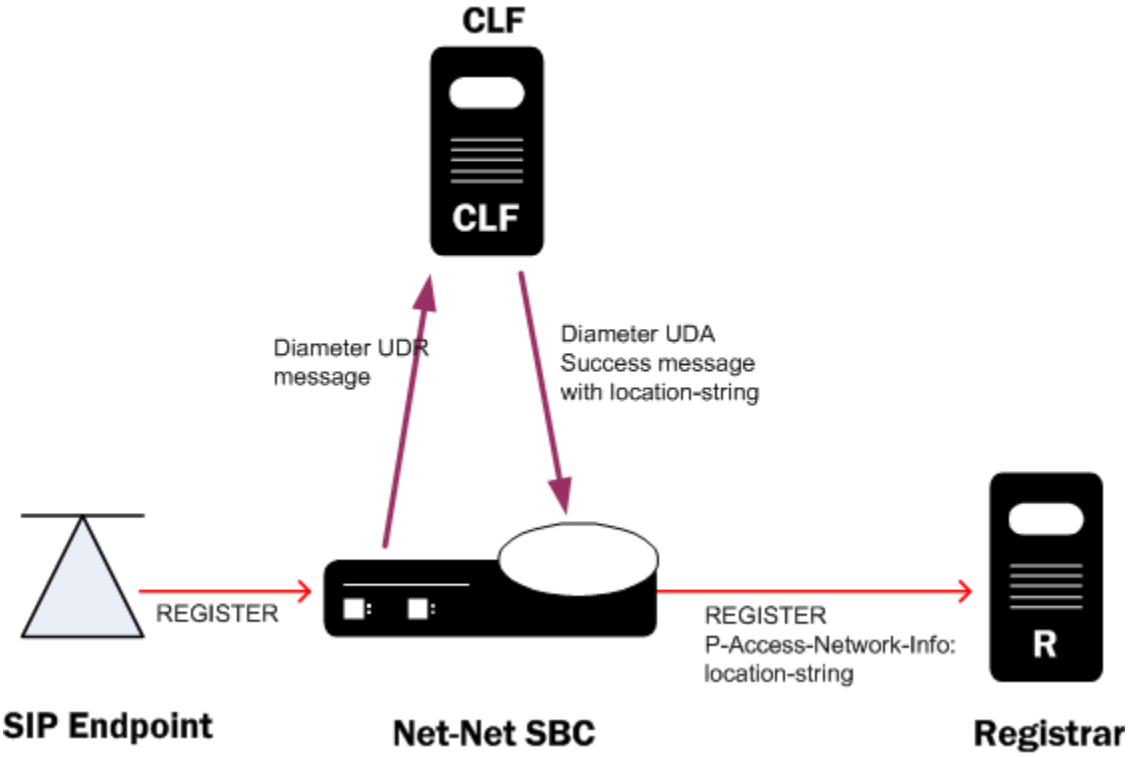
CLF Behavior

The Oracle Communications Session Border Controller and a CLF only interact with each other when an endpoint registers or re-registers. The Oracle Communications Session Border Controller, acting as the P-CSCF, is the first SIP device that the REGISTER message reaches. Upon receiving the REGISTER message(1), the Oracle Communications Session Border Controller queries the CLF using the Diameter protocol. The endpoint's (public) IP address and port, and the Oracle Communications Session Border Controller's IP information are sent to the CLF in a Diameter User-Data-Request (UDR) message(2).

The CLF responds to the Oracle Communications Session Border Controller with a Diameter User-Data-Answer (UDA) message(3). If the request is approved, then the CLF also sends a location-string value to be inserted in one of the SIP headers. The Oracle Communications Session Border Controller inserts a P-Access-Network-Info header containing the location-string into the incoming REGISTER message and forwards this message(4) to the SIP registrar/I/S-CSCF.



The Oracle Communications Session Border Controller inserts this P-Access-Network-Info header into all subsequent SIP messages from this endpoint as they are forwarded into the core network. The P-Access-Network-Info header is inserted into all SIP requests and responses except for ACK and CANCEL messages. For all boundaries where SIP messages pass from trusted to untrusted SIP interfaces or session agents, the Oracle Communications Session Border Controller will strip out the P-Access-Network-Info header as expected.



If the CLF responds with a Reject UDA message, the Oracle Communications Session Border Controller rejects the registration, and sends a 503 - Service Unavailable message back to the registering endpoint. In this way, the CLF can be used for admission control.

The Oracle Communications Session Border Controller communicates with the CLF solely for retrieving location information from the CLF, and not for notifying the CLF about an endpoint's registration state or activity. When an endpoint's registration ends, either through a normal expiration, getting rejected by the registrar, or through specific de-registering or error conditions, the Oracle Communications Session Border Controller deletes the locally cached registration location string. The Oracle Communications Session Border Controller does not inform the CLF about any registrations that have been deleted.

P-Access-Network-Info Header Handling

The P-Access-Network-Info header is created and populated according to the following rules:

1. If the CLF returns an Accept UDA message with a location string, the Oracle Communications Session Border Controller inserts the location string into a P-Access-Network-Info header in the outgoing REGISTER message.
2. If the CLF returns an Accept UDA message without a location string, the Oracle Communications Session Border Controller inserts the configured default string into a P-Access-Network-Info header in the outgoing REGISTER message.
3. If the CLF returns an Accept UDA message without a location string and no location string is configured on Oracle Communications Session Border Controller, the outgoing REGISTER message is forwarded out of the Oracle Communications Session Border Controller, but no P-Access-Network-Info header is created for the REGISTER message.

P-CSCF PANI Enhancements

The Oracle Communications Session Border Controller always adds PANI headers to the egress SIP message with the value set to either:

- The location of information recieved from the remote CLF agent
- The default-location-string

When you enable **allow-pani-for-trusted-only** in the SIP config configuration element, the Oracle Communications Session Border Controller only forwards PANI headers to and from trusted sources or if access-info does not have network-provided in the header. A trusted domain is determined by the trust-level of the corresponding realm or SIP interface.

CLF Re-registration

The Oracle Communications Session Border Controller will send a new UDR message to the CLF to request a new location string if any of the following events occur:

1. The endpoint's contact address changes.
2. The SIP Register message's Call-ID header changes.
3. The endpoint's public IP Address or UDP port changes.
4. The endpoint connects to a different SIP interface, port, or realm on the Oracle Communications Session Border Controller than it did in the initial REGISTER message.
5. The registration expires in the Oracle Communications Session Border Controller's registration cache.

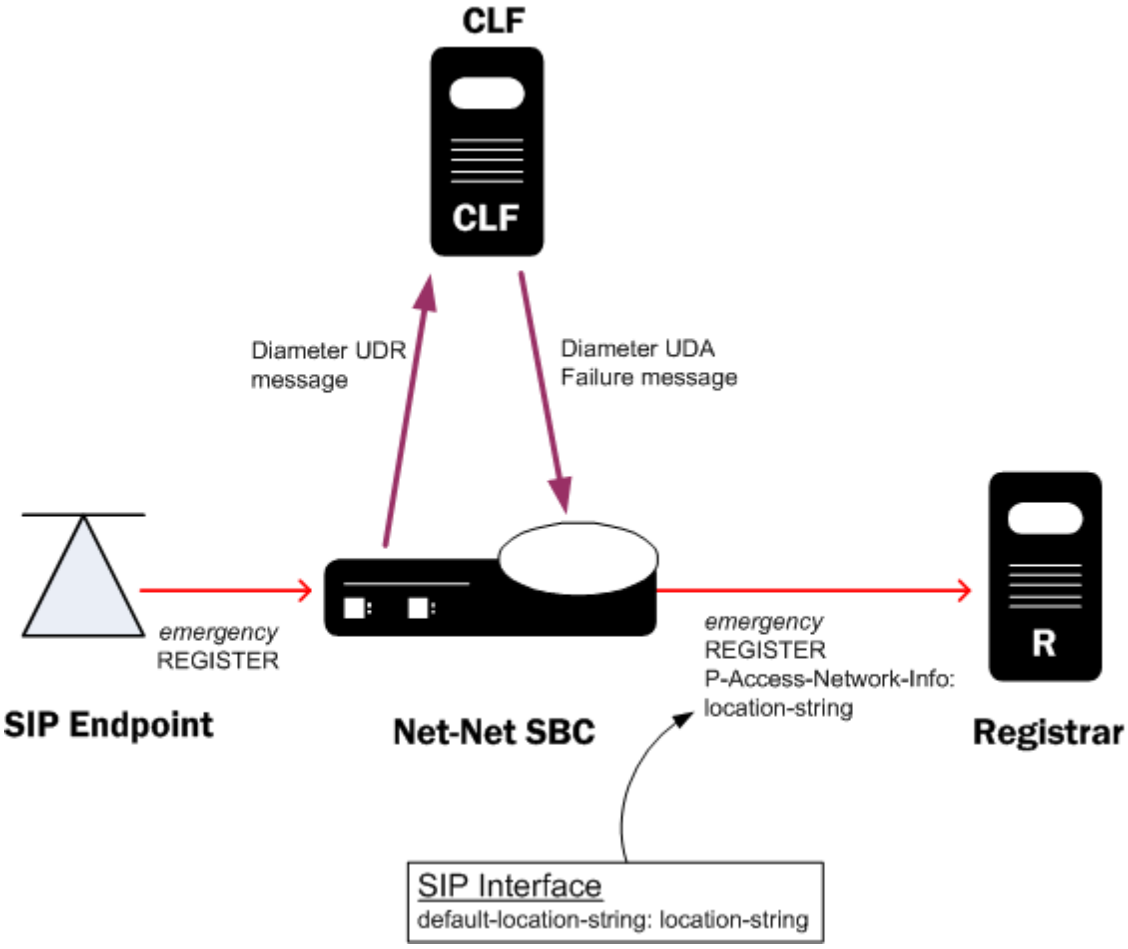
CLF Failures

If a Diameter connection fails, the Oracle Communications Session Border Controller will continually try to re-establish the connection. Endpoints that are already registered will stay registered unless they timeout or if the registrar rejects their refreshes. When the Diameter connection has not been established, and an endpoint registers on a SIP interface that is configured to use CLF, the Oracle Communications Session Border Controller forwards new REGISTER messages to the registrar using the default location string.

CLF Emergency Call Handling

The Oracle Communications Session Border Controller allows emergency calls into the network even if the endpoint that places the emergency call is not registered. In the expected fashion, the Oracle Communications Session Border Controller will query the CLF first for an incoming emergency call sourced from an unregistered endpoint. If the CLF response is successful, then the Oracle Communications Session Border Controller will insert the string returned from the CLF into a P-Access-Network-Info header, and insert this header into the emergency call's REGISTER message. If no location string is returned with a successful CLF response, the default location string is inserted into P-Access-Network-Info header.

If the CLF's response is to reject the emergency call, the Oracle Communications Session Border Controller will insert the configured default location string into the P-Access-Network-Info header and forward the emergency call's REGISTER message toward the registrar. For emergency calls where the endpoint has already successfully registered, the call will be routed into the network using the expected methods for emergency call routing.



If the Diameter connection to the CLF is down, emergency calls from un-registered endpoints are still allowed into the network using the default string inserted into the emergency messages.

CLF Diameter e2 Error Handling

When the Oracle Communications Session Border Controller receives an error, as defined in RFC 3588, in a UDA message from a CLF, it replies to the UA with a 503 Service Unavailable message. There are exceptions for certain values embedded in either the Result-Code AVP or Experimental Result-Code AVP.

Result-Code AVP

If the CLF sends a Result-Code AVP in a UDA message with a value of DIAMETER_UNABLE_TO_COMPLY (5012), the Oracle Communications Session Border Controller forwards the REGISTER message using the default configured location string in the P-Access-Network-Info header. If the configured default string is left blank the Oracle Communications Session Border Controller forwards the SIP REGISTER message without the P-Access-Network-Info header.

Experimental-Result-Code AVP

If the CLF sends a grouped Experimental Result AVP in a UDA message that is not successful, as per RFC 3588, the Oracle Communications Session Border Controller performs the following actions with respect to receipt of these two result code values:

- DIAMETER_ERROR_USER_UNKNOWN(5001): The Oracle Communications Session Border Controller inserts the configured default-string in for the P-Access-Network-Info header when forwarding the SIP message.
- DIAMETER_ USER_DATA_NOT_AVAILBLE (4100): The Oracle Communications Session Border Controller inserts the configured default-string in for the P-Access-Network-Info header when forwarding the SIP message.

For both cases, if the configured default string is left blank the Oracle Communications Session Border Controller forwards the SIP REGISTER message without the P-Access-Network-Info header. For all other failure result codes, the Register is rejected.

Diameter CLF e2 Error Bypass

When a CLF returns error messages i.e., any other Diameter answer than DIAMETER_SUCCESS, you can configure the Oracle Communications Session Border Controller to override the default behavior and still forward REGISTER messages with P-Asserted-Network-Id header containing the **default-location-string** parameter value. If the **default-location-string** parameter value is empty, the REGISTER message is forwarded without the P-Access-Network-Info header. This error bypass feature is configured by setting the **permit-on-reject** parameter to **enabled** for an external policy server configuration element with application-mode set to e2.

CLF-only AVPs

Globally-Unique-Address AVP

When endpoints registering through the Oracle Communications Session Border Controller reside in nested realms, the Oracle Communications Session Border Controller allows you to set the realm that appears in the Globally-Unique-Address AVP in Diameter UDR messages destined for a CLF.

The **ingress-realm-location** parameter in the external policy server configuration specifies whether to use the realm on which a signaling message arrived, or to use that realm's parent. If you choose to use the parent realm, the Oracle Communications Session Border Controller uses the one associated with the SIP interface on which the REGISTER request arrived.

e2 Interface

You can configure a value to be sent in the Address-Realm AVP (communicated in the Globally-Unique-Address AVP) for the Diameter e2 interface. This AVP is sent on a per-realm basis in the Location Information Query (UDR) query the Oracle Communications Session Border Controller (as a P-CSCF) sends to the CLF.

The CLF maintains details about IP connectivity access sessions associated with user equipment connected in the network. This CLF supports the standardized Diameter e2 interface with an Application Function; from it, the application function (i.e., the Oracle Communications Session Border Controller in the role of P-CSCF) retrieves location

information and other data related to the access session. To do so, the AF sends a UDR containing Global-Unique-Address and Requested Information attributes. Then the CLF returns a Location Information Response (UDA) containing either a success result code with location information or an error result code.

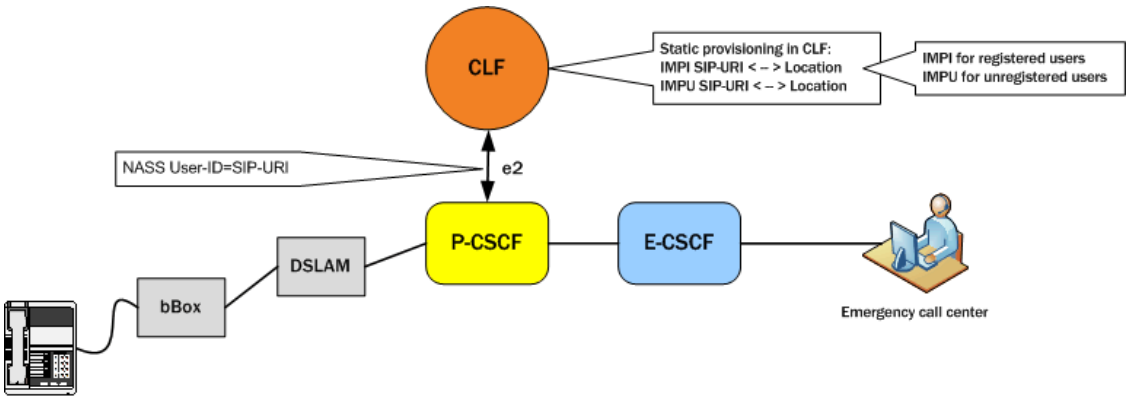
In the UDR's Global-Unique-Address, the Oracle Communications Session Border Controller currently supports the mandatory parameters: the Framed-IP-Address AVP and the Address-Realm AVP. The address realm AVP is the realm address in FQDN form, populated based on the realm on which a REGISTRATION arrived or the SIP interface. With nested realms configured, using a realm for this value can quickly become complicated.

You can configure the value sent in the Address-Realm AVP on a per-realm basis. You set the external policy server's **ingress-realm-location** parameter to the **diam-address-realm** value, pointing the Oracle Communications Session Border Controller to the associated realm from which it will learn Address-Realm AVP information. This access realm (or child realm, the realm on which enter registration came in) has its **diam-e2-address-realm** value set to the value with which to populate the Address-Realm AVP for the outgoing UDR message.

CLF e2 Interface User-Name AVP Support

In compliance with ETSI ES 283 -35 V1.2.1, the Oracle Communications Session Border Controller's e2 interface can query the CLF using the SIP endpoint's IP address or its NASS User-ID. The system's e2 interface uses the uses the SIP-URI to query the CLF by including the User-Name AVP in the User Data Request (UDR). The CLF can then furnish the P-CSCF (i.e., the Oracle Communications Session Border Controller) with the INSEE code in the Location-Information AVP in its User Data Answer (UDA).

This diagram shows how this capability works. The Oracle Communications Session Border Controller acts as the P-CSCF.



- Registering users—When it receives a registration request, the Oracle Communications Session Border Controller checks the incoming SIP interface to determine CLF use. If CLF use is unnecessary, the Oracle Communications Session Border Controller forwards the registration message to its destination.
  When CLF is required, the Oracle Communications Session Border Controller selects the AVPs to send the CLF, including the User-Name AVP before sending it a UDR. A **none** setting for this parameter means the Oracle Communications Session Border Controller does not include the User-Name AVP in any UDRs. The Oracle Communications Session Border Controller adds the User-Name AVP to the UDR if the **user-name-mode** parameter in the external policy server configuration is set to:
  - **endpoint-id**—IP address of the registering endpoint is sent as the payload for the User-Name AVP

  - **public-id**—SIP-URI portion of the TO header from the register message is sent as the payload for the User-Name AVP
  - **auth-user**—Username attribute of the Authorization header from the register is sent as the payload for the User-Name AVP; if there is no authorization header, the Oracle Communications Session Border Controller will not consult the CLF and will forward the registration message

- Unregistered users—When it receives an INVITE request, the Oracle Communications Session Border Controller checks the incoming SIP interface to determine if it should use an external policy server. If it does not need to use an external policy server, the Oracle Communications Session Border Controller forwards the INVITE message to its destination.
  When the Oracle Communications Session Border Controller does need to use an external policy server, it also checks to determine if the INVITE is in a registration and if location data in the registration cache is avaible for that endpoint. These requirements being met, the Oracle Communications Session Border Controller inserts the P-Access-Network-Info header with the location string into the INVITE it forwards to the destination. If these requirements are not met, the Oracle Communications Session Border Controller consults the CLF before forwarding the INVITE. The following describe the impact of the user-name-mode setting in such instance:
  - **endpoint-id**—IP address of the endpoint that issued the INVITE is sent as the payload for the User-Name AVP
  - **public-id and auth-user**—SIP-URI portion of the INVITE is sent as the payload for the User-Name AVP:
    With a P-Asserted-Id header present in the INVITE, the Oracle Communications Session Border Controller uses the first PAID header (if there are multiple PAID headers)

    Without a P-Asserted-Id header present in the INVITE's P-Preferred-Identity header, the Oracle Communications Session Border Controller uses the first PPI (if there are multiple PPI headers)

    With neither P-Asserted-Id nor P-Preferred-Identity header present, the Oracle Communications Session Border Controller uses the From header

HA Functionality

The location strings generated by the CLF are replicated on the standby Oracle Communications Session Border Controller in an HA pair. This is required so that a Oracle Communications Session Border Controller in an HA pair can instantly continue processing calls using the previously learned CLF information.

Configuring Diameter-based CLF

SIP Interface Configuration

In the following configuration examples, we assume that your baseline configuration passes SIP traffic, with the Oracle Communications Session Border Controller in the role of an Access Oracle Communications Session Border Controller. In this example, you will configure additions to the ream configuration and the new external policy server configuration.

SIP Interface Configuration for CLF Support

To configure the SIP interface configuration for CLF support:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# sip-interface
ORACLE(sip-interface)#
```

4. Type **select** and the number of the pre-configured sip interface you want to configure for CLF. This should be the ingress SIP interface for

```
ORACLE(sip-interface)# select 1
ORACLE(sip-interface)#
```

5. **ext-policy-svr**—Set this parameter to the same name as the External Policy Server configured that you configured for the CLF server.
6. **default-location-string**—Set this parameter to the default location string you want inserted into a P-Access-Network-Info header for when the CLF server does not return a unique location string.
7. Save your work using the ACLI **done** command.

External Policy Server Configuration

To configure the external policy server for use with a CLF:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the media-related configurations.

```
ORACLE(configure)# media-manager
```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. **name**—Set this parameter to an applicable name for this CLF instance of the external policy server. The value of this parameter will be entered in the SIP interface configuration element to reference this CLF.

5. **state**—Set this parameter to **enabled** to enable this CLF. The default value is **enabled**. The valid values are:
   - enabled | disabled
6. **operation-type**—Set this parameter to **admission-control** for the Oracle Communications Session Border Controller to communicate with a CLF. The default value is **disabled**.
7. **protocol**—Set this parameter to **DIAMETER** to connect with a CLF via the DIAMETER protocol.
8. **address**—Set this parameter to the IP address or FQDN of the CLF.
9. **port**—Set this parameter to the port which the CLF uses for Diameter transactions. Port 3868 is the default Diameter port. (The default value is **80**.) The valid range is:
   - Minimum—0
   - Maximum—65535
10. **realm**—Set this parameter to the realm where the CLF exists.
11. **permit-conn-down**—Enable or disable the Oracle Communications Session Border Controller's ability to permit calls if there is no connection to the external policy server. The default value is **disabled**. The valid values are:
    - enabled | disabled
12. **permit-on-reject**—Change this parameter to enabled if you want the Oracle Communications Session Border Controller to forward the session on at a best-effort. Leave this parameter set to disabled (default), if you want the Oracle Communications Session Border Controller to deny the session on attempts to revert to the previously-requested bandwidth. (The default value is disabled.) Valid values are:
    - enabled | disabled
13. **disconnect-on-timeout—**Disable this parameter to prevent timeouts triggered by Gate-Set or Gate-Delete message sequences between the Oracle Communications Session Border Controller and a policy server from tearing down their connection. Retaining the default (enabled) allows all timeouts to tear down and re-establish the TCP connection. The valid values are:
    - **Default enabled**
    - **enabled | disabled**
14. **product-name**—Enter text string that describes the vendor-assigned name for the CLF. This parameter is required.
15. **application-mode**—Enter the type of interface you want to use. Your choices are: **Rq**, **Rx**, **Gq**, **e2**, and **none**.
16. **application-id**—Enter a numeric application ID that describes the interface used to communicate with the CLF. The default value is zero (**0**). For the e2 CLF reference point, the application id is **16777231**.
17. **framed-ip-addr-encoding**—Enter the format of the Frame-IP-Address (AVP 8) value in Diameter messages. The default value is **octet-string**. The valid values are:
    - **ascii-string**—Example: 192.168.10.1
    - **octet-string**—Example: 0xC0A80A01
18. **dest-realm-format**—Enter the format you want to use for the Destination-Realm AVP. The default value is **user_with_realm**. The valid values are:
    - user_with_realm | user_only | realm_only

19. **ingress-realm-location**—Set this parameter to configure the child realm or its parent for the Address-Realm in the Globally-Unique-Address AVP in Diameter UDR messages that the Oracle Communications Session Border Controller sends to the policy server. There are two choices:
    - **realm-in** (default)—This setting means that the Oracle Communications Session Border Controller will use the same realm on which the REGISTRATION request arrived.
    - **sip-interface**—This setting means that the Oracle Communications Session Border Controller will use the realm associated with the SIP interface on which the REGISTRATION request arrived.
    - **diam-address-realm**—For the e2 interface, this value enables configurable Address-Realm AVPs. This setting points the system to the associated realm from which it will learn Address-Realm AVP information. The default is realm-in.

      Remember to set the diam-e2-address-realm parameter in the realm this parameter points to.
20. **user-name-mode**—Determines how the User-Name AVP is constructed. Used primarily with e2 based CLF functionality. There are four choices.
    - none—the system does not include the User-Name AVP in any UDRs (default)
    - endpoint-ip—IP address of the registering endpoint is sent as the payload for the User-Name AVP
    - public-id—SIP-URI portion of the TO header from the register message is sent as the payload for the User-Name AVP
    - auth-user—Username attribute of the Authorization header from the register is sent as the payload for the User-Name AVP; if there is no authorization header, the Oracle Communications Session Border Controller will not consult the CLF and will forward the registration message
21. **domain-name-suffix**—Enter the suffix you want to use for Origin-Realm and Origin-Host AVPs that have a payload string constructed as a domain name Your value can be any string, to which the Oracle Communications Session Border Controller will prepend with a dot if you do not include one. The default value is **.com**.
22. **gate-spec-mask**—With this parameter, you can configure the Oracle Communications Session Border Controller to use a mask comprised entirely of zeros (0). The default value is 255. This parameter sets the value to use for the COPs pkt-mm-3 interface. This interface maintains a persistent TCP connection to the external policy server, even without repsonses to requests for bandwidth. This permits calls to traverse the Oracle Communications Session Border Controller even though the external policy server either fails to respond, or rejects the session.
    - Default 255
    - Values Min: 0 / Max: 255
23. **allow-srv-proxy**—Set to enabled in order to include the proxy bit in the header. The presense of the proxy bit allows the Oracle Communications Session Border Controller to tell the external policy server whether it wants the main server to handle the Diameter message, or if it is okay to proxy it to another server on the network (disabled). The default is enabled. The valid values are:
    - enabled | disabled
24. **reserve-incomplete**—Set this parameter to **enabled** if you want the Oracle Communications Session Border Controller to send a message to the CLF that does not

include the endpoint's true port number. A value of 0 will be used for the port number. The default value is **enabled**. The valid values are:

- enabled | disabled

25. **watchdog-ka-timer—**Enter the interval in seconds of Oracle Communications Session Border Controller-initiated watchdog/keep-alive messages. Valid values range between 0 and 65535 seconds.
26. **trans-expires**—Set the amount of time, in seconds, that the Oracle Communications Session Border Controller waits before performing an expiration if a Diameter base protocol transaction does not occur. The default value is 15 seconds. Valid values range between 1 and 15.
27. Save your work using the ACLI **done** command.

CLF Debugging

A new argument has been added to the show command for viewing CLF statistics. From the user prompt, type **show ext-clf-svr**.

```
ORACLE# show ext-clf-svr
14:17:14-114
EBM Status                  -- Period -- -------- Lifetime --------
                  Active    High   Total       Total  PerMax    High
Client Trans           0       0       0           0       0       0
Server Trans           0       0       0           0       0       0
Sockets                0       0       0           0       0       0
Connections            0       0       0           0       0       0
                           ---- Lifetime ----
                     Recent      Total  PerMax
CLF Requests              0          0       0
CLF Admits                0          0       0
CLF Errors                0          0       0
CLF Rejects               0          0       0
CLF Expires               0          0       0
CLFD Errors               0          0       0
```

Retrieve the CLF statistics in the log.embd file.

E2 CLF Configurable Timeout

You can configure a transaction timer value for each external policy server based on the round-trip and response times for each specific policy server. Defining this transaction timer value allows you to avoid situations when policy/DIAMETER servers fail to respond to the Oracle Communications Session Border Controller 's requests, ensuring that transactions proceed in the desired amount of time and avoiding undesirable delays in set-up times.

The time you set for the DIAMETER CLF must be from 1 second to 15 seconds. For purposes of backward compatibility, 15 seconds is the default. It is recommended that you set the timer at 6 seconds or greater in accordance with the smallest DIAMETER Watchdog Request/ Answer (DWR/DWA) per RFC 3588.

Activating a Configuration with the E2 CLF Timeout Defined

When you define a value for the E2 CLF timeout and activate your configuration, the change in time will only affect new client transactions. Previously existing client transactions will use the value in effect prior to your having activated the configuration.

E2 CLF Timeout Configuration

To define the E2 CLF timeout for an external policy server:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **ext-policy-server** and press Enter.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. **trans-expires**—Enter the time (in seconds) for the E2 CLF timeout, a transaction timer value for each external policy server based on the round-trip and response times for each specific policy server. The default for this parameter is 15 seconds; your entry must be between 1 and 15.
5. Save your work.

Diameter Policy Server High Availability

External Policy Server HA Cluster

When using TCP transport, the Oracle Communications Session Border Controller (SBC) can provide external policy server redundancy through a combination of multiple servers being returned in one FQDN query and maintaining state of these servers.

> **Note:**
>
> Server configurations using SCTP transport (for the Rx interface) use SCTP multihoming configurations on the SBC and/or as provided by the server to establish redundancy.

When multiple IP addresses are returned in a response to a DNS query for Diameter-based external policy servers, the SBC assembles the IP addresses into an HA cluster which provides redundancy for Diameter-based applications. This feature is enabled by configuring the external policy server's **address** parameter with an FQDN.

The number of servers maintained in the HA cluster is configured with the **max connections** parameter. Thus if the **max connections** parameter is set to 3, the SBC maintains 1 active external policy server with 2 back up servers.

Standby Server Prioritization

When using TCP transport, the Oracle Communications Session Border Controller looks at the priority and weight fields in the DNS response to create the preferred order of the primary, secondary, tertiary, and quartiary Policy Servers. These 4 addresses are known as the top-level PSs in the cluster.

The Oracle Communications Session Border Controller uses the priority and weights according to RFC 2782. However, weights only apply when multiple servers share the same priority value. If the priorities are different, then the Oracle Communications Session Border Controller will not use weights. If the priorities are the same, then weight of the two (or more) contested servers is used to determine which one to use.

Server States

An HA cluster can contain up to 4 Policy Servers, where the TCP/Diameter connection is established and monitored. Diameter session traffic is only sent to the active policy server in the cluster. The policy servers exist in the following states:

- active-TCP and Diameter connection established. Oracle Communications Session Border Controller using this server for policy decisions. The policy server with the highest priority/ weight begins in this state.
- standby- TCP and Diameter connection established. Server in standby mode.
- inactive - Diameter connection not successfully established. Oracle Communications Session Border Controller tries to reconnect to inactive servers.

HA Cluster Refresh (TCP Transport)

The Oracle Communications Session Border Controller sends follow-up SRV queries to the DNS server to refresh the list of available policy servers in the cluster in the following instances:

- DNS cache expires after the TTL is exceeded
- a new policy server with FQDN for an address is configured, saved, and activated on the Oracle Communications Session Border Controller
- after an SPU switchover, the newly active SPU performs a DNS query

When the Oracle Communications Session Border Controller re-queries the DNS server for Diameter external policy servers, the cluster is refreshed with the new/changed servers. Policy server priority is also reconfigured based on newly returned priorities and weights. Upon a cluster refresh, the Oracle Communications Session Border Controller:

- closes connections with standby policy servers that are no longer in the cluster
- creates connections with policy servers which are new in the set

If the currently active policy server remains a member of the cluster after a refresh, it remains active even if its priority has changed. If the active-before-the-refresh policy server is not a member of the cluster after a DNS refresh, the Oracle Communications Session Border Controller gracefully closes the connection to this server. The Oracle Communications Session Border Controller then installs the highest priority server as the active policy server

DNS Failure

If the Oracle Communications Session Border Controller fails to get a response from the DNS server or does not receive at least one IP address in the SRV RR, it continues to send SRV queries periodically, starting with 5 seconds and doubling the interval for every sequential failure, until it receives a valid response. While waiting for a successful DNS response, the Oracle Communications Session Border Controller uses the existing Diameter servers in the DNS cache.

Policy Server Failover

When configured for TCP transport, the active external policy server fails over to the highest priority standby server when:

- the TCP connection is closed due to a RST or FIN
- the Diameter connection (CER/CEA exchange) could not be established
- a configured number of consecutive Diameter message timeouts occur. This number is configured in the **max timeouts** parameter. The max timeouts parameter refers to all Diameter messages except for the following:
- You can configure Diameter keepalive message time-outs separately from all other Diameter messages by setting **watchdog ka timer** in the external policy server configuration element. This will failover the active policy server based on a timeout value for DWR messages.
- You can configure Diameter STR message time-outs separately from all other Diameter messages by setting the **str-retry=**<timeout number> option in the external policy server configuration element. This will failover the active policy server based on a unique timeout value for STR messages.

If the Oracle Communications Session Border Controller sends an AAR/STR message to the active policy server and then switches to a different policy server, any new Diameter messages related to that session are sent to the same policy server as long as it is not inactive. If that server becomes inactive, messages will be sent to the new policy server. The new policy server however will not recognize the sessionID and discard the request.

External Policy Server High Availability Configuration

To configure an **ext-policy-server** that uses TCP transport for high availability clustering:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

4. **address**—Enter the IP address or FQDN of the external policy server. To use external policy server redundancy with TCP transport, you must enter the address as an FQDN.
5. **port**—If you configure this parameter, it will override the port returned in a DNS reply.
6. **max-connections**—Set the number of servers to maintain in an external policy server cluster. Valid values range from 1 - 20.
7. **srv-selection-strategy**—Leave this parameter at its default, Failover setting.
8. **max-timeouts**—Set the number of request timeouts before the Oracle Communications Session Border Controller sets this external policy server to inactive. You can separately set the number of DWR timeouts that trigger a server to be inactive as an option. Valid values range from 0 - 200.
9. Save and activate your configuration.

Redundancy for Rx Servers over SCTP

The Oracle Communications Session Border Controller (SBC) uses multihomed IPs advertised in the SCTP handshake and the IP configured in the remote-multi-addr-list parameter to establish Rx server redundancy. You configure multi-homing within the **ext-policy-server** configuration and the applicable timing, on a global basis, in the **system-config**.

See Configuring the Rx Interface for SCTP for instructions on configuring multi-homing for an external policy server. See Configuring SCTP Support for SIP for global SCTP configuration including timing settings.

You must have a thorough understanding of network and the routing path to the server prior to configuration to achieve connectivity and redundancy. You configure the **ext-policy-server**'s primary IP address and multi-homing addresses based on this understanding. The SBC must be able to establish paths between it and the **ext-policy-server**'s address as well as between the multi-homing addresses via separate **network-interface**s and unique routes within the same realm.

> **Note:**
>
> Multi-homing configuration fields accept IPv4 and IPv6 IP addresses, but not FQDNs.

If you configure an FQDN for an **ext-policy-server address**, you must have also configured a DNS server on the realm's first **network-interface**. When triggered to connect to these policy servers, the SBC sends the DNS queries on that **network-interface**.

The SBC connects to servers using the following steps:

1. The SBC attempts to establish a connection between itself and the **ext-policy-server**'s configured **address** through the first **network-interface** listed in the realm configuration. If you have configured an FQDN as an **address**, the SBC performs a DNS lookup and uses the first address provided as the server address.
2. If the first connection attempt fails and you have not configured the **remote-multi-addr-list** parameter with at least one address, the connection to the server fails.
3. If you have configured multi-homing addresses and the first connection attempt fails, the SBC attempts to establish a connection with the IP you configured in the **remote-multi-addr-list** parameter. This IP can be reachable through any of the configured **network-interface**s present in the realm.

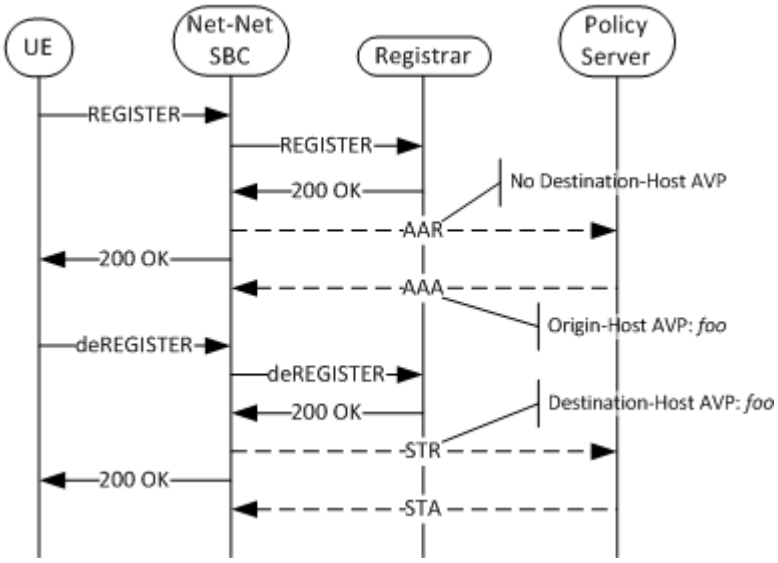
4. If the SBC cannot determine a route, it tries to use a route between the local primary IP and the default gateway.
5. During the init and init-ack, the remote agent may advertise additional multi-homing address. If it receives these additional addresses, the SBC can use these addresses as fail-over addresses should the existing connection subsequently fail.

Diameter Multi-tiered Policy Server Support

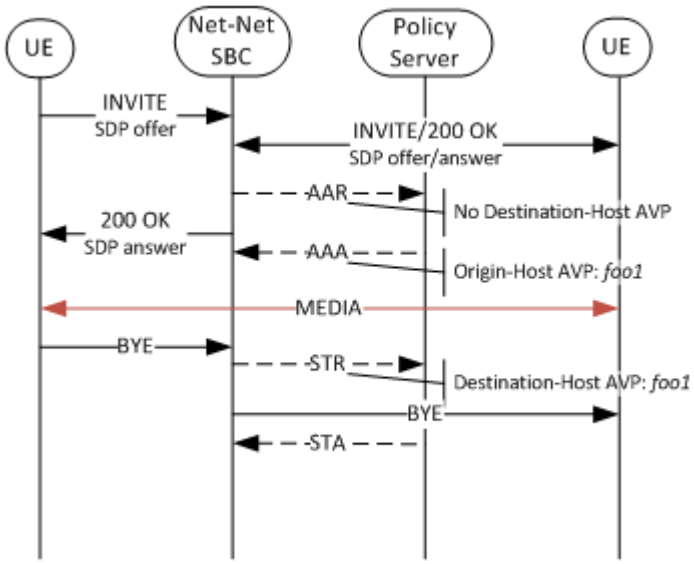
When accepting a call in a 2 tiered policy server environment, theOracle Communications Session Border Controller queries the Tier-1 policy server, running in stateless mode, for AAR-type requests. The Tier-1 PS, identifies an appropriate Tier-2 policy server for every incoming Diameter session. The mapping of session-ID to Tier-2 PS is performed between the Tier-1 and Tier-2 policy servers. This architecture is used to increase overall performance.

Two-tiered policy support is implemented per user registration and per call. To act in the P-CSCF role with a stateless Tier-1 policy server, the system recalls the Origin-Host AVP value returned in the AAA answer response to the initial Diameter AAR request. The Origin-Host-AVP value is then inserted into the Destination- Host AVP in subsequent Diameter messages (AAR, STR) corresponding to that user registration or call. Set the cache-dest-host parameter to enabled in the ext-policyserver configuration element to enable this feature.

The following call flow is an example two tiered PS support for user registration. While the initial AAR's Destination-Host AVP is empty, the system inserts the value of the received Origin-Host AVP into the registration-terminating STR.



The following call flow is an example of a two tiered PS support for a call. If this call occurs within the scope of the previous registration, the value inserted into the Destination-Host AVP will still be unique per session. Therefore, it is not the value learned from the AAA in the REGISTER, previously illustrated; it is the value learned from the AAA responding to the INVITE setting up the call.



In the case of a reINVITE changing an aspect of the call like the codec use, foo1 is inserted into the AAR's Destination-Host AVP. If the Origin-Host AVP value, as received by the system, changes mid session for a call, all subsequent Destination-Host AVP values are updated with the change.

Diameter Multi-tiered Policy Server Support

To configure Multi-tiered policy server support:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type media-manager and press Enter to access the media-related configurations.

   ```
   ACMEPACKET(configure)# media-manager
   ```

3. Type ext-policy-server and press Enter.

   The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(media-manager)# ext-policy-server
   ACMEPACKET(ext-policy-server)#
   ```

4. cache-dest-host—Set this parameter to enabled for the system to participate as a P-CSCF in a multi-tiered policy server environment.
5. Save and activate your configuration.

Diameter Message Manipulations

The Oracle Communications Session Border Controller can perform manipulations on all grouped and non-grouped AVPs. This is referred to as Diameter Manipulation Rules (DMR). A message manipulation is the ability to search for a predefined string within an AVP and then replace it with another value. This is similar to the Oracle Communications Session Border Controller's header manipulation rules functionality.

A diameter manipulation configuration element is defined by a name parameter. You can optionally add a description field to the diameter manipulation. Within each diameter manipulation you can configure multiple diam manipulation rule subelements. The manipulation rule subelements are the configuration where AVPs are identified, searched, and in which the data is replaced.

The user can apply diameter manipulations to external policy server configurations. These manipulations affect traffic between the Oracle Communications Session Border Controller and the applicable policy server.

> **Note:**
>
> The Oracle Communications Session Border Controller also supports diameter manipulation across the Cx interface, with the user configuring these manipulations to home subscriber server configurations. The range of manipulation supported over the Cx interface is the same as that over the Rx interface.

Manipulation Rule

Creating a manipulation rule is divided into three parts, defining the message type and AVP where the manipulation is performed, defining how the search on the AVP is performed, and defining what to replace the found string with.

You must first define the name parameter of the diam manipulation rule configuration element. Optionally you can add a descr avp code parameter that is a description of this manipulation rule.

Naming Diameter Manipulations

The Oracle Communications Session Border Controller restricts the way you can name a diameter-manipulation rule. Specifically, observe the rules below when naming manipulation elements:

- Character limit - diameter manipulation rule names cannot be longer than 24 characters.
- Numeric characters - diameter manipulation rule names must not start with a numeric character.
- Special characters - Special characters are not supported within diameter manipulation rule names, with the exception of the underscore and hyphen characters.
- Capital letter characters - The Oracle Communications Session Border Controller includes reserved keywords that are named using all-capital letters. To avoid conflicts between keywords and diameter manipulation rules, do not configure diameter manipulation rule names using ll capital letters.

Note that, although diameter-manip-rule and avp-header-rule names have the same character-type restrictions, they do not have a character limit.

Message Based Testing

When the Oracle Communications Session Border Controller first receives a message applicable for manipulation, it checks if the message type as **request**, **response**, or **all** as configured in the msg type parameter. The Oracle Communications Session Border Controller continues to look at the message command code. Matching values are defined by configuring the msg cmd code parameter with a numeric value. You can enter a single value, multiple

comma-separated values, or you can leave this parameter blank to indicate all message codes.

AVP Search Value

After the Oracle Communications Session Border Controller has identified the messages where it can look for an AVP, the avp code must be defined with a numeric AVP value to be searched. Also the AVP data type is defined so Oracle Communications Session Border Controller knows how to correctly parse the AVP once found. This is configured in the avp type parameter with valid values of: octet-string, octet-hex, integer32, unsignedint32, address, utfstring, diameteruri, or enumerated.

The comparison type is defined so that the Oracle Communications Session Border Controller knows the correct way to determine if the match value appears in the avp code. Valid comparison types are:

- Case-sensitive—The comparison-type of both case-sensitive and case-insensitive literally compares the value contained in the match-value against the received value.
- Case-insensitive—The comparison-type of both case-sensitive and case-insensitive literally compares the value contained in the match-value against the received value.
- pattern-rule—the match-value is treated as a regular expression.
- boolean—Used when it is necessary to compare the results of two or several manipulation rules with varying logic (e.g. if ($rule1 & ($rule2 | $rule3))). When the comparison-type is set to boolean, the match-value will be evaluated as a boolean expression.

Finally, the match operation is configured by defining a match value, which is the string to find. The Oracle Communications Session Border Controller evaluates if the match value is found in the avp code AVP. You may also leave the match value empty for the DMR to be applied on the AVP without testing for a match.

Reserved Keywords

The Oracle Communications Session Border Controller employs certain reserved keywords as a short hand for configuration/message parameters. These keywords are as follows:

HOSTNAME—This keyword refers to the agent hostname this rule is being referenced by. If the rule is applied to a realm/interface then the value of the hostname keyword will be an empty string. If the rule is applied to the group, then the hostname for the agent picked will be used.

ORIGINREALM—This keyword refers to the Origin-Realm value for the configured realm/interface. If the rule is applied to a Diameter Director Agent, then the origin-realm value is derived from the Diameter Director Interface the agent belongs to.

ORIGINHOST—This keyword refers to the Origin-Host value for the configured realm/interface. If the rule is applied to a Diameter Director Agent, then the origin-host value is derived from the Diameter Director Interface the agent belongs to.

Actions on Found Match Value

When the match-value is found, the Oracle Communications Session Border Controller references the action parameter. This is configured as either **none**, **add**, **delete**, **replace**, **store**, **diameter-manip**, **find-replace-all**, **log** or **group-manip** and indicates the action to perform on the string. If the match-value is not found, the Oracle Communications Session Border Controller continues processing the message without any AVP manipulation. These actions mean the following:

none

None disables a manipulation rule.

add

This action inserts the value from the **new value** parameter, creates a new AVP of the specified type and adds it to the list of AVPs at the specified position.The message length and padding are re-computed to account for this newly added AVP.

delete

This action removes the specified AVP from the list of AVPs being operated on. The message length and padding will be re-computed to account for this deleted AVP.

replace

This action substitutes the existing value with the **new value** parameter. The message length and padding and AVP length and padding will be re-computed to account for changes. This is mostly applicable to variable length AVP types such as octet-string.

store

Each manipulation rule has the ability to store the data that was contained in the AVP as a string. This is useful for creating conditional logic to make decisions whether to execute other manipulation rules or to duplicate information within the Diameter message.

Every manipulation rule performs an implicit store operation prior to executing the specified action type. All store operations are based on regular expression patterns configured in the **match value**. The information that is stored in the rule is the resultant of the regular expression applied against the specified string. The **comparison-type** is ignored when the action is set to store as the Oracle Communications Session Border Controller assumes that the **match value** is a regular expression. Conditional logic cannot be used to make a decision whether to perform a store operation or not; storing always occurs. Values stored in a manipulation rule are referred to as user defined variables.

diameter-manip

When the action is set to **diameter-manip**, the Oracle Communications Session Border Controller executes the diameter-manipulation **name** configured in the **new value**. The diameter-manipulation name in the **new value** must match another diameter-manipulation name exactly (case is sensitive).

diameter-manip action type is primarily to reuse diameter-manipulations that may be common to other use cases. A diameter-manip action should never call back to itself either directly or indirectly through a different diameter-manipulation.

find-replace-all

The **find-replace-all** action searches the object's string for the regular expression defined in the match-value and replaces every matching occurrence of that expression with the value supplied in the **new value**. If the regular expression contains sub-groups, a specific sub-group can be specified to be replaced by adding the syntax [[:n:]] at the end of the expression, where

n is the sub-group index (zero-based). When the action is find-replace-all, the comparison-type is ignored and the match-value is always treated as a regular expression.

group-manip

The **group manip** action is used to manipulate AVPs inside grouped AVPs. For this diameter manipulation, you must set the avp-type to **grouped**.

The **group manip** action is similar to the **diameter manip** action in that the Oracle Communications Session Border Controller executes the diameter-manipulation configured in the new value.

There is an important difference between **group manip** and diameter manip. The **diameter-manip** action is context insensitive, meaning when it jumps from one diameter-manipulation to the next diameter-manipulation, it starts looking for the specified AVP from the top of the message.

The **group manip** action is context sensitive, meaning when the processing jumps from one diameter-manipulation to the next diameter-manipulation, it will look for the specified AVP within the grouped AVP. In short, the **group manip** works at an AVP level. All actions are allowed in the subsequent manipulations that are invoked, with the key difference being that those manipulation rules will be applied to the current grouped AVP in the chain. Thus it is possible to apply manipulation to an AVP at any level in the hierarchy.

Consider the following examples:

In order to replace the experimental-result, experimental-result-code AVP value from 5002 to 3002, a **group manip** can be configured as follows:

```
diam-manipulation
        name                            manipExpRslt
        description
        diameter-manipulation-rule
                name                            expRslt
                avp-code                        297
                descr-avp-code
                avp-type                        grouped
                action                          group-manip
                comparison-type                 case-sensitive
                msg-type                        response
                msg-cmd-codes                   316,317,318
                match-value
                new-value                       expRsltCode
        last-modified-by                admin@console
        last-modified-date              2011-09-13 18:50:33
diam-manipulation
        name                            expRsltCode
        description
        diameter-manipulation-rule
                name                            expRsltCode
                avp-code                        298
                descr-avp-code
                avp-type                        unsignedint32
                action                          replace
                comparison-type                 case-sensitive
                msg-type                        response
                msg-cmd-codes                   316,317,318
```

```
            match-value                    5002
            new-value                      3002
    last-modified-by               admin@console
    last-modified-date             2011-09-13 18:56:14
```

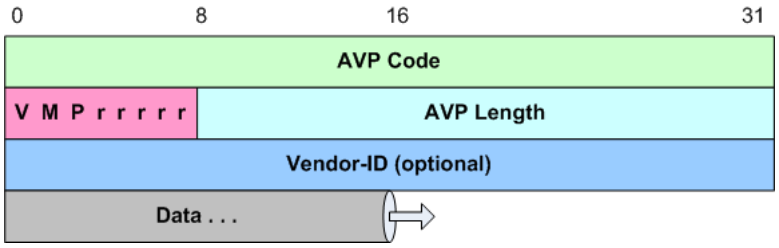
Further, if you want to add a new AVP called AvpD at the following location in the chain of AVPs Message: GrpAvpA, GrpAvpB, GrpAvpC, AvpD, then the manipulation chain would look like this

- diameter-manipulation (name=grpAvpA, action=group-manip, new-value=grpAvpB)
- diameter-manipulation (name=grpAvpB, action=group-manip, new-value=grpAvpC)
- diameter-manipulation (name=grpAvpC, action=group-manip new-value=AvpD)
- diameter-manipulation (name=AvpD action=add new-value="added new value")

> **Note:**
>
> using diameter-manip from inside the group-manip chain may have unexpected impact and must be avoided.

AVP Header Manipulation

In addition to manipulating AVPs, you can manipulate the AVP header itself. After performing AVP DMR, the AVP length and padding is recomputed. This is crucial for scenarios where a vendor-id is added or removed from the header. This functionality is configured in the avp header rules sub element. The following represents a single AVP's header:



AVP Flag Manipulation

You can manipulate the AVP flags by configuring the **header-type** parameter to **avp-flags**, this invokes operation on the flags byte in the AVP header. AVP flags are 1 byte long, where the first 3 bits represent (1) vendor, (2) must and (3) protected. The last 5 bits are reserved.

The vendor flag is critical to consider here, since it has interdependency with Vendor-Id field in the header shown above. As per RFC 3588, The 'V' bit, known as the Vendor-Specific bit, indicates whether the optional Vendor-ID field is present in the AVP header. When set, the AVP Code belongs to the specific vendor code address space. Please find specific details about the rest of the flags in RFC3588 Section 4.1.

When manipulating AVP flags, a bitwise comparison is performed between the received value and the **match value**. For ease of configuration, the **match value** is configured as a comma-separated enumerated list of vendor, must, and protected. So the **new value** and the **match value** will be used to indicate what bit in the **avp-flag** to operate on. If the **match value** is

empty, the configured action is performed without any matching tests. In addition, the new value is configured using the same enumerations. The AVP header rules configuration element appears as follows:

```
avp-header-rules
   name          replaceAvpFlags
   header-type   avp-flags
   action        replace
   match-value   must,protected
   new-value          must
```

According to the example configuration, the Oracle Communications Session Border Controller makes a positive match when only the must and protected bits are set in the received avp-flags. If only the 'M' bit is set, then the match fails, the Oracle Communications Session Border Controller continues to the next header-rule.

When the match is successful (or if the **match value** is left empty), the configured **action** is performed. Consider all following actions applicable to the avp header rules sub element:

- none— no action will be performed
- add—the flags specified in the **new value** are enabled in the header, and state of the existing flags will not be changed.
  If **new value** is empty, the add operation will not be performed.

  If the **new value**=vendor, and the 'V' bit was not originally set, then the 'V' bit is now be set including a vendor-id of 9148 inserted into AVP. 9148 is the Acme Packet vendor-id assigned by IANA. It is expected that a second header-rule will be used to change this to the desired vendor-id.
- replace—all the received avp-flags will be reset. The values in the **new value** parameter will be set.
  If the **new value** is empty, the replace operation will not be performed.

  If the **new value**=vendor, and the 'V' bit was not originally set, then the 'V' bit will now be set and also a vendor-id of 9148 (Acme Packet's vendor-id) is added to the AVP header. A second header-rule may be used to change this to the desired vendor-id.

  If the **new value** does not contain vendor, and the 'V' bit was originally set, then the 'V' bit will be cleared and the vendor-id will also be set to 0 effectively removing it from the AVP header.
- delete—all flags configured in **new value**, will be deleted from the AVP header
  If the **new value** is empty, then no flags are deleted.

  If the particular flag is already empty, then it will be skipped. For example, if the **new value**=must and 'M' bit is not set, after applying the DMR the 'M' bit will still be not set.

  If the **new value**=vendor, then the 'V' bit will be cleared (if not cleared already) and the vendor-id is set to 0, effectively removing the vendor-id from the avp-header.

vendor-id Manipulation

You can manipulate the Vendor ID value in the AVP header by configuring the **header-type** parameter to **avp-vendor-id**. This performs the DMR manipulation on the 4-byte vendor-id in the AVP header. AVP vendor id is present in the AVP header only when the 'V' bit is set in the flags. This is important because the DMR application relies upon the bit being set to determine where the data payload begins.

The avp-vendor-id search invokes an unsigned integer comparison between the received value and the **match-value**. If the **match-value** is empty, the configured action is performed without doing any match.

For the case where **match-value** is non-empty, as in the following example, the DMR engine checks whether the 'V' bit is set in the received header. If not, then the vendor id is not present either and the comparison is unsuccessful. If the 'V' bit is set, and the match succeeds, the match is successful. (An unsuccessful match has the DMR proceed to the next header-rule.)

```
avp-header-rules
   name         replaceAvpFlags
   header-type  avp-vendor-id
   action       add
   match-value  9148
   new-value    10415
```

When the match is successful (or if the **match value** is left empty), the configured **action** is performed. Consider all following actions:

- none—no action will be performed
- add—a configured vendor-id value in the **new-value** parameter is added to the AVP header and the 'V' bit set to indicate it's presence. If you prefer to set the 'V' bit in an AVP, it is better to do an avp-vendor-id action first and then manipulate the rest of the flags.
  If the **new-value** is empty, the add operation is not performed.

  If a vendor-id already exists in the AVP header, then it is replaced by **new-value**.
- replace—the existing vendor-id value is replaced with the **new-value**.
  If the **new-value** is empty, the replace operation is not performed.

  If the vendor-id does not exist in the header, then one will be added with the **new-value** and the 'V' bit is set to indicate its presence.
- delete—the vendor-id will be removed from the AVP header and 'V' bit will be reset to indicate its absence.
  If the **new-value** is empty, then the delete operation will not be performed.

  If the vendor-id does not exist, then the delete operation is not performed.

Multi-instance AVP Manipulation

Some AVPs can appear multiple times within a message. To choose a specific occurrence of an AVP, the **avp code** parameter supports indexing logic. By default, the Oracle Communications Session Border Controller operates on all instances of the specified AVP. However, after configuring an avp-code, you can specify in square brackets, a specific instance of the AVP on which to operate on. The indexing is zero-based. For example,

```
diameter-manipulation-rule
                name                          manip
                avp-code                      293[2]
```

Special characters that refer to non-discrete values are:

- Last occurrence—avp-code[^]
- All—avp-code [*]

The last (^) character is used to refer to the last occurring instance of that AVP. Any [^] refers to the first matching header that matches the specified conditional matching criteria. All [*] is the default behavior, and operates on all headers of the specified-type. For **group manip** action, the AVP index applies to the instance within that grouped AVP.

ACLI Instructions

Diameter Manipulation

To configure a diameter manipulation configuration element:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **diameter-manipulation** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# diameter-manipulation
   ORACLE(diameter-manipulation)#
   ```

4. **name**—Enter the name of this Diameter manipulation element.
5. **description**—Enter an optional description for this Diameter manipulation.
6. Type **done** and continue.

Manipulation Rule

1. Type **diameter-manip-rules** to continue and enter individual policy attributes. The system prompt changes to let you know that you can begin configuring individual parameters.
2. **name**—Enter the name of this manipulation rule. This value will be referenced in when creating a chain of rules.
3. **descr-avp-code**—Enter a description of the AVP code to manipulate.
4. **msg-cmd-code**—Enter the command code number of the message to execute the manipulation on.
5. **msg-type**—Set this to the type of message this manipulation applies to as **request**, **response**, or **all**.
6. **avp-code**—Enter the AVP by code number where this manipulation applies. You can add a multi instance identifier to the end of the avp code value, enclosed in brackets.
7. **avp-type**—Set this to the data type of the content of the match field. Refer to the Diameter standards document for the encodings of individual AVPs. Valid values are:

   none | octet-string | octet-hex | integer32 | unsignedint32 | address | diameteruri | enumerated | grouped
8. **match-value**—Enter the value within the match-field to find and make a positive match on.

9. **action**—Enter either **none, add, delete, store, diameter-manip, group-manip, find-replace-all**, or **replace** as the action to take after making a positive match on the previously entered match-value.
10. **new-value**—Enter the value that should be added or replaced in the old match-value's position.
11. Type **done** and continue.

AVP Header Manipulation

1. Type **avp-header-rules** to configure AVP header manipulation rules. The system prompt changes to let you know that you can begin configuring individual parameters.
2. **name**—Enter the name of this AVP Header manipulation rule.
3. **header-type**—Set this to either avp-flag or avp-vendor-id depending on which part of the AVP header you are manipulating.
4. **action**—Enter either none, add, delete, or replace as the action to take after making a positive match on the previously entered match-value.
5. **match-value**—Enter the value in the AVP flag field or Vendor ID field to match against.

   When matching in the avp flag field, then match-value is interpreted as comma-separated list of enumerated values <vendor,protected,must>. When matching in the Vendor ID field, then match-value is interpreted as 32 bit unsigned integer <1-4294967295>
6. **new-value**—Enter the new value when the match value is found. The resultant new value is entered as the match value is configured.
7. Type **done** to save your work.

Applying the Manipulation

You can apply a diameter manipulation by name to an external policy server configuration. This element contains the two applicable parameters: diameter-in-manip and diameter-out-manip.

> **Note:**
>
> The user can also apply diameter manipulation to a home subscriber server configuration using these same parameters.

1. To navigate to external policy server configurations from Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Enter the configuration element where you wish to apply the manipulation.
4. Type **ext-policy-server** and press Enter.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

5. Type **select** and then choose the pre-configured external policy server you want to configure.

```
ORACLE(ext-policy-server)# select
<hostname>:
1: ext-pol-svr-1
2: ext-pol-svr-2
selection: 1
```

You may now add a Diameter manipulation to one or both directions of message flows.

6. **diameter-in-manip**—Enter a name of an existing diameter manipulation to apply as received by the Oracle Communications Session Border Controller on this element.
7. **diameter-out-manip**—Enter a name of an existing diameter manipulation to apply as forwarded from the Oracle Communications Session Border Controller on this element.
8. Type **done** and continue.

Diameter Manipulation Example - Supported Features AVP

This section shows you a configuration example for diameter manipulation rules. This section shows the configuration for the rule that the Oracle Communications Session Border Controller applied, and sample results of the manipulation. These examples present configurations as an entire list of fields and settings for each ruleset, nested header rules and nested element rules. If a field does not have any operation within the set, the field is shown with the setting at the default or blank.

For this manipulation rule, the Oracle Communications Session Border Controller inserts a Supported Features AVP into every request.

This is a sample of the configuration:

```
diameter-manipulation
        name                                    diamManip1
        description
        diameter-manip-rule
                name                                    rule1
                avp-code                                628
                descr-avp-code
                avp-type                                grouped
                action                                  add
                msg-type                                request
                msg-cmd-code                            265
                comparison-type                         case-sensitive
                match-value
                new-value
        diameter-manip-rule
                name                                    rule2
                avp-code                                628
                descr-avp-code
                avp-type                                grouped
                action                                  group-manip
                msg-type                                any
                msg-cmd-code
                comparison-type                         case-sensitive
                match-value
```

```
                new-value                               diamManip2
```

This second rule, which defines a new value for the first rule, builds the Feature-List-ID and Feature-List AVPs to be included within the context of the Supported Features group.

```
diameter-manipulation
        name                                    diamManip2
        description
        diameter-manip-rule
                name                                    rule1
                avp-code                                266
                descr-avp-code
                avp-type                                unsignedint32
                action                                  none
                msg-type                                any
                msg-cmd-code
                comparison-type                         case-sensitive
                match-value                             10
                new-value
        diameter-manip-rule
                name                                    rule2
                avp-code                                629
                descr-avp-code
                avp-type                                unsignedint32
                action                                  none
                msg-type                                any
                msg-cmd-code
                comparison-type                         case-sensitive
                match-value                             11
                new-value
        diameter-manip-rule
                name                                    rule3
                avp-code                                630
                descr-avp-code
                avp-type                                unsignedint32
                action                                  none
                msg-type                                any
                msg-cmd-code
                comparison-type                         case-sensitive
                match-value                             124
                new-value
```

COPS-based External Policy Servers

The Common Open Policy Service (COPS) [RFC 2748] is a protocol supported by the Oracle Communications Session Border Controller to perform and implement Call Admission Control (CAC) based on the policies hosted in an external policy server. While the Oracle Communications Session Border Controller already supports internal CAC policies, they are not as flexible as a Resource and Admission Control Function / Policy Decision Function (RACF/PDF), the generic resource and admission control functional architecture conceived by the ITU-T and the IETF.

The Oracle Communications Session Border Controller COPS model includes a Policy server, functionally called the policy decision point (PDP), and the edge router, functionally called the policy enforcement point (PEP), the Oracle Communications Session Border Controller itself. The PDP and the PEP communicate with each other via the COPS protocol.

COPS Connection

The COPS session is established over a persistent TCP connection between the PDP and PEP. A COPS Client-Open (OPN) message is sent from the Oracle Communications Session Border Controller to the RACF, which responds with a COPS Client-Accept (CAT) message. A COPS Client-Close (CC) message is sent to either side to gracefully close the persistent connection. This COPS connection is expected to never close, unless an error occurs.

COPS Failures

Connection failures are discovered through a keep alive mechanism. Keep alive (KA) messages are periodically sent by the Oracle Communications Session Border Controller to the RACF regardless if any other COPS messages have been exchanged. When a KA message is not received, a connection failure is flagged. If the COPS connection fails, the Oracle Communications Session Border Controller will continually try to re-establish the connection to the PDP. Previously established calls will continue unaffected, but the Oracle Communications Session Border Controller will deny new calls from being established until the COPS connection is restored.

Failure Detection

A COPS connection failure is triggered by one of the three following events:

1. COPS KA timeout. The Oracle Communications Session Border Controller flags a COPS KA timeout when it does not receive a response for the KA it sent to the PDP. The PDP flags a COPS KA timeout when it does not receive the KA message within its requested timer time from the Oracle Communications Session Border Controller. At a minimum, when the COPS KA message times out, the TCP socket is closed.
2. Explicit COPS CC. The Oracle Communications Session Border Controller closes a COPS connection if it receives a COPS CC message from the PDP. The PDP closes a COPS connection if it receives a CC message from the Oracle Communications Session Border Controller. After the COPS layer connection is closed, then the TCP socket is closed too.
3. TCP socket termination. If either side receives a TCP FIN or RST, the TCP socket closes as expected. The COPS layer then detects that the socket has been closed before sending any further messages, and thus the COPS connection is closed.

Failure Recovery

The Oracle Communications Session Border Controller assumes that the PDP has a mechanism that re-uses the same logical IP Address, restarts itself in a timely manner, or fails over to another PDP. Therefore, no backup PDP IP address is configured on the Oracle Communications Session Border Controller.

The Oracle Communications Session Border Controller will try to re-open the COPS connection to recover from a connection failure. The PDP is never the device to initiate a connection. The Oracle Communications Session Border Controller increases its retry interval after successive reconnect failures. Once the retry interval has grown to every five minutes, the Oracle Communications Session Border Controller continues to retry to open the COPS connection at the five minute interval.

COPS PS Connection Down

You can configure whether or not you want the Oracle Communications Session Border Controller to reject or allow new calls to be established despite the failure of a policy server (PS) connection.

You enable this feature in the external policy server configuration using a new parameter. When you enable the feature, the Oracle Communications Session Border Controller allows new SIP calls to be established even though the connection to the PS has failed. In this case, the PS will not respond and will not be aware of the established sessions. When you disable this feature, the Oracle Communications Session Border Controller behaves as it did in prior releases by responding to a connection failure with a 503 Service Unavailable.

HA Support

The Oracle Communications Session Border Controller's high availability (HA) capabilities have been extended to support COPS. When one Oracle Communications Session Border Controller in an HA configuration goes down, the MAC addresses are reassigned to a healthy Oracle Communications Session Border Controller. IP addresses "follow" the MAC addresses to provide a seamless switchover between HA nodes.

After an HA failover, the COPS connection on the primary Oracle Communications Session Border Controller is either gracefully torn down, or times out depending on behavior of the PDP. The backup Oracle Communications Session Border Controller attempts to create a new COPS connection with the PDP. The OPN message uses the same PEPID and Client Type as in the previous pre-failover session.

Application Types

The Oracle Communications Session Border Controller supports the following COPS-based methods for interfacing with a RACF:

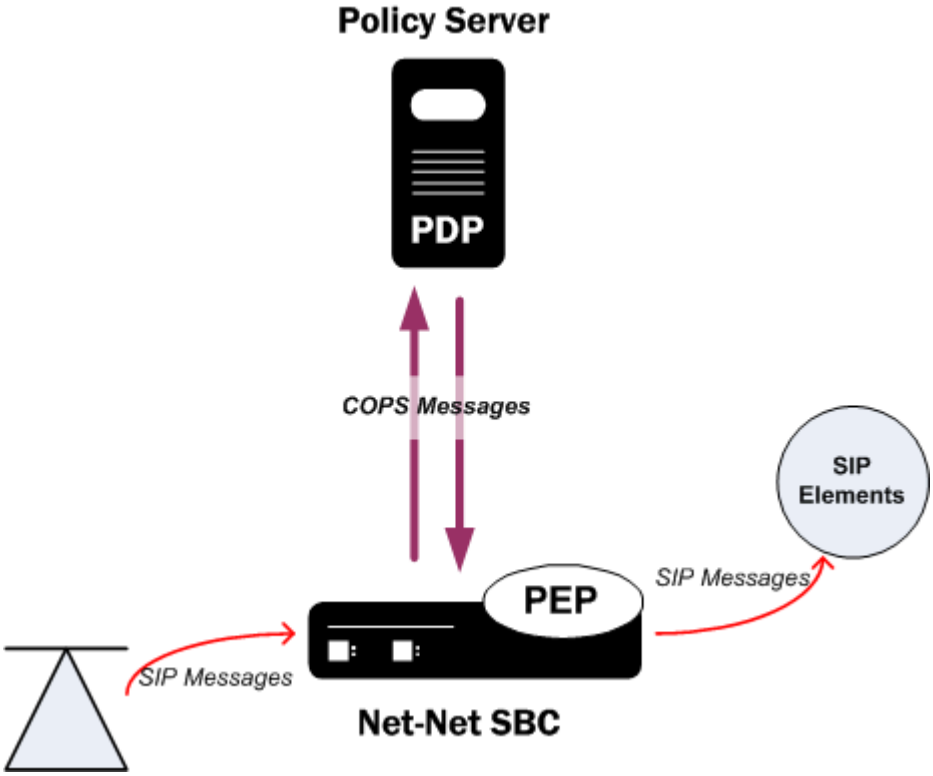
- PKT-MM3 (PacketCable™ Specification Multimedia Specification PKT-SP-MM-I03-051221) (client type: 0x800A)
- Acme Packet proprietary (client type: 0x7926)

The Oracle Communications Session Border Controller supports the following COPS-based methods for interfacing with a CLF:

- Oracle proprietary (client type: 0x7929)

COPS: RACF

CAC is performed according to the following typical scenario. When the Oracle Communications Session Border Controller receives a SIP INVITE, it sends a COPS request (REQ) message to the PDP. The REQ message includes the call ID, the SIP client's IP address, the Oracle Communications Session Border Controller's IP address and port number of the ingress interface for the call, and SDP based bandwidth requirements. The PDP responds with a COPS Decision (DEC) message with either the Install or Remove command. An Install command directs the Oracle Communications Session Border Controller to forward the INVITE to the next SIP device. A Remove command directs the Oracle Communications Session Border Controller send a SIP 503 Service Unavailable message sent back to the UA and reject the call.



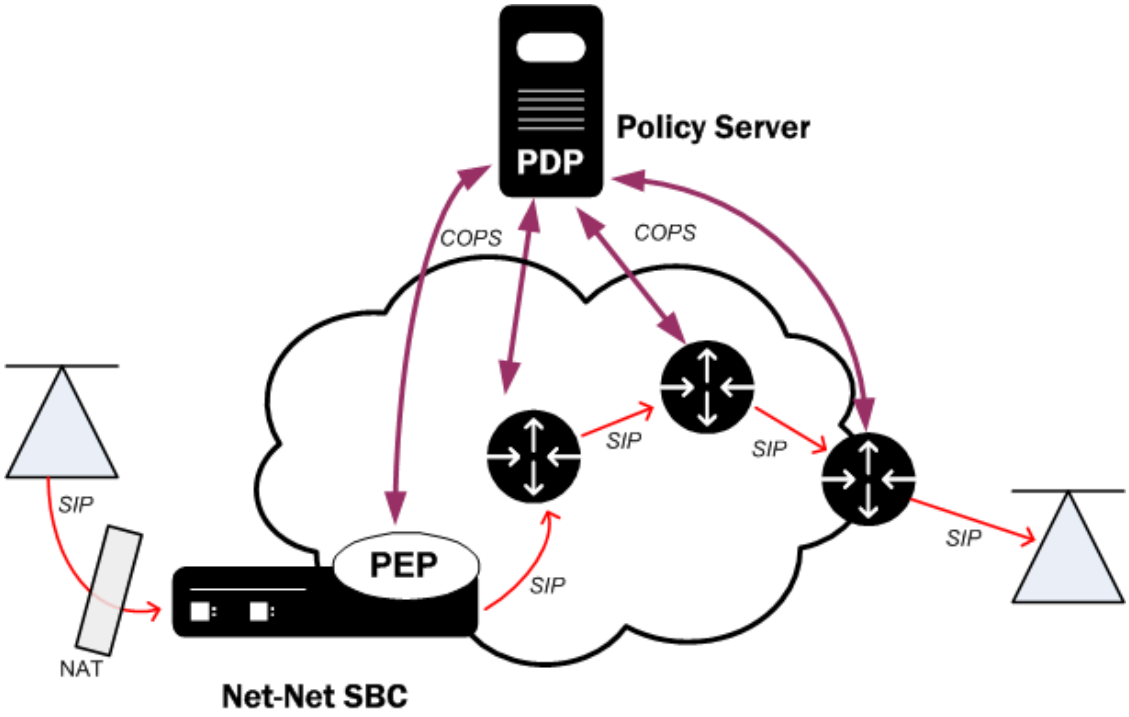
The Oracle Communications Session Border Controller can be configured so that both sides of a call, based on realm, are subject to COPS bandwidth enforcement. Each flow is treated as a unique call/event, because from a media and signaling perspective, they are distinct. As the Oracle Communications Session Border Controller functions as one side of a call, its IP address is inserted into the REQ message regardless of whether it is the calling or called party. This allows for the COPS install or remove decision to be made before the Oracle Communications Session Border Controller receives the 200 OK response, and before ringing the far-end phone. Only one external policy server can be used within a single realm.

When a call ends, either with the expected SIP BYE or CANCEL methods, or due to other error conditions, the Oracle Communications Session Border Controller will delete the reservation on the PDP by sending a COPS delete request state (DRQ) message to the PDP. All ended calls must be deleted from the PDP in order to accurately track used and available bandwidth.

Implementation Features

As the Oracle Communications Session Border Controller proxies and forwards calls, caller media information is known before the INVITE reaches the callee. The PEP can request a specific amount of bandwidth for a call, and the PDF can reserve this amount of bandwidth for a call before the called phone rings. A call's required bandwidth can also be reserved by network devices along the path from the caller to callee if the QoS admission criteria is pushed to PEPs such as routers, along this path to the callee.

The RACF can apply its hosted policies for calls originating at SIP UAs located behind NATs. This is a standard part of the Oracle Communications Session Border Controller's ability to provide seamless HNT.



Bandwidth Negotiation

Because the decision whether to admit or reject a call is made before the INVITE is forwarded to the called party, some information is not available to the PDP at the initial request. The final IP Address, UDP port number, that transport the RTP flow, and the codec used are not known by the Oracle Communications Session Border Controller until the called party responds with its own SDP information (either in the 180 or 200 response).

The SBC sends a request to the PDP requesting as much bandwidth as the codec with the highest bandwidth in the SDP message requires. If the call is admitted, and when the called party returns finalized SDP information, the SBC modifies the original reservation with the chosen codec's bandwidth requirements. This ensures the PDP has current and accurate information with which to make policy decisions.

COPS pkt-mm-3 Policy Control

You can configure the Oracle Communications Session Border Controller's COPS pkt-mm-3 interface to maintain a persistent TCP connection to the external policy server, despite there being no responses to requests for bandwidth. This permits calls to traverse the Oracle Communications Session Border Controller even though the external policy server either fails to respond or rejects the session. Without this functionality configured, the Oracle Communications Session Border Controller waits for the external policy server's authorization decision or the request to the external policy server times out, resulting in COPS pkt-mm-3 interface error responses and time-outs.

While these type of error responses and time-outs might be handled in some networks, for others they cause unacceptable call failures and call denials. To avoid this situation, you can select the granularity at which your network admits calls through a best-effort pipe and guarantee the Oracle Communications Session Border Controller's TCP connection to the external policy server remains uncompromised. The external policy server offers three parameters allowing you to enable this capability: **permit-on-reject**, **disconnect-on-timeout**, and **gate-spec-mask**.

Relationship to the TCP Connection

The TCP connection between the Oracle Communications Session Border Controller and the external policy server plays an important role because second-tier networks can experience issues between the policy server and the cable modem termination system (CMTS). Since one policy server can be connected to multiple others, one CMTS's failure can cascade and cause multiple failures for other CMTSs and policy servers.

Enables the **disconnect-on-timeout** parameter allows the Oracle Communications Session Border Controller to maintain its TCP connection to the external policy server regardless of upstream issues between PSs and CMTSs. When you disable this setting, the Oracle Communications Session Border Controller can send Gate-Set and Gate-Delete messages to in response the PS's timeouts and guard against impact to the TCP connection between the Oracle Communications Session Border Controller and the PS.

COPS Gate-Set Timeout

Without this capability enabled, the Oracle Communications Session Border Controller views the situation as a rejection when its COPS Decision Message times out at the PS. As such, the Oracle Communications Session Border Controller denies the session or attempts to revert to the previously requested bandwidth from either a G**ate**-Set add or Gate-Delete modify message. The **permit-on-reject** parameter, however, enables the system to forward the session on at a best-effort.

Enabling both the **permit-on-reject and the disconnect-on-timeout** parameters prompts yet another Oracle Communications Session Border Controller behavior. Consider the situation where a session is in the processing of being established on the Oracle Communications Session Border Controller, and it carries COPS Gate-Set messages to the PS. Typically in this scenario, one voice call has two COPS decision messages: one for the upstream audio, and one for the downstream audio. If any of the Gate-Set message do not receive a response, the Oracle Communications Session Border Controller forwards on the session under as a best effort. And if a non-final Gate-Set message in the session does not receive a response, any remaining Gate-Set messages for the session are not forwarded to the PS; the whole session is forwarded on as a best effort. In other words, if the first Oracle Communications Session Border Controller Gate-Set message out of two times out, the second for the session is not sent.

At every opportunity, the Oracle Communications Session Border Controller tried to elevate the session to a QoS session from its best-effort status. When you enable the external policy server configuration's **reserve-bandwidth** parameter, the Oracle Communications Session Border Controller checks twice for bandwidth: once upon INVITE, and a second time upon the 200 OK. Alternatively, when Oracle Communications Session Border Controller receives a ReINVITE, it checks for bandwidth then, too. At the time of these second checks, a call's best-effort status can improve to that of a QoS session.

When a Gate-Set Times Out

If the Oracle Communications Session Border Controller labels a session best-effort, it does so because the **Gate-Set** message it has sent to the PS have timed out. The Oracle Communications Session Border Controller does not send a **Gate-Delete** in this situation because it has not received a valid **Gate-Id** for that session from the PS.

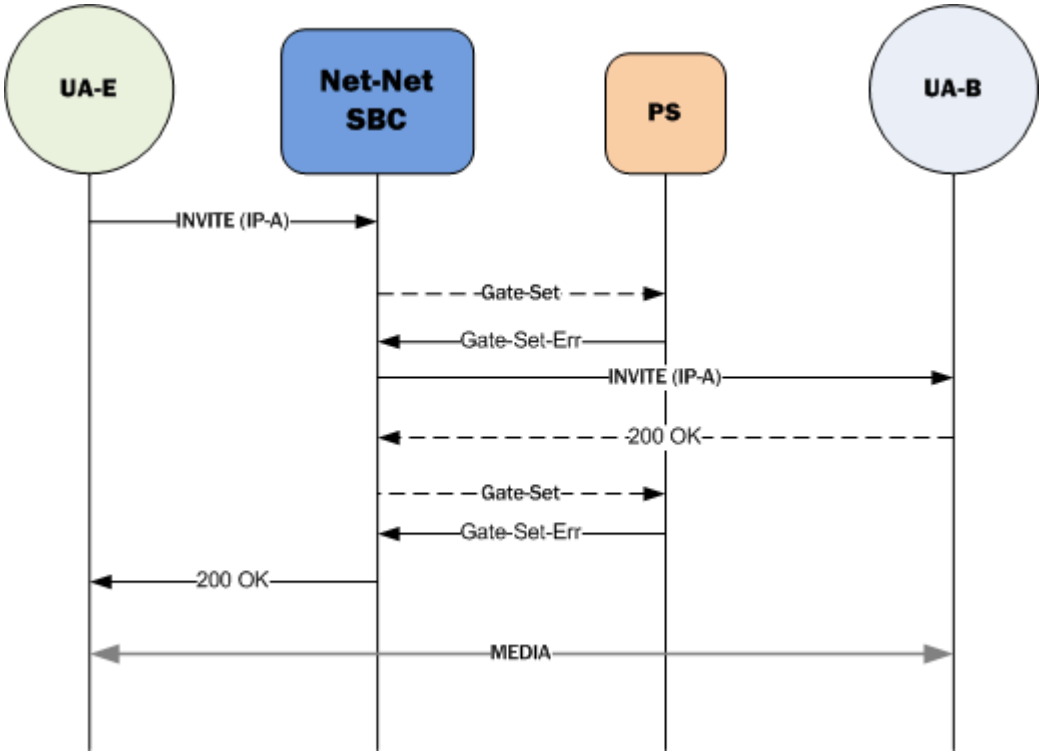
It is also possible that when a Gate-Set message times out, the CMTS and PS might have allocated the session after the Oracle Communications Session Border Controller's timeout.

One of two errors scenarios result from this occurrence, neither of which is of significance. However, you should be aware of them:

- If a Gate-Set message times out on the INVITE side of the session, the subsequent Gate-Set message triggered by the 200 OK looks like a request to allocate the same resources. If the CMTS and PS are able to successfully install the flow outside the Oracle Communications Session Border Controller's eight-second request time-out, issues can ensue.
- Assuming the Gate-Set message time out but are nonetheless installed after the Oracle Communications Session Border Controller's request time-out, the Oracle Communications Session Border Controller does not send Gate-Delete messages for the flows because it believes they are unallocated on the PS. Because the Oracle Communications Session Border Controller has no Gate-Id data, it remains unaware the sessions are installed on the PS and CMTS. The PS and CMTS might then be left with stranded gates.

This diagram shows a call flow where the Gate-Set times out. It assumes an access-to-core configuration, where the external bandwidth management configuration shows:

- **disconn-on-timeout**=disabled
- **permit-conn-down**=enabled



COPS Decision Gate-Set Message Rejected

The COPS pkt-mm-3 interface reply as it does when this feature is not enabled (reply with an error to the signaling application) when:

- The PS responds to the Gate-Set message with an error, either denying or rejecting the flow
- The **permit-on-reject** parameter is disabled (its default setting)

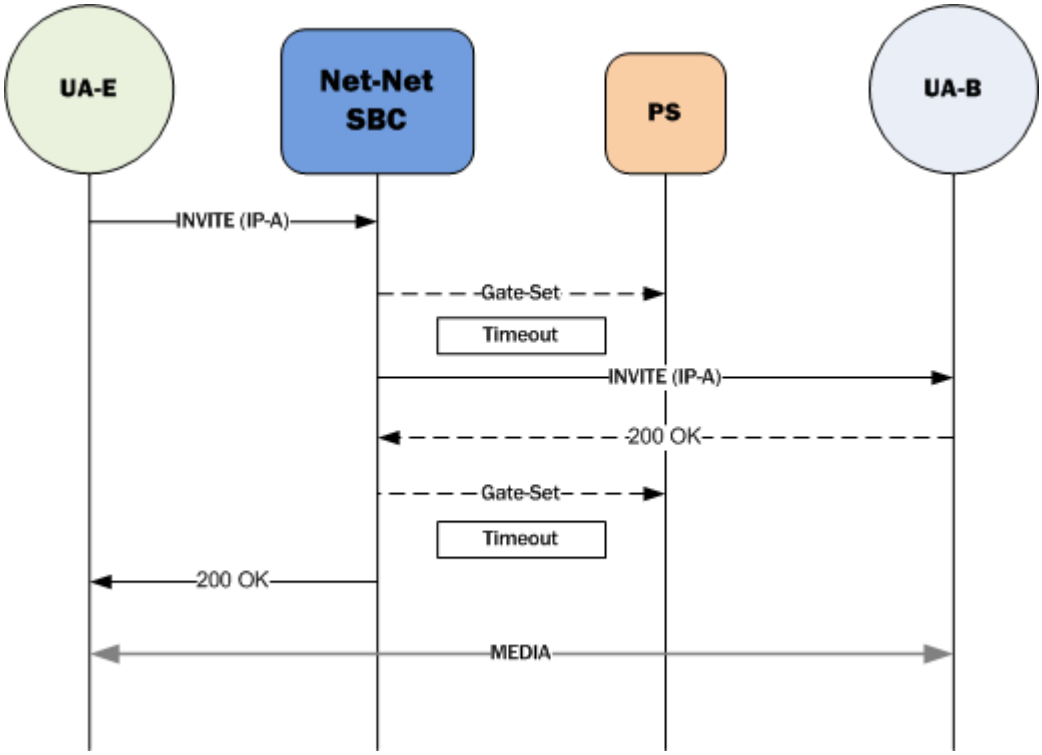
This again results in the situation where a session is in the processing of being established on the Oracle Communications Session Border Controller, and it carries COPS Gate-Set messages to the PS. Typically in this scenario, one voice call has two COPS decision messages: one for the upstream audio, and one for the downstream audio. If any of the Gate-Set message do not receive a response, the Oracle Communications Session Border Controller fowards on the session under as a best effort when the **permit-on-reject** parameter is enabled.

If a non-final Gate-Set message for a session elicits an error or deny response, the remaining Gate-Set message(s) for that session are not sent to the PS. The Oracle Communications Session Border Controller forwards the entire session as a best effort. This means that if the Oracle Communications Session Border Controller sends two Gate-Set messages and the first meets an error or deny response, then the second one is not sent. Still, the Oracle Communications Session Border Controller attempts to elevate the session's status from best-effort to QoS given the multiple requests for bandwidth it might issue.

Note

> **Note:**
>
> The Oracle Communications Session Border Controller does not send Gate-Delete message for session that have been deemed best-effort. This is because the Gate-Set messages the Oracle Communications Session Border Controller sent to the PS have timed out, and the Oracle Communications Session Border Controller does not have the PS's Gate-Id data for the session.

This diagram shows a call flow where Gate-Set messages are denied. t assumes an access-to-core configuration, where the external bandwidth management configuration shows a **permit-on-reject set to enabled.**



About the Gate-Spec Mask

Vendors of policy servers and those who incorporate PSs into their networks must allow the CMTS to overwrite IP packets with ToS bytes because the value of the Oracle Communications Session Border Controller 's DSCP/TOS mask is 0xFF. The CMTS uses the following equation to determine whether or not a ToS byte should be overwritten:

```
new-ip-tos=((orig-ip-tos AND tos-and-mask) OR (tos-or-mask)
```

Using the **gate-spec-mask** parameter, you can configure theOracle Communications Session Border Controller to use a mask comprised entirely of zeros (0s). Doing so allows the PS to reset the byte before calculations using the media policy configuration's DCSP field can take place.

COPS Debugging

A new argument has been added to the show command for viewing COPS and CAC statistics. From the user prompt, type **show ext-band-mgr**.

```
ORACLE# show ext-band-mgr
10:11:38-194
EBM Status                  -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                1       1       1          1       1       1
Connections            0       0       0          0       0       0
                         ---- Lifetime ----
                    Recent      Total  PerMax
Reserve                  0          0       0
Modify                   0          0       0
Commit                   0          0       0
Remove                   0          0       0
EBM Requests             0          0       0
EBM Installs             0          0       0
EBM Errors               0          0       0
EBM Rejects              0          0       0
EBM Expires              0          0       0
EBMD Errors              0          0       0
```

Retrieve the COPS statistics in the log.embd file.

COPS-based RACF Configuration

In the following configuration examples, we assume that your baseline configuration passes SIP traffic. In this example, you will configure additions to the ream configuration and the new external bandwidth manager configuration. You must also configure media profiles to accept bandwidth policing parameters.

Realm Configuration

To configure the realm configuration for COPS support in a CAC scenario:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure.

   ```
   ORACLE(realm-config)# select 1
   ORACLE(realm-config)#
   ```

5. **mm-in-realm**—Set this parameter to enabled so that calls from devices in the same realm have their media flow through the Oracle Communications Session Border Controller to be subject to COPS CAC. The default value is **disabled**. The valid values are:
   - enabled | disabled
6. **mm-in-network**—Set this parameter to **enabled** so that the Oracle Communications Session Border Controller will steer all media traveling between two endpoints located in different realms, but within the same network. If this field is set to **disabled**, then each endpoint will send its media directly to the other endpoint located in a different realm, but within the same network. The default value is **enabled**. The valid values are:
   - enabled | disabled
7. **ext-bw-manager**—Enter the name of the external bandwidth manager configuration instance to be used for external CAC for this Realm.
8. Save your work using the ACLI **done** command.

External Bandwidth Manager Configuration

To configure the external bandwidth manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. **name**—Enter the name for this external bandwidth manager instance. This parameter is used to identify the PDP used for that will be used in each Realm configuration.
5. **state**—Set state to **enabled** to enable this external policy server.
6. **operation-type**—Enter **bandwidth-mgmt** for this external policy server configuration element to perform bandwidth management and communicate with a RACF. This sets the COPS client type to **0x7926**. If another vendor's Policy Server is supported, it will be a different protocol value.
7. **protocol**—Enter **COPS** to support COPS-based CAC. The **A-COPS** protocol implicitly sets the Oracle Communications Session Border Controller to use **0x4AC0** as the COPS client type. The default value is **C-SOAP**.
8. **address**—Enter the IP Address or FQDN of the external COPS-based policy server.
9. **port**—Enter the port number the COPS connection connects to on the PDP. The default value is **80**. (The standard port for COPS is 3288.) The valid range is:
   - Minimum—0
   - Maximum—65535
10. **realm**—Enter the name of the Realm in which this Oracle Communications Session Border Controller defines the external policy server. This is NOT necessarily the Realm that the Oracle Communications Session Border Controller performs admission requests for.
11. **application-mode**—Enter the type of interface you want to use. Your choices are: **Rq**, **Rx**, **Gq**, **e2**, and **none**. Set this to **none** or **pkt-mm3**.
12. **application-id**—Enter a numeric application ID that describes the interface used to communicate with the RACF. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
13. **permit-conn-down**—Enter enabled for the Oracle Communications Session Border Controller to establish new SIP sessions despite PS connection failure. The default value is **disabled**. The valid values are:
    - enabled | disabled
14. **reserve-incomplete**—Set this parameter to **enabled** when communicating with a PDP via COPS. The parameter allows the Oracle Communications Session Border Controller to make admission requests before learning all the details of the flows and devices (e.g., not knowing the final UDP port numbers for the RTP media streams until after the RTP has begun). The default value is **enabled**. The valid values are:
    - enabled | disabled
15. **permit-on-reject**—Set this parameter to **enabled** if you want the Oracle Communications Session Border Controller to forward the session on a best-effort basis. Leave this parameter set to **disabled**, its default, if you want the system deny the session or attempts to revert to the previously requested bandwidth.
16. **disconnect-on-timeout**—Leave this parameter set to enabled, its default, so the Oracle Communications Session Border Controller can maintain its TCP connection to the external policy server regardless of upstream issues between PSs and CMTSs. When you

disable this setting, the Oracle Communications Session Border Controller can send Gate-Set and Gate-Delete messages to in response the PS's timeouts and guard against impact to the TCP connection between the Oracle Communications Session Border Controller and the PS.

17. **gate-spec-mask**—This parameter sets the value to use for Gate-Spec mask for the COPS pkt-mm-3 interface. The default is **.255**. The minimum value is **0**, and the maximum is **255**.
18. Save your work using the ACLI **done** command.

Media Profile Configuration

To configure the media profile configuration for COPS support in a CAC scenario:

Values for the following parameters can be found in the PacketCable™ Audio/Video Codecs Specification PKT-SP-CODEC-I06-050812 document.

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session router path.

```
ORACLE(configure)# session-router
```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# media-profile
ORACLE(media-profile)#
```

4. Type **select** and the number of the pre-configured media profile you want to configure.

```
ORACLE(media-profile)# select 1
ORACLE(media-profile)#
```

5. **peak-rate-limit**—Enter the r, P value:
   - **r**—bucket rate
   - **p**—peak rate
6. **max-burst-size**—Enter the b, m, M value:
   - **b**—Token bucket size
   - **m**—Minimum policed unit
   - **M**—Maximum datagram size
7. Save your work using the ACLI **done** command.

COPS: CLF

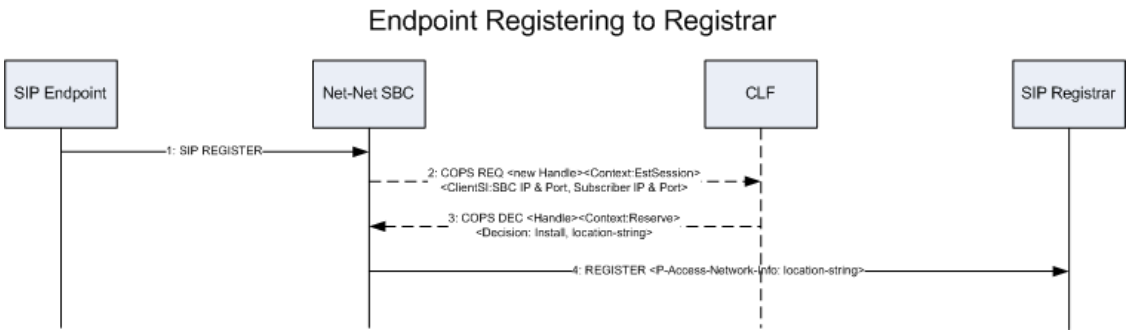
A Connectivity Location Function (CLF) maintains mappings between endpoints with dynamically assigned IP addresses and their physical location. The Oracle Communications Session Border Controller , acting as a P-CSCF, is the intermediary device between a registering endpoint and a CLF. The CLF thus validates and tags a registering endpoint, and the Oracle Communications Session Border Controller applies the CLF's actions. The Oracle

Communications Session Border Controller and the CLF maintain a connection with each other using the COPS protocol.

CLF Behavior

The Oracle Communications Session Border Controller and a CLF only interact when an endpoint registers or re-registers. The Oracle Communications Session Border Controller, acting as the P-CSCF, is the first SIP device that the REGISTER message reaches. Upon receiving the REGISTER message(1), the Oracle Communications Session Border Controller queries the CLF using the COPS protocol. The endpoint's (public) IP address and port, and the Oracle Communications Session Border Controller 's IP information are sent to the CLF in a COPS REQ message(2).

The CLF responds to the Oracle Communications Session Border Controller with an Approve or Reject COPS DEC message(3). If the request is approved, then the CLF also sends a location-string value to be inserted in one of the SIP headers. The Oracle Communications Session Border Controller inserts a P-Access-Network-Info header containing the location-string into the incoming REGISTER message and forwards this message(4) to the SIP registrar/I/S-CSCF.



The Oracle Communications Session Border Controller will insert this P-Access-Network-Info header into all subsequent SIP messages from this endpoint as they are forwarded into the core network. The P-Access-Network-Info header is inserted into all SIP requests and responses except for ACK and CANCEL messages. For all boundaries where SIP messages pass from trusted to untrusted SIP interfaces or session agents, the Oracle Communications Session Border Controller will strip out the P-Access-Network-Info header as expected.

If the CLF responds with a Reject DEC message, the Oracle Communications Session Border Controller rejects the registration, and sends a 503 - Service Unavailable message back to the registering endpoint. In this way, the CLF can be used for admission control.

The Oracle Communications Session Border Controller communicates with the CLF solely for retrieving location information from the CLF, and not for notifying the CLF about an endpoint's registration state or activity. When an endpoint's registration ends, either through a normal expiration, getting rejected by the registrar, or through specific de-registering or error conditions, the Oracle Communications Session Border Controller deletes the locally cached registration location string. The Oracle Communications Session Border Controller does not update the CLF about any registrations that have been deleted.

P-Access-Network-Info Header Handling

The P-Access-Network-Info header is created and populated according to the following rules:

- If the CLF returns an Accept DEC message and a location string, the Oracle Communications Session Border Controller inserts the location string into a P-Access-Network-Info header in the outgoing REGISTER message.

- If the CLF returns an Accept DEC message without a location string, the Oracle Communications Session Border Controller inserts the configured default string into a P-Access-Network-Info header in the outgoing REGISTER message.
- If the CLF returns an Accept DEC message without a location string and no location string is configured on Oracle Communications Session Border Controller , the outgoing REGISTER message is forwarded out of the Oracle Communications Session Border Controller , but no P-Access-Network-Info header is created for the REGISTER message.

CLF Re-registration

The Oracle Communications Session Border Controller will send a new REQ message to the CLF to request a new location string if any of the following events occur:

- The endpoint's contact address changes.
- The SIP Register message's Call-ID header changes.
- The endpoint's public IP Address or UDP port changes.
- The endpoint connects to a different SIP interface, port, or realm on the Oracle Communications Session Border Controller than it did in the initial REGISTER message.
- The registration expires in the Oracle Communications Session Border Controller 's registration cache.

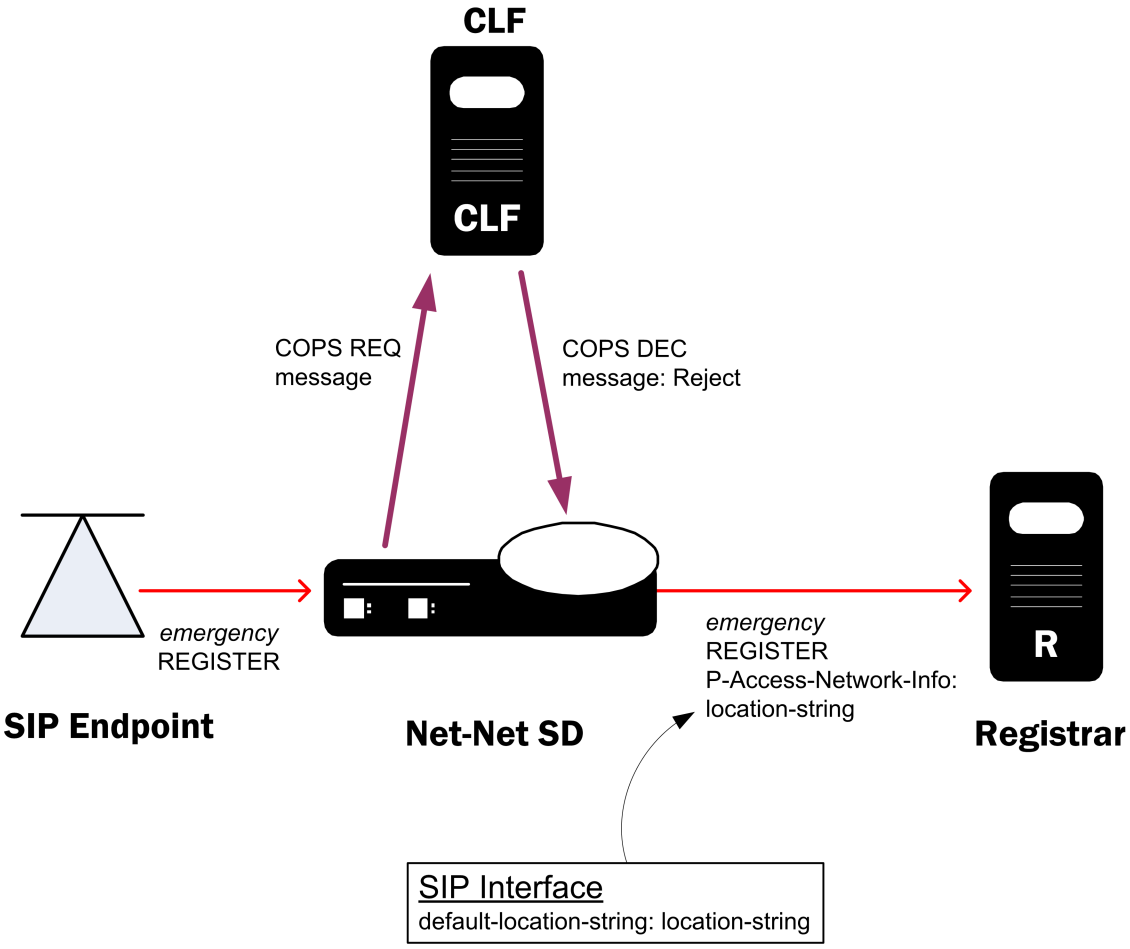
CLF Failures

If a COPS connection fails, the Oracle Communications Session Border Controller will continually try to re-establish the connection. Endpoints that are already registered will stay registered unless they timeout or if the registrar rejects their refreshes. When the COPS connection has not been established, and an endpoint registers on a SIP interface that is configured to use CLF, the Oracle Communications Session Border Controller forwards new REGISTER messages to the registrar using the default location string.

CLF Emergency Call Handling

The Oracle Communications Session Border Controller allows emergency calls into the network even if the endpoint that places the emergency call is not registered. In the expected fashion, the Oracle Communications Session Border Controller will query the CLF first for an incoming emergency call sourced from an unregistered endpoint. If the CLF response is successful, then the Oracle Communications Session Border Controller will insert the string returned from the CLF into a P-Access-Network-Info header, and insert this header into the emergency call's REGISTER message. If no location string is returned with a successful CLF response, the default location string is inserted into P-Access-Network-Info header.

If the CLF's response is to reject the emergency call, the Oracle Communications Session Border Controller will insert the configured default location string into the P-Access-Network-Info header and forward the emergency call's REGISTER message toward the registrar. For emergency calls where the endpoint has already successfully registered, the call will be routed into the network using the expected methods for emergency call routing.



If the COPS connection to the CLF is down, emergency calls from un-registered endpoints are still allowed into the network using the default string inserted into the emergency messages.

HA Functionality

The location strings generated by the CLF are replicated on the standby Oracle Communications Session Border Controller in an HA pair. This is required so that a Oracle Communications Session Border Controller in an HA pair can instantly continue processing calls using the previously learned CLF information.

CLF Debugging

A new argument has been added to the show command for viewing CLF statistics. From the user prompt, type **show** <space> **ext-clf-svr** <return>.

```
ORACLE# show ext-clf-svr
14:17:14-114
EBM Status                  -- Period -- -------- Lifetime --------
                   Active     High   Total       Total  PerMax     High
Client Trans            0        0       0           0       0        0
Server Trans            0        0       0           0       0        0
Sockets                 0        0       0           0       0        0
Connections             0        0       0           0       0        0
                                 ---- Lifetime ----
                     Recent      Total  PerMax
```

```
CLF Requests              0          0         0
CLF Admits                0          0         0
CLF Errors                0          0         0
CLF Rejects               0          0         0
CLF Expires               0          0         0
CLFD Errors               0          0         0
```

Retrieve CLF statistics in the log.embd file.

COPS-based CLF Configuration

In the following configuration examples, we assume that your baseline configuration passes SIP traffic, with the Oracle Communications Session Border Controller in the role of an Access Oracle Communications Session Border Controller . In this example, you will configure additions to the ream configuration and the new external policy server configuration.

SIP Interface Configuration

To configure the SIP interface configuration for CLF support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure for CLF. This should be the ingress SIP interface for

   ```
   ORACLE(sip-interface)# select 1
   ORACLE(sip-interface)#
   ```

5. **ext-policy-svr**—Set this parameter to the same name as the External Policy Server configured that you configured for the CLF server.
6. **default-location-string**—Set this parameter to the default location string you want inserted into a P-Access-Network-Info header for when the CLF server does not return a unique location string.
7. Save your work using the ACLI **done** command.

   To configure the external policy server for use with a CLF:

8. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

9. Type **media-manager** and press Enter to access the media-related configurations.

```
ORACLE(configure)# media-manager
```

10. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

11. **name**—Set this parameter to an applicable name for this CLF instance of the external policy server. The value of this parameter will be entered in the SIP interface configuration element to reference this CLF.
12. **state**—Set this parameter to **enabled** to enable this CLF. The default value is **enabled**. The valid values are:
    - enabled | disabled
13. **operation-type**—Set this parameter to **admission-control** for the Oracle Communications Session Border Controller to communicate with a CLF. The default value is **disabled**.
14. **protocol**—Set this parameter to **COPS** to connect with a CLF via the COPS protocol. The default value is **C-SOAP**. The valid values are:
    - **COPS**—Standard COPS implementation. COPS client type is 0x7929 for CLF, and 0x7926 for PDP/RACF usage as defined in the operation-type parameter.
    - **A-COPS**—Vendor specific protocol. COPS client type is 0x4AC0 for admission-control operation-type.
15. **address**—Set this parameter to the IP address or FQDN of the CLF.
16. **port**—Set this parameter to the port which the CLF uses for COPS transactions. The standard port for COPS is **3288**. The default value is **80**. The valid range is:
    - Minimum—0
    - Maximum—65535
17. **realm**—Set this parameter to the realm in which the CLF exists.
18. **num-connections**—Set this parameter to the number of connections the Oracle Communications Session Border Controller will create with the CLF. The default value is **1**. The valid range is:
    - Minimum—0
    - Maximum—65535
19. **reserve-incomplete**—Set this parameter to **enabled** if you want the Oracle Communications Session Border Controller to send a COPS REQ message to the CLF that does not include the endpoint's true port number. A value of 0 will be used for the port number. The default value is **enabled**. The valid values are:
    - enabled | disabled
20. Save your work using the ACLI **done** command.

Application Type / Interface Matrix Reference

Bandwidth Management Applications

Standards Reference Point	SBC as	External Policy Server as	Protocol	Interface
TISPAN	P-CSCF	A-RACF	Diameter	Rq
3GPP (R15)	P-CSCF	PCRF	Diameter	Rx
3GPP (R6)	P-CSCF (AF)	PDF	Diameter	Gq
Packet Cable	P-CSCF (AF)	PDF	COPS	PKT-MM3
N/A	N/A	RACF	ACME COPS	N/A

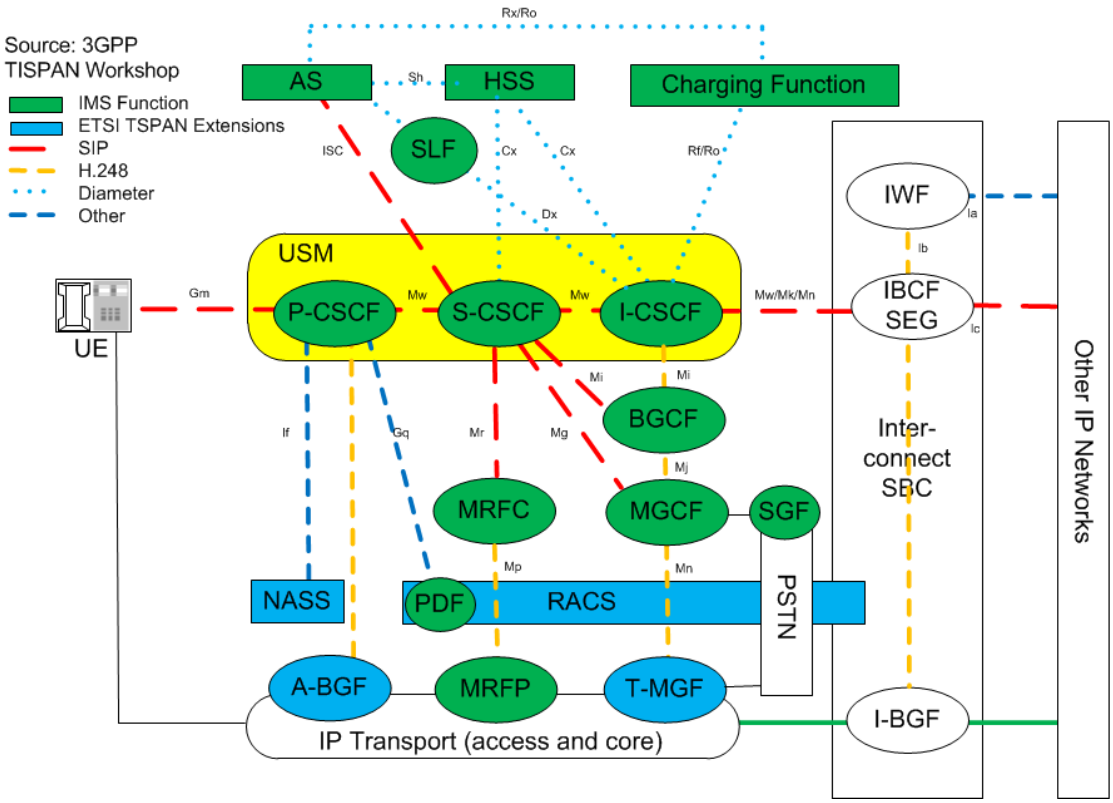
Emergency Location Services

Standards Reference Point	SBC as	External Policy Server as	Protocol	Interface
N/A	P-CSCF	CLF	Diameter	e2
N/A	N/A	CLF	ACME COPS	N/A

18

IMS Support

The ETSI TISPAN NGN defines several subsystems that make up the NGN architecture. The model for the target NGN architecture is depicted below. The Oracle Communications Session Border Controller is an integrated session control, policy enforcement and media management solution that incorporates functional components of the IP multimedia subsystem (IMS), the Resource and Admission Control Subsystem (RACS) and functions necessary for connecting with other IP networks/domains. The functions of the Oracle Communications Session Border Controller within the NGN architecture are divided into the interconnect border functions and the access border functions. The diagram below depicts the mapping of these functions across IMS architecture.



Oracle Communications Session Border Controller Access Border Functions

- Proxy CSCF (P-CSCF)
- Access/Core Border Gateway Function (A/C-BGF)
- RACF AF and SPDF functions

Oracle Communications Session Border Controller Interconnect Border Functions

- Interconnect Border Control Function (I-BCF)
- Interworking Function (IWF)
- Interconnect Border Gateway Function (I-BGF)

IMS Access Border Functions

The Oracle Communications Session Border Controller is deployed as the access point between the core IMS network and UEs to deliver the functions defined in the TISPAN architecture as the P-CSCF, and A-BGF. These two functions can not be separated. The Oracle Communications Session Border Controller performs the following functions as the Access Oracle Communications Session Border Controller:

P-CSCF Functions

The Oracle Communications Session Border Controller performs the following functions in the role of P-CSCF:

- Forwards SIP REGISTER messages and maintains a cached mapping of the user info and the UE's Address of Record (AoR), including the far-end NAT address in the case of hosted NAT traversal (HNT).
- Forwards SIP messages to a S-CSCF based on service route discovery procedures.
- Performs local emergency session handling—Local routing policy is used by the Oracle Communications Session Border Controller to identify emergency sessions and provide unique routing (e.g. can route to a dedicated S-CSCF function for emergency session handling).
- Operates as a UA (B2BUA) for generating independent SIP transactions for security purposes and handling of abnormal conditions.
- Offers current session timers which are used to monitor for media faults and abandoned calls.
- Generation of CDRs—The Oracle Communications Session Border Controller generates real-time accounting records via RADIUS.
- Authorization of bearer resources and QoS management—With integrated BGF capabilities, the Oracle Communications Session Border Controller allocates bearer resources (NAPT flows) and applies QoS policies (including packet marking) based on local policies and/or policies acquired via interaction with the A-RACF (PDF).
- Interaction with the A-RACF (PDF) for session-based policy enforcement and admission control—The Oracle Communications Session Border Controller PDF interface options include COPS and SOAP/XML.
- Traffic Policing—Traffic is policed at the session and media/transport layer. At the signaling layer, the Oracle Communications Session Border Controller polices at a number of levels including:
  - Capacity—Total number of concurrent calls to/from each realm
  - Session set-up rate—Maximum rate of call attempts to/from each signaling element

  - Signaling message rate—Each endpoint's signaling message rate is monitored and policed
  - Signaling bandwidth—each endpoint's signaling bandwidth is policed individually

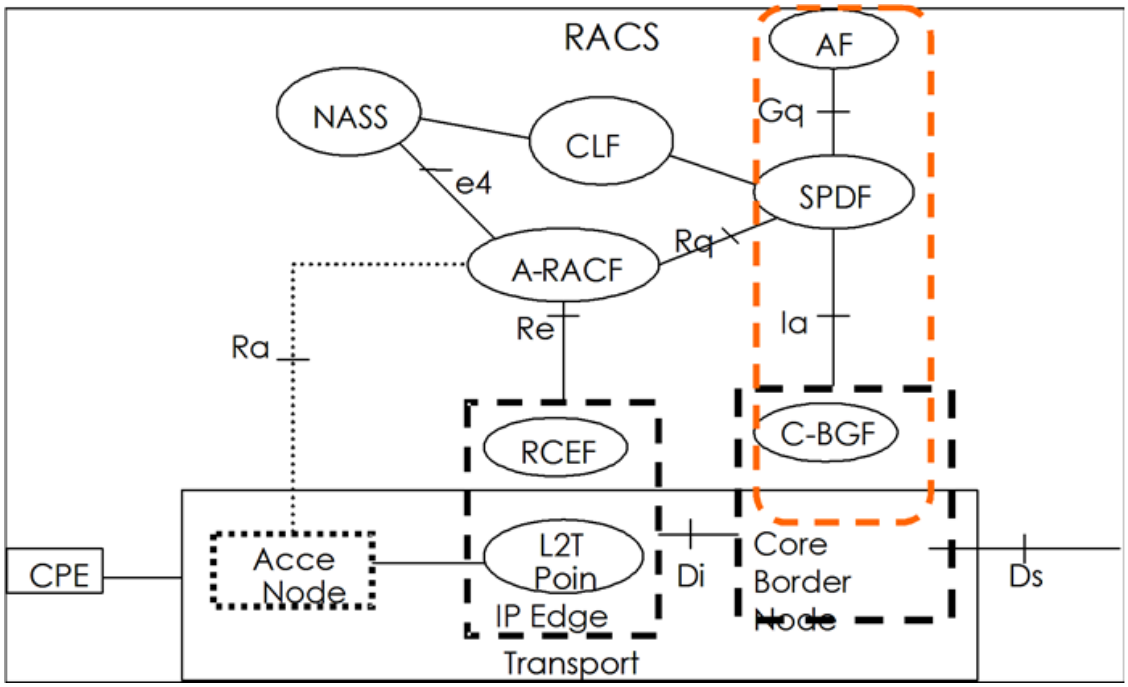
A-BGF Functions

The Oracle Communications Session Border Controller performs the following IMS BGF functions:

- Opening and closing gates/packet filtering—The Oracle Communications Session Border Controller opens and closes gates (media pinholes) on a session-by-session basis. Packet filtering rules include full source and destination IP address and port number.
- Per-session DiffServ or ToS marking—Media flows destined for the IMS core network can be explicitly marked using ToS or DiffServ. Media packets can be marked by VPN, by codec (voice, video) or by E.164 phone number prefix.
- NAPT-PT and topology hiding—The Oracle Communications Session Border Controller provides NAPT for all media flows associated with a session on a per session-basis. Double NATing, NATing both source and destination sides, is utilized to fully hide topology in each direction for RTP and RTCP. Local IP addresses and port resources are dynamically allocated from steering pools provisioned on the Oracle Communications Session Border Controller.
- Hosted NAT traversal—The Oracle Communications Session Border Controller supports HNT function that allows media flow traversal through the CPE firewall/NAT without upgrading the CPE equipment. The system interacts with the endpoints to dynamically establish and maintain bindings in the CPE firewall/NAT that allow the signaled communications to pass through. The Oracle Communications Session Border Controller's registration management and media relay functions make CPE-based NATs transparent to the service delivery elements.
- Traffic Policing—Traffic is policed at the session and media/transport layer. At the signaling layer, the Oracle Communications Session Border Controller polices at a number of levels including:
  - Policing of Media (e.g. RTP & RTCP) traffic on a per-flow basis—CBR policing is applied to each flow based on negotiated offered and negotiated media codecs.

Resource and Admission Control (RACS) Functions

The figure below illustrates the mapping of Oracle Communications Session Border Controller functions to the RACS functional model. In this model, the Oracle Communications Session Border Controller incorporates the Application Function (in the case of IMS this is the P-CSCF function), the SPDF (Service Policy Decision Function) and the Core Border Gateway function.



The Oracle Communications Session Border Controller, acting as the SPDF, interfaces with the PDF (A-RACF policy decision function) for resource authorization and admission control on a call-by-call basis. COPS is the supported PDF interface.

IMS Interconnect Border Functions

The Oracle Communications Session Border Controller is deployed at IP interconnect points between service providers to deliver the functions defined in the TISPAN architecture as the I-BCF, IWF and I-BGF. The Oracle Communications Session Border Controller performs the following functions as the Interconnect border Oracle Communications Session Border Controller:

Interworking Function (IWF)

- Interworking SIP profiles and other protocols (e.g. H.323)

Interconnect Border Control Function (I-BCF)

- Interaction with I-BGF (including NAPT and firewall functions)
- Insertion of the IWF when appropriate
- Topology hiding—screening of signalling information

Interconnect-Border Gateway Function (I-BGF)

- Gate opening/closing
- NAPT and packet filtering
- Packet marking
- Resource allocation and bandwidth reservation

- Security and topology hiding
- Session admission control, resource and traffic management
- Upstream/downstream flow policing
- Quality monitoring and reporting
- Usage metering - CDR generation
- Lawful Intercept

IMS Path and Service Route Header Support

The Oracle Communications Session Border Controller supports the Path header and the Service-Route header used in the registration phase of a SIP transaction. The Oracle Communications Session Border Controller will learn the route vectors from the SIP URIs contained in these headers in order to preload SIP headers with the correct route vectors in subsequent SIP message exchanges between the UA and the S-CSCF across the Oracle Communications Session Border Controller. This is how the Oracle Communications Session Border Controller supports RFC 3608 and RFC 3327.

Path Header

When a UE registers to an S-CSCF, the Oracle Communications Session Border Controller adds the Path header in the REGISTER message as it is proxied to the S-CSCF. The Path header includes the SIP URIs that form the route vector which describes how the UE reaches the Oracle Communications Session Border Controller, through a specific series of proxies. This route vector is saved in the Oracle Communications Session Border Controller's registration entry for the UE, routing all subsequent SIP messages from the S-CSCF to the UE. As the Path header is sent to the S-CSCF, the Oracle Communications Session Border Controller, as P-CSCF, inserts the SIP URI of itself as the top entry in the Path header.

The Path header only appears in SIP messages exchanged during the registration process.

If the REGISTER request already contains a Path header, the Oracle Communications Session Border Controller stores the contents of the Path header(s) for this endpoint for routing back to the endpoint in subsequent messages.

Service Route Header

When a UE registers through the Oracle Communications Session Border Controller to the registrar, the registrar returns a Service-Route header in a 200 OK message in response to the REGISTER message to the UE. This header contains the route vector that directs traffic through a specific sequence of proxies used to reach the S-CSCF. The Service-Route header only appears during the SIP registration process.

The P-CSCF (Oracle Communications Session Border Controller) will now store the URIs listed in the Service-Route header(s) in the registration entry of the UE for use in routing subsequent traffic to the S-CSCF. The Oracle Communications Session Border Controller inserts this sequence of proxies into and outgoing message’s Route headers; this is called a pre-loaded route. This route is only applicable for the traffic flowing between the originating UE and the contacted S-CSCF.

When receiving subsequent requests from the UE, the Oracle Communications Session Border Controller looks at the UE's registration entry for a service route, and will insert the route vector as appropriate Route headers. If the service route is not found in the registration entry, the routing is performed in the usual fashion.

As an exception, you may wish for the Oracle Communications Session Border Controller to not use the Service-Route header to route subsequent Register requests.

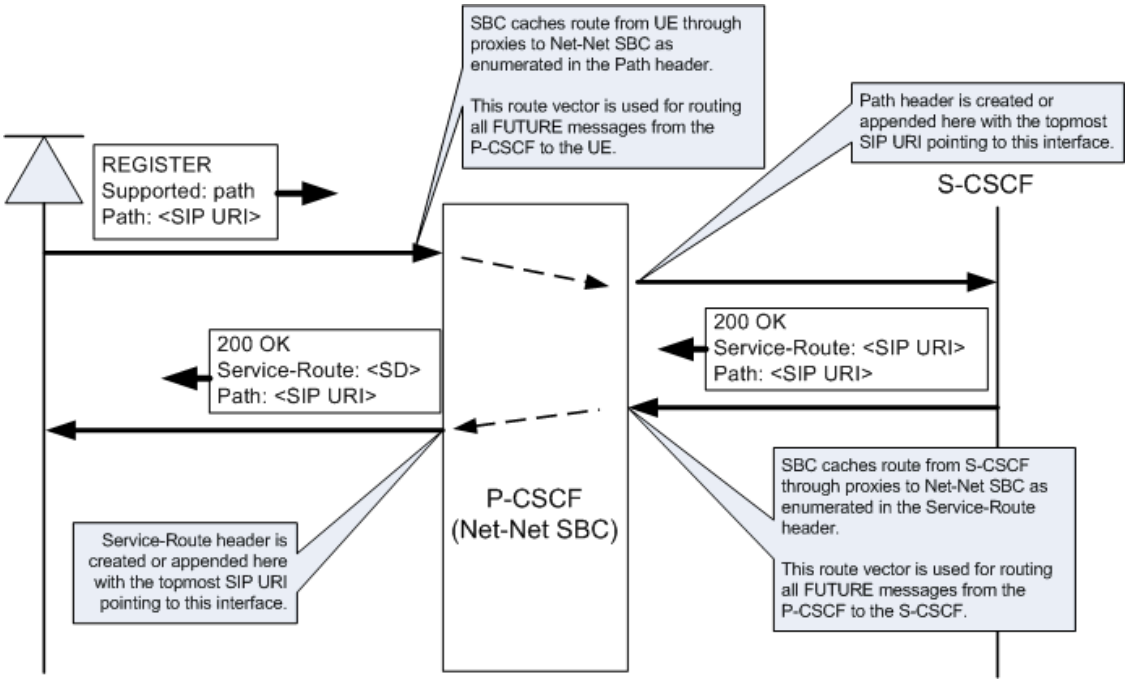
> **Note:**
>
> In the configuration section the way to disable Service-Route header routing.

The manner in which passing or stripping of Service-Route headers sent from the S-CSCF is done is determined by local configuration on the Oracle Communications Session Border Controller. There is no verification of configured local policy against the route included in the stored service route. The Service-Route header, as created by the Oracle Communications Session Border Controller, and exiting back to the UE, contains a SIP URI pointing to itself as the topmost entry. This is used so that other proxies can learn the route back to the Oracle Communications Session Border Controller.

Summary

If a request originates at the UE, the routes enumerated in the Service-Route header are used to route the request to the S-CSCF. If a request is meant to terminate at a UE, the routes enumerated in the Path header are used to route the response to the UE. Service-Route routes take priority over configured local policy.

Path headers received in a 200 OK response from the registrar are transmitted to the UE unchanged. If you want them stripped as the SIP message leaves the Oracle Communications Session Border Controller, you can use the SIP Header Manipulation function.



IMS Path and Service Route Headers Configuration

IMS and all related functions must be enabled on both the access-side and core-side SIP interfaces. Only IMS features discussed up to this point are enabled by the following procedure.

To enable RFC 3608 and RFC 3327 support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure.

   ```
   ORACLE(sip-interface)# select 1
   ```

5. **sip-ims-feature—** Enable IMS functionality on this SIP interface. The default value is **disabled**. Valid values are:

   - enabled | disabled

     ```
     ORACLE(sip-interface)# sip-ims-feature enabled
     ```

     This completes enabling IMS for a given SIP interface.

     If you wish to disable subsequent routing of Register messages via the Service-Route header, type route-register-no-service-route and press Enter.

6. Save your work using the ACLI **done** command.

Network Provided Location Information During Registration

For most cases, location information is relevant at the time of the session request. However, Network Provided Location Information (NPLI) upon REGISTER is required for some Authorization-Authentication Requests (AAR) and Authorization-Authentication Answers (AAA).

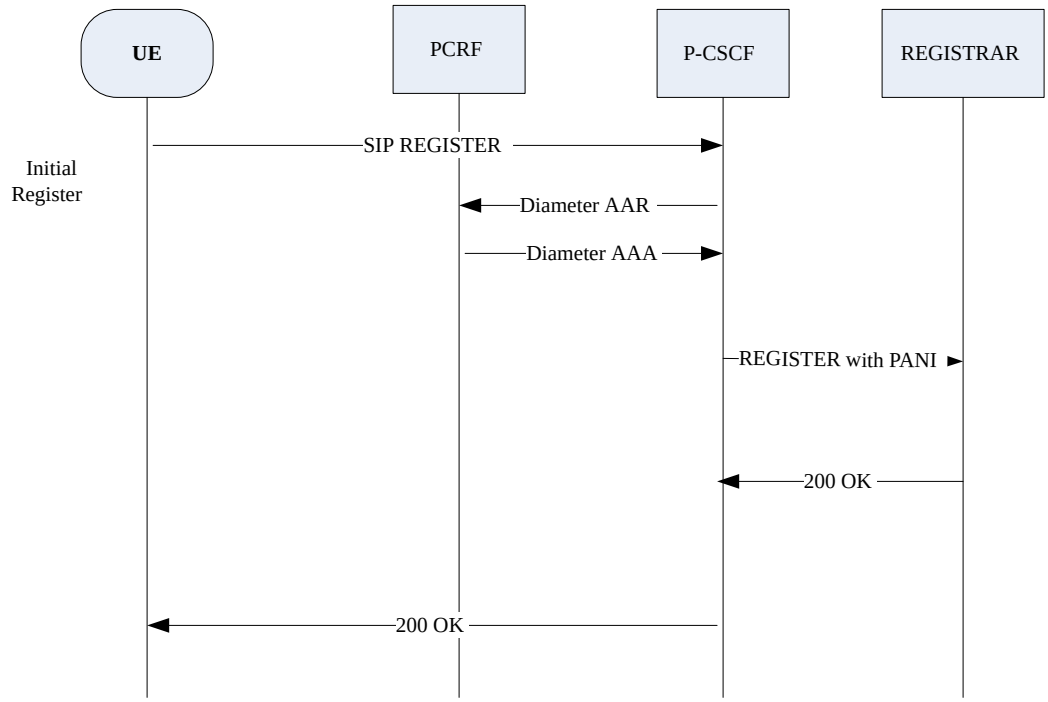
The access awareness feature in the Serving Call Session Control Function (S-CSCF) uses the P-Access-Network-Info header in the initial REGISTER request for selecting the registration and authentication profile. This in turn depends on the access class of the IP Connectivity Access Network (IP-CAN) being used by the user equipment. For that reason, the Oracle Communications Session Border Controller Proxy Call Session Control Function (P-CSCF) requires that the Authorization-Authentication Answer (AAA) from the Policy Charging and Rules Function (PCRF) to contain the Radio Access Technology (RAT-type) Attribute

Value Pair (AVP). It uses this information to populate the P-Access-Network-Info (PANI) header in the forwarded REGISTER request according to that value. The **npli-upon-register** parameter must be set to **enabled** for this behavior.

Example Call Flows

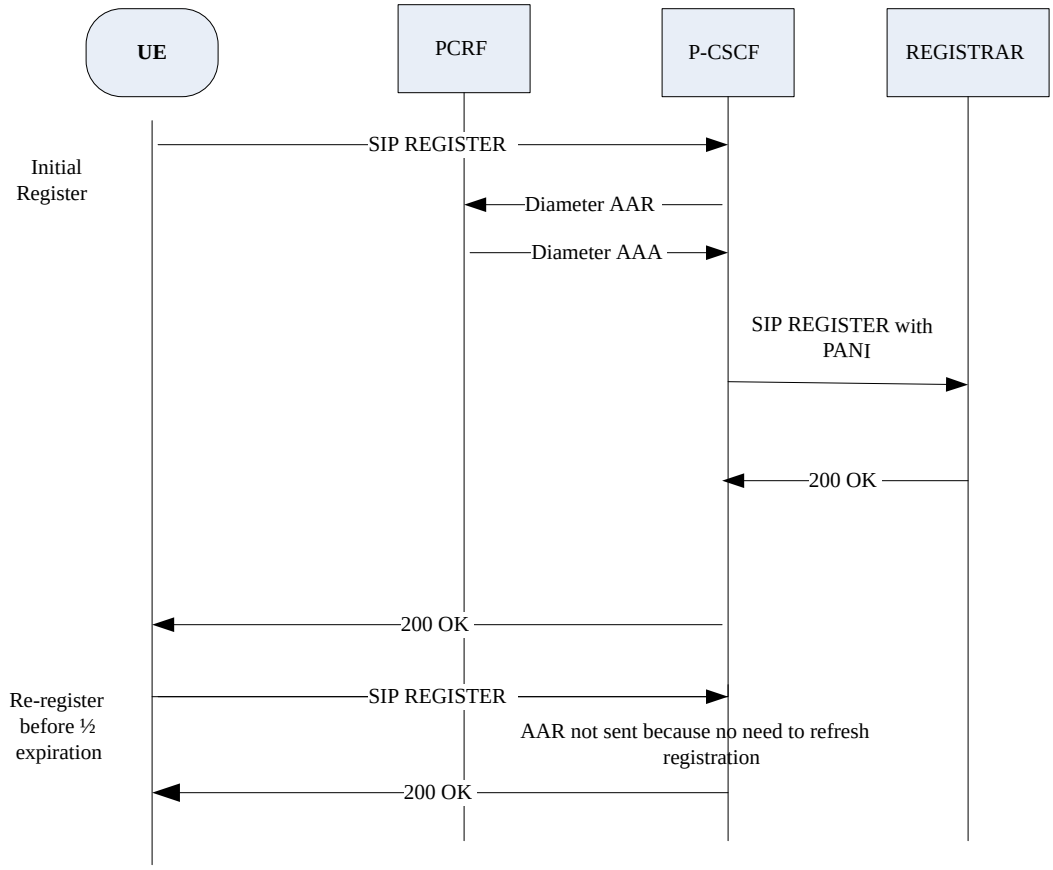
When user equipment (UE) registers for the first time the Oracle Communications Session Border Controller buffers the incoming REGISTER and sends an AAR to the PCRF. The AAA from the PCRF contains a 3GPP-User-Location Info AVP: IP-CAN-Type and RAT-Type AVP. The Oracle Communications Session Border Controller uses those AVPs to populate a PANI header in the REGISTER that is forwarded to the Registrar.

Figure 18-1 Initial Registration Flow



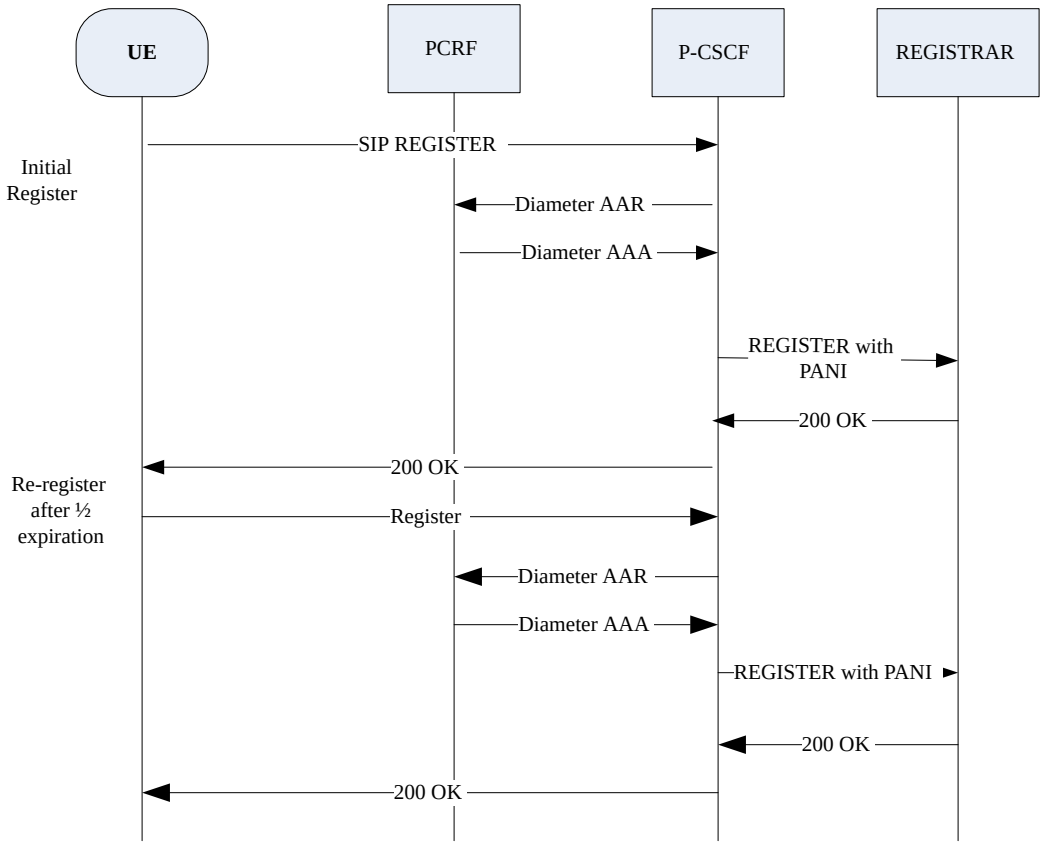
If a registration refresh is received before the half time of the registration expiration interval then the registration cache is not updated and the Oracle Communications Session Border Controller sends a 200 OK to the UE. No AAR is sent to the PCRF.

Figure 18-2 Re-Register before Half Time of Expiration



If a registration refresh is received after the half time of the registration expiration interval or if any registration information is changed then the Oracle Communications Session Border Controller will send an AAR to the PCRF. The AAA response from the PCRF contains a 3GPP-User-Location Info AVP: IP-CAN-Type and RAT-Type AVP. The Oracle Communications Session Border Controller will use the AVPs to populate a PANI header in the REGISTER that is forwarded to the registrar.

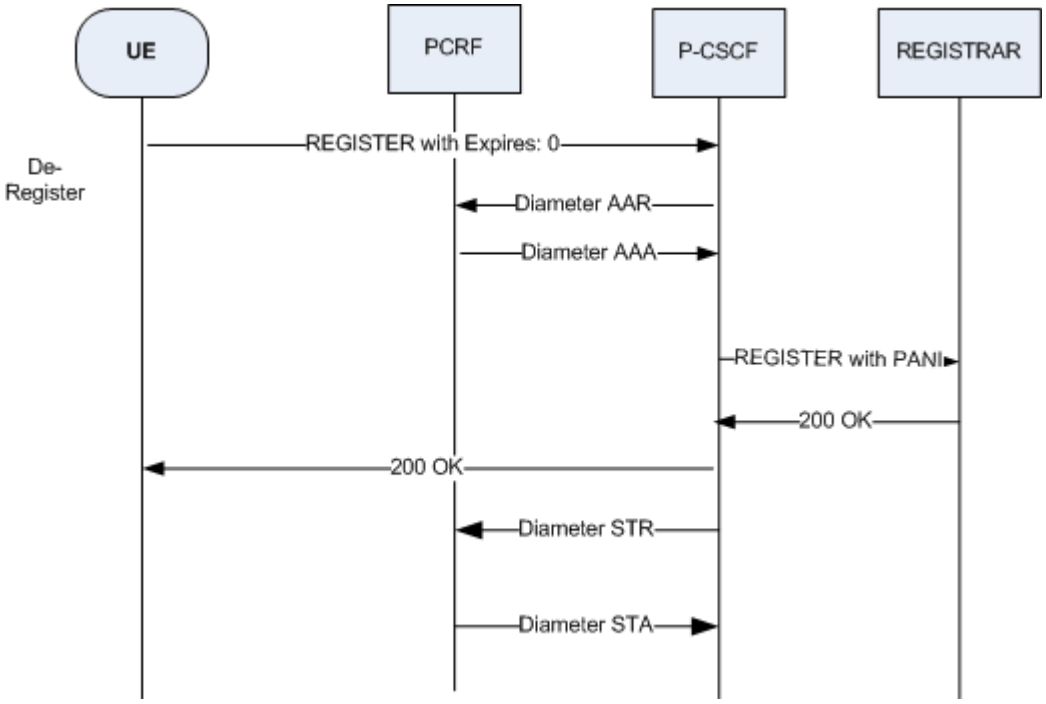
Figure 18-3 Re-Register after Half Time of Expiration



De-Register

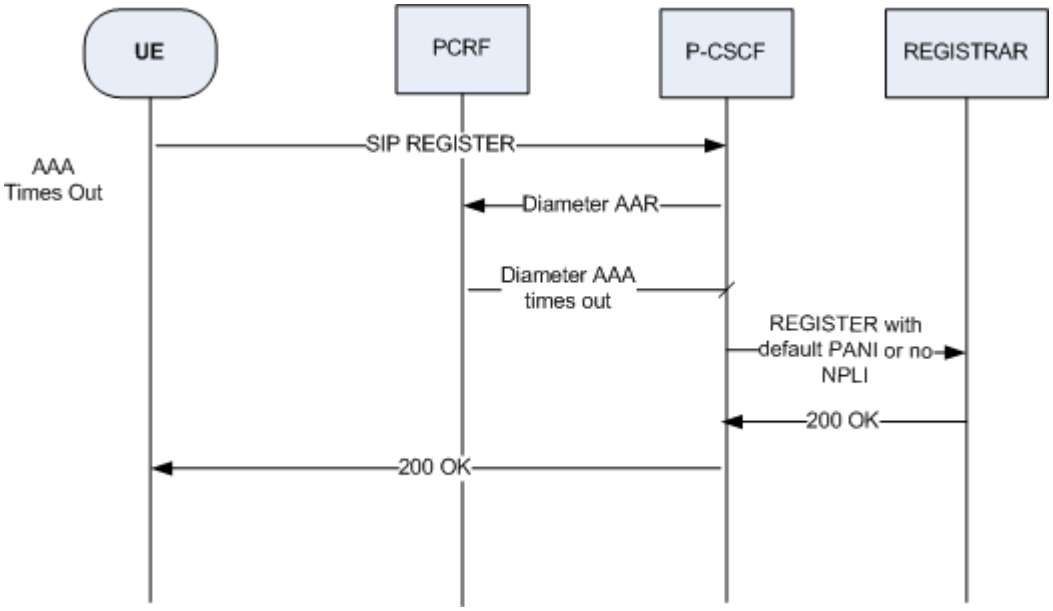
When an endpoint removes all of its contacts from registration by sending a REGISTER with **expires=0**, the Oracle Communications Session Border Controller will buffer the incoming REGISTER and send an AAR to the PCRF. The AAA response from the PCRF contains a 3GPP-User-Location Info AVP: both IP-CAN-Type and RAT-Type AVP. The Oracle Communications Session Border Controller will use the AVPs to populate a PANI header in the REGISTER that is forwarded to the Registrar. After the Oracle Communications Session Border Controller receives the 200 OK response to the REGISTER, it will send a STR message to the PCRF to terminate the session.

Figure 18-4 De-Registration



If the AAA is not received before the timeout, the AAA returns with an error, or the AAA is missing the required location information, then the Oracle Communications Session Border Controller will search the NPLI cache for cached location information to construct the PANI header to forward to the Registrar. If there is no cached location information, then the *default-location-string* configuration parameter will be used to create the PANI header. The *default-location-string* is obtained from the configuration in either the Realm or the SIP interface, with the Realm taking precedence; in both cases the value will be taken from the Access Realm. If the *default-location-string* is not set, then the REGISTER will be forwarded to the core without a PANI header.

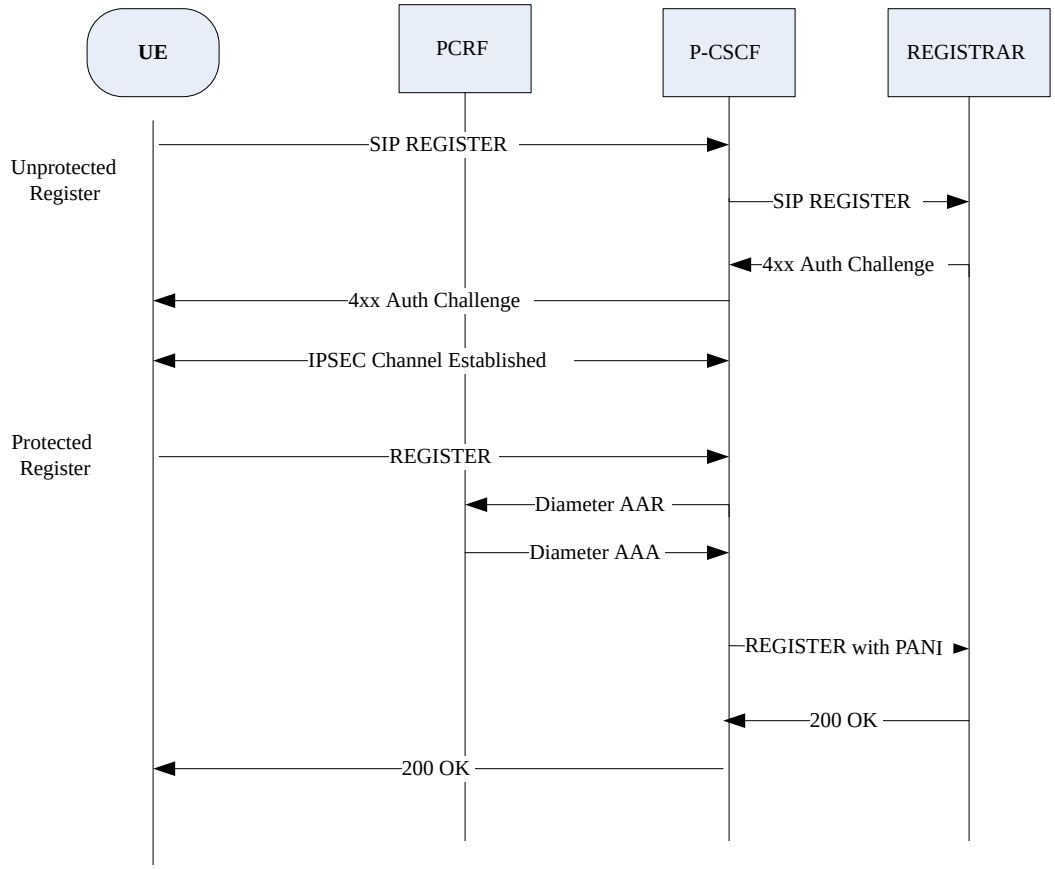
Figure 18-5 AAA Timeout/Error or AAA does not have Location Information



Register with IMS-AKA Enabled

If the IP Multimedia Subsystem Authentication and Key Agreement (IMS-AKA) is enabled then when a UE registers for the first time, the Oracle Communications Session Border Controller will forward the unprotected REGISTER to the Registrar without triggering an AAR. The Registrar will reply with a 401 Authentication Required response. The UE and the PCRF will use the information in the 401 response to create a secure channel. The UE will then send a protected REGISTER that will trigger the Oracle Communications Session Border Controller to send an AAR to the PCRF. The AAA response from the PCRF contains a 3GPP-User-Location Information AVP: both IP-CAN-Type and RAT-Type AVP. The Oracle Communications Session Border Controller will use those AVPs to populate a PANI header in the REGISTER that is forwarded to the registrar.

Figure 18-6 Register with IMS-AKA Enabled



Network Provided Location Information upon Register Configuration

Enable the**npli-upon-register** parameter to allow the capture of Network Provided Location Information during the registration process.

1. Access the **sip-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. Select the **sip-config** object to edit.

```
ORACLE(sip-config)# select


ORACLE(sip-config)#
```

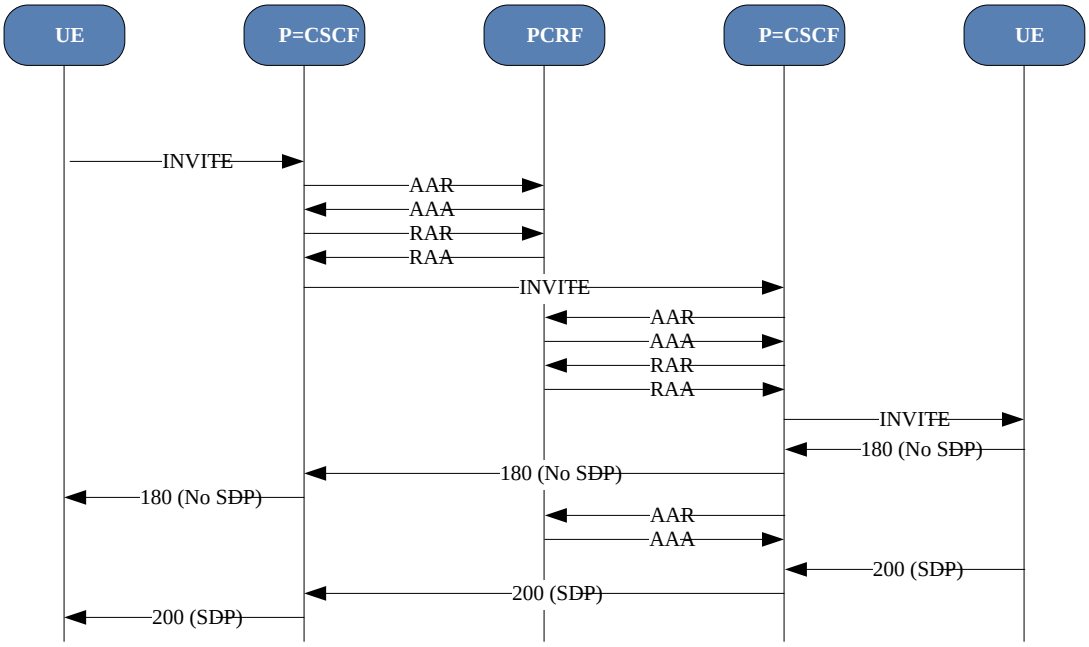
3. **npli-upon-register**— Enable to allow capture of Network Provided Location Information (NPLI) during the registration process
   - **enabled | disabled**
4. Type **done** to save your configuration.

Adding NPLI to Interim CDRs

You can configure the Oracle Communications Session Border Controller (SBC) with a trigger to generate an INTERIM Call Detail Record (CDR) during applicable originating and terminating Voice over LTE and WiFi call flows when the INVITE egresses the system. This CDR contains Network Provided Location Information (NPLI) information received from the Policy and Charging Rules Function (PCRF) or other source that may be more accurate than network location information presented in the start CDR. When configured, the SBC adds this NPLI to RADIUS, DIAMETER, and/or local CSV CDRs.

A typical call flow includes the P-CSCF function retrieving policy information, including NPLI, from the PCRF via DIAMETER. This happens during the authorization/authentication (AAR/AAA) and, with the exception of WiFi, re-authorization request/answer (RAR/RAA) sequences over the Rx interface.

In the following diagram, the SBC is performing the P-CSCF function. The timing of these exchanges affects the contents of CDRs, wherein the start record may have 'stale' location information, but an interim record sent after the egress INVITE can include updated location information from the PCRF.



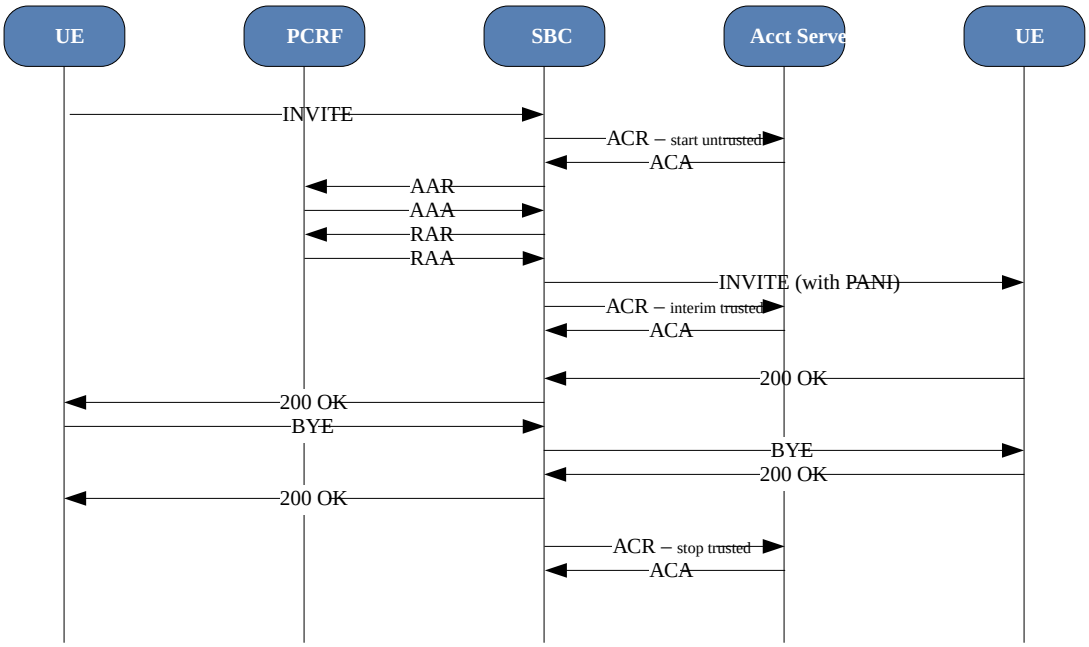
Having obtained updated NPLI, the SBC distributes and/or records NPLI using the following protocol objects:

- DIAMETER — The system provides NPLI using the **Access-Network-Information** AVP (code 1263) in the ACR (Rf interface), which is of type OctetString. Per 3GPP TS 32.299, this AVP indicates the SIP P-header, **P-Access-Network-Information**.
- RADIUS — The system provides NPLI using the **Access-Network-Information** VSA in the CDR (code 248) in the Accounting-Request message.
- Local CSV CDR Files — The system records NPLI using the Access Network Information entry, ACME 248, in Start, Interim, and Stop records.

You configure the SBC to produce these interim CDRs using the **account-config**'s **generate-interim**'s parameter. Set this parameter to **Egress-Invite**.

```
ORACLE(account-config)# generate-interim Egress-Invite
```

When the interim CDR is set to **Egress-Invite**, and all other parameters are at their default, the applicable SIP and DIAMETER signaling call flow would appear similar to the flow shown below. Assuming the PCRF provides NPLI in the authentication exchange, the SIP signaling would include the NPLI in the PANI header in both the interim and stop CDRs.



When configured, the system provides NPLI in interim CDRs for the following call types, based on the SBC's perspective:

- VoLTE originating
- VoLTE terminating
- VoLTE WiFi originating — RAR is not relevant
- VoLTE WiFi terminating — RAR is not relevant

Other configuration parameters that can impact this NPLI inclusion behavior include:

- **reserve-incomplete**— When enabled, the SBC sends an AAR when it receives the SDP offer. This provides an opportunity for the egress-invite ACR to contain the AAA NPLI, assuming it was present in the AAA. When disabled, the SBC sends and AAR only after the SDP answer. This means the egress-invite ACR would not contain AAA NPLI, instead using the SBC's default-location-string, assuming it is configured.
- **hold-invite-calls-for-loc-info**— The **hold-invite-calls-for-loc-info** attribute, in the **sip-config**, is the time the systems waits for the RAR, assuming it is greater than 0 and **reserve-incomplete** is enabled.
- **intermediate-period**— If **intermediate-period** attribute set to > 0, the SBC generate interim CDRs periodically for SIP calls, all of which would include this NPLI.

- **npli-upon-register**— If enabled, the SBC sends an AAR to the PCRF when it receives REGISTER messages. The SBC stores any NPLI received in its registration cache and uses that information for all relevant purposes even if it is stale.
- **cache-loc-info-expire**— Specifies the timer (default of 32 seconds) when the SBC removes the NPLI from its NPLI cache. The system would not include NPLI in any context until it receives another applicable RAR.
- **hold-invite-calls-for-loc-info**— Specifies the wait timer that the system waits for RAR. If this expires, the system inserts the default location string defined in the applicable sip-interface as NPLI in CDRs.

The system writes a DEBUG level log message in log.sipd when NPLI is inserted into a CDR for debug purpose.

```
Aug 9 19:25:01.803
[SESSION] (3) GenerateStart: adding NPLI string
3GPP-GERAN;network-provided
Aug 9 19:25:45.525
[SESSION] (3) GenerateInterim: adding NPLI string
3GPP-GERAN;network-provided
Aug 9 19:26:45.267
[SESSION] (3) GenerateStop: adding NPLI string
3GPP-GERAN;network-provided
```

This configuration does not require a reboot, and is supported by HA deployments.

Network Provided Location Information for Short Message Service

For most cases, location information is relevant at the time of the session request. Network Provided Location Information (NPLI) for SIP is delivered by DIAMETER in Authentication-Authorization Answers (AAA) and Re-authentication-Authorization Requests (RAR). In certain countries, it is a regulatory requirement to provide location information also for the Short Message Service (SMS), which in LTE networks is implemented using the SIP MESSAGE method to carry the text.

The access awareness feature in the Serving Call Session Control Function (S-CSCF) uses the P-Access-Network-Information (PANI) header in the initial MESSAGE request for selecting the registration and authentication profile RAT-type and User Location Information. This in turn depends on the access class of the IP Connectivity Access Network (IP-CAN) being used by the user equipment (UE). For that reason, the Oracle Communications Session Border Controller Proxy Call Session Control Function (P-CSCF) requires that the Authorization-Authentication Answer(AAA) from the Policy Charging and Rules Function (PCRF) to contain IP-Can, the Radio Access Technology (RAT-type) and user location information attribute value pairs (AVPs). All three values are used to populate the PANI header in the forwarded MESSAGE request according to that value.

The Oracle Communications Session Border Controller P-CSCF will map User Location Information AVP and RAT type AVP into the PANI header for all subsequent SIP messages towards the core and acknowledge the RAA to the PCRF. The **np** parameter will be added to the PANI header to indicate a PANI header field is provided by a network element. This content can differ from a PANI header field without this parameter, which is provided by the UE. In the case the received message by the P-CSCF already contains information provided by the UE, that information will be preserved if **include-ue-loc-info** is enabled.

If the location information is not received from PCRF before the holding time expires, the Oracle Communications Session Border Controller will use the default location string if it is present as **default-location-string** in either the SIP Interface or Realm configuration.

Registration caching has to be enabled, e.g. **registration-caching=enabled**, for this feature to work. NPLI will be cached on a per-contact basis.

The **msghold-for-loc-info** object must be set to a non-zero value for this behavior. The location information will be held for no longer than the value set in **cache-loc-info-expire** unless the **keep-cached-loc-info-after-timeout** option is set

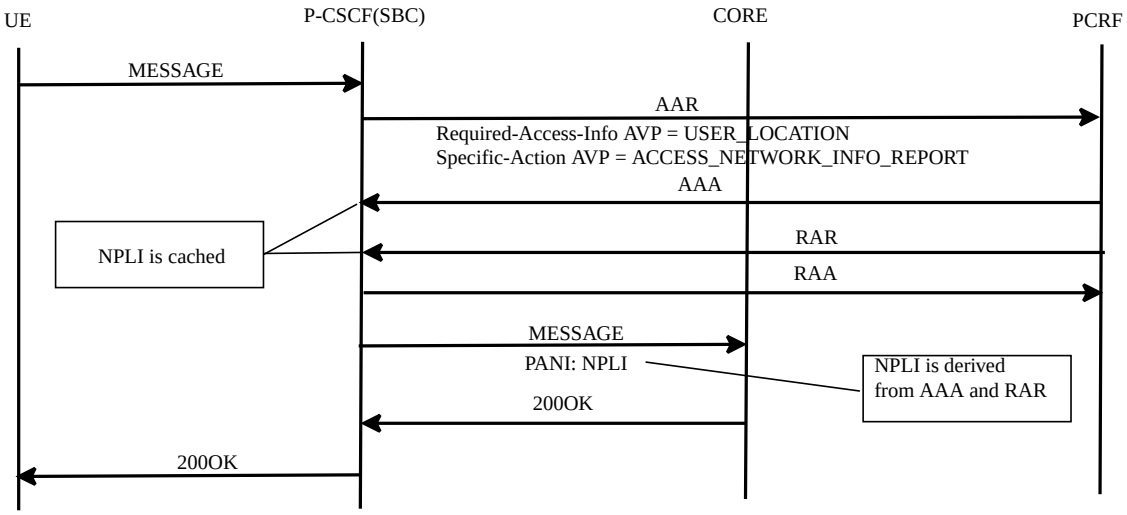
If this feature is disabled (i.e. **msg-hold-for-loc-info** = 0), SMS messages will not be held for NPLI and PANI headers will not be present in the SMS MESSAGEs. There is no need to set **cache-loc-info-expire** in this case.

NPLI for Short Message Service Examples

The following scenarios illustrate theOracle Communications Session Border Controller behavior when the Network Provided Location Information(NPLI) is not present in cache or the cached entry has expired. There are two types of scenarios: mobile originated(MO) and mobile terminated(MT). Descriptions precede each diagram.

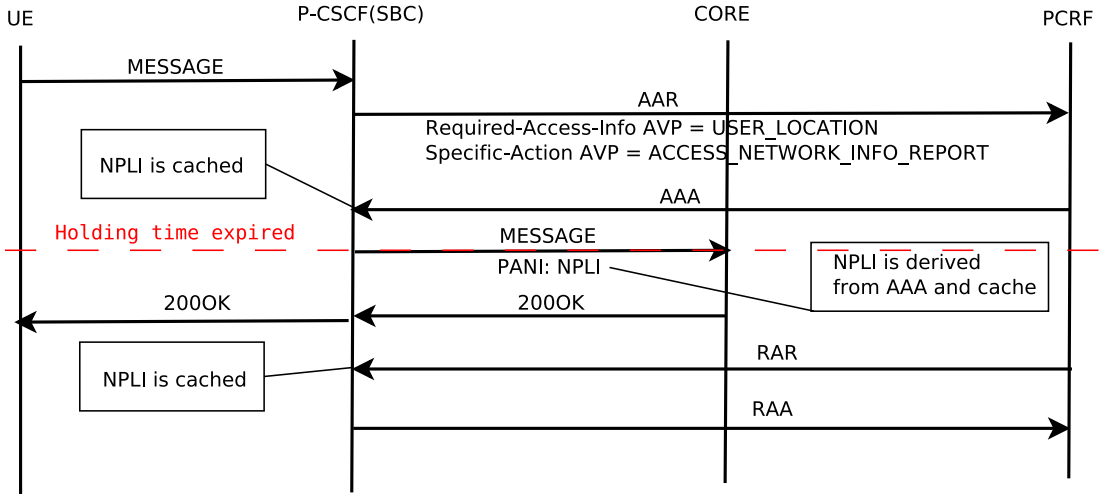
MO message with NPLI information provided in the Authorization-Authentication Request and Answer(AAR/AAA) message and later updated with the Reauthorization-Reauthentication Request and Answer(RAR/RAA) message. Both AAA and RAR arrive within the holding period. The NPLI value is present in the P-Access Network Information (PANI) header.

Figure 18-7 MO Message with AAR/AAA and RAR/RAR and within Holding Period



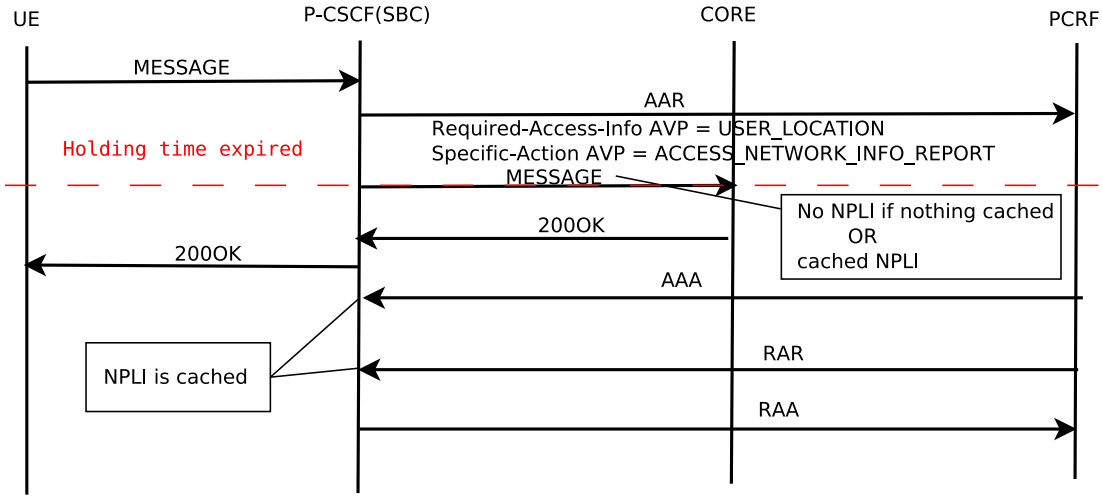
The below example shows a MO MESSAGE with NPLI information provided in the AAA, but the RAR arrives after the holding period expires. The NPLI value is present in the PANI header

Figure 18-8 MO Message with AAR/AAA but RAR arrives after Holding Period



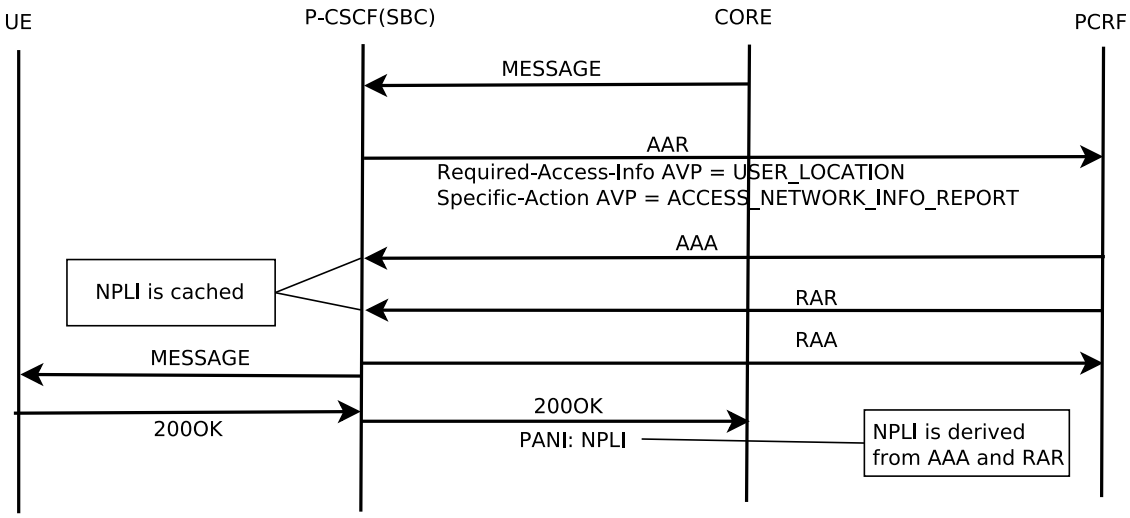
MO MESSAGE with no NPLI or cached NPLI. Both AAA and RAR arrive after the holding period expires.

Figure 18-9 MO Message with no/cached NPLI; both AAR and RAR arrive after Holding Period



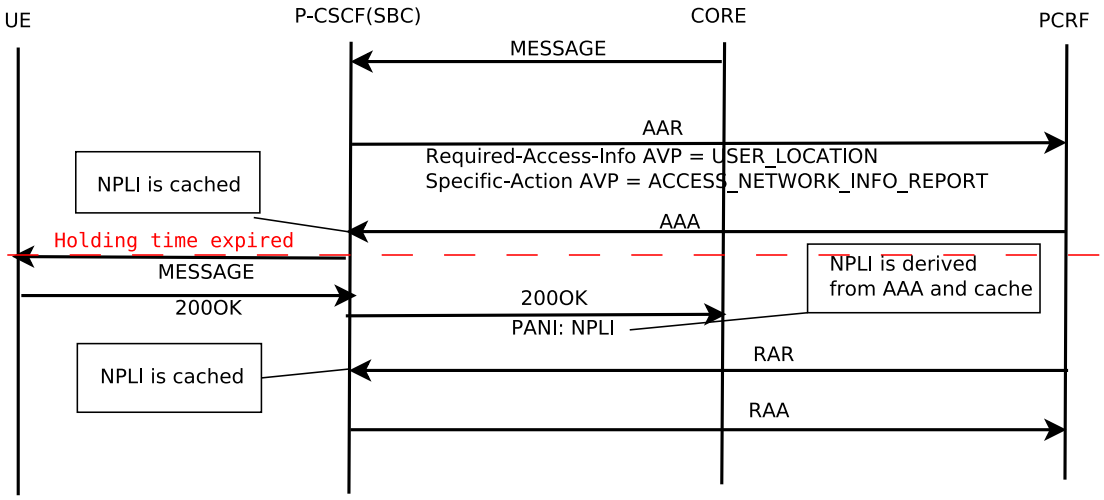
MT MESSAGE with NPLI provided in AAA and later updated with RAR. Both AAA and RAR arrive within the holding period. The NPLI value is present in the PANI header in 200OK response.

Figure 18-10 MT Message with NPLI in both AAR/AAA and RAR/RAA within the Holding Period



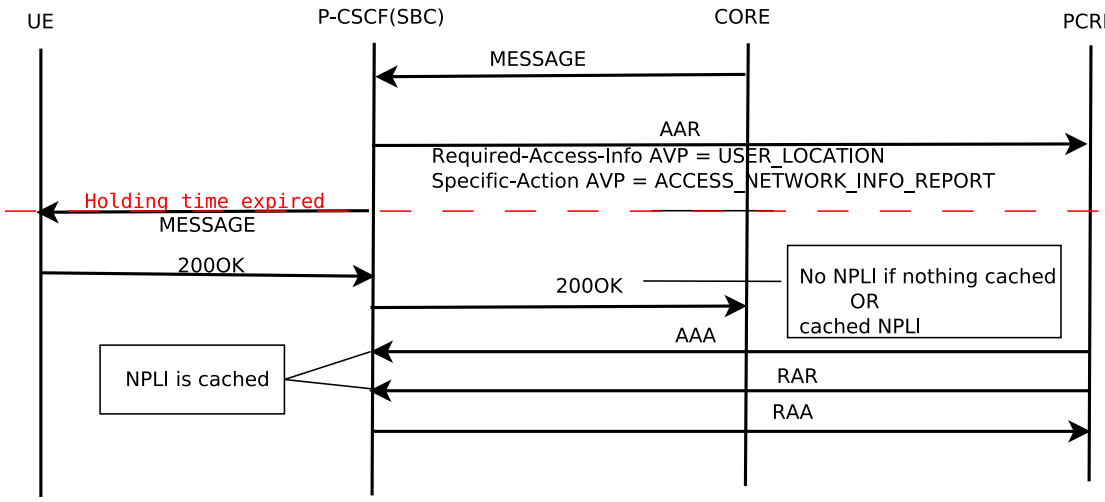
MT MESSAGE with NPLI provided in AAA, RAR arrives after the holding period expires. The NPLI value is present in the PANI header in 200OK response.

Figure 18-11 MT Message with NPLI provided in AAR/AAA, RAR arrives after Holding Period



MT MESSAGE with neither NPLI nor cached NPLI. Both AAA and RAR arrive after the holding period expires.

Figure 18-12 MT Message with no NPLI; both AAA and RAR arrive after the Holding Period

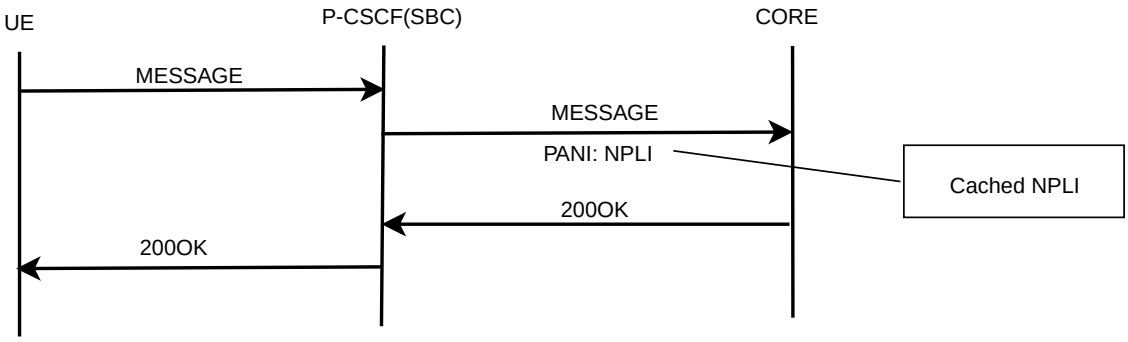


Network Provided Location Information present in Cache Examples

The following scenarios describe the Oracle Communications Session Border Controller behavior when the Network Provided Location Information (NPLI) is present in the cache.

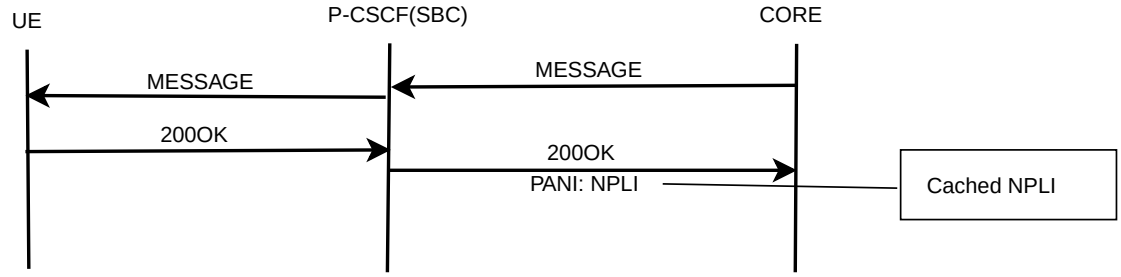
MO MESSAGE cached NPLI value present in PANI header as the cached NPLI is current. Note that there is no AAR/AAA/RAR/RAA exchange.

Figure 18-13 MO Message with cached NPLI in PANI Header



Mobile terminated(MT) MESSAGE cached NPLI included in PANI header as the cached NPLI is current. Note that there is no AAR/AAA/RAR/RAA exchange

Figure 18-14 MT Message with cached NPLI in PANI header



NPLI for Short Message Configuration

Set to create a new or update the existing P-Access-Network Information (PANI) header based on the Network Provided Location Information (NPLI): values IP Connectivity Access Network (IP-CAN) , RAT-Type and user-location-information for SMS.

1. Access the **sip-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. Select the **sip-config** object to edit.

```
ORACLE(sip-config)# select


ORACLE(sip-config)#
```

3. **msg-hold-for-loc-info**— maximum number of seconds that the Oracle Communications Session Border Controller will hold MESSAGEs for location information.
   - **1-30**— seconds
   - **0**— disabled; default
4. **cache-loc-info-expire**— number of seconds after which the Oracle Communications Session Border Controller will drop network location information.
   - **1-4294967295**— seconds
   - **32**— default
5. **keep-cached-loc-info-after-timeout**— If this option is enabled, the location information will be left in the cache and used in subsequent MESSAGEs after the **cache-loc-info-expire** time expires.

   options +keep-cached-loc-info-after-timeout
6. Type **done** to save your configuration.

Wildcard Public User Identity (PUI)

The Oracle Communications Session Border Controller supports wildcard Public User Identies (PUI). This capability is most often used to streamline processing of REGISTER and INVITE messages between a PBX and an IMS core.

The HSS (Home Subscriber Server, an IMS network-wide database) is pre-provisioned with the extension numbers associated with the PBX's base telephone number. When the PBX registers its own base telephone number with the S-CSCF, that server downloads a wildcarded PUI -- a regular expression that describes all extension numbers associated with the registering PBX

The S-CSCF returns the wildcarded PUI in the 200 OK response to the REGISTER message. Upon receiving the 200 OK, the SBC (acing in the P-CSCF role) uses the wildcarded PUI to construct a registration cache entry to implicitly treat subsequent INVITEs from PBXs extensions that match the wildcard as registered endpoints.

A wildcarded PUI consists of a delimited regular expression located either in the user info portion of the SIP URI or in the telephone-subscriber portion of the Tel URI. The regular expression takes the form of an Extended Regular Expressions (ERE) as defined in chapter 9 of The Single UNIX Specification (IEEE 1003.1-2004 Part 1). The exclamation mark (!) serves as the delimiter.

For example, the following PUIs will match to the wildcard ERE "sip:chatlist!.*!@example.com".

```
sip:chatlist1@example.com
sip:chatlist2@example.com
sip:chatlist42@example.com
sip:chatlistAbC@example.com
sip:chatlist!1@example.com
```

In addition to the ability of the P-CSCF to recognize and parse a wildcarded PUI, The P-CSCF must also support the PATH and P-Profile-Key SIP headers.

Path Header

The Path header is defined in RFC 3327, Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts. The Path header enables the accumulation and transfer of a list of proxies between a SIP UA and a REGISTER. Its operation is very similar to the Record-Route header in dialog initiating request. Within the context of support for wildcarded PUIs, the Path header is used to ensure that REGISTER responses containing an wildcard ERE transit the P-CSCF, thus enabling the implicit registration of PBX extensions.

P-Profile-Key Header

The P-Profile-Key header is defined in RFC 5002, The Session Initiation Protocol (SIP) P-Profile-Key Private Header (P-Header).If the identity of the server user of the request was via a wildcarded PUI, that wildcarded PUI is included in the P-Profile-Key header in the outgoing request to the core. This Header will be included for requests like INVITE, SUBSCRIBE, and so on that match the registered wildcarded PUI. The header will occur in all subsequent requests to the core as well.

For this header to be present, the request needs to be going from access to core and the ims-feature must be enabled on both the access and core SIP interfaces.

Registration Event Package

If subscription to the Registration Event package (defined in RFC 3680, A Session Initiation Protocol (SIP) Event Package for Registrations) is enabled, the P-CSCF receives asynchronous NOTIFYs from the core reporting registration state changes. Each NOTIFY provides a </reg-info> XML element that reports state changes. A sample XML report is shown below.

```
   <registration aor="sip:user1_public1@home1.net" id="as10" state="active">

          <contact id="86" state="active" event="created">
               <uri>sip:[5555::aaa:bbb:ccc:ddd]</uri>
          </contact>

   </registration>



   <registration aor="sip:ep_user1@home1.net" id="as11" state="active">

          <contact id="86" state="active" event="created">
               <uri>sip:[5555::aaa:bbb:ccc:ddd]</uri>
          </contact>


          <ere:wildcardedIdentity>sip:ep_user!.*!@home1.net</
ere:wildcardedIdentity>

   </registration>

</reginfo>
```

If a </wildcardedIdentity> sub-element is included in the <registration> element (as in the second </registration>, the wildcarded PUI is taken from the </wildcardedIdentity> sub-element, and that wildcarded PUI replaces any previously stored wildcarded PUIs in the registration.

In the absence of </wildcardedIdentity> sub-element(as in the first registration), the PUI is taken from the ‘aor’ attribute of the registration element, and the registration cache is left unchanged.

Message Flows

A basic registration exchange with support for PUIs enabled.

From UA to P-CSCF

```
REGISTER sip:192.168.1.232:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.231:5060;branch=z9hG4bK-13794-1-0
From: <sip:7818888@hxu.com;tag=1
To: sut <sip:7818888@hxu.com>
Call-ID: 1-13794@192.168.1.231
CSeq: 1 REGISTER
```

```
Contact: <sip:7818888@192.168.1.231:5060>;expires=3600
Content-Length: 0
```

From P-CSCF to Registrar

If reg-cache-route is enabled on egress sip-interface, P-CSCF provides Path support with a value of P-CSCF-contact, so INVITE from core will have route header with value of this path per RFC 3327.

```
REGISTER sip:192.168.200.231:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.232:5060;branch=z9hG4bK0g6tnu200o10tfgl0641.1
From: <sip:7818888@hxu.com;tag=1
To: sut <sip:7818888@hxu.com>
Call-ID: 1-13794@192.168.1.231
CSeq: 1 REGISTER
Contact: <sip:7818888-
rrbgth3ot667c@192.168.200.232:5060;transport=udp>;expires=3600
Content-Length: 0
Supported: path
Path: <sip:7818888-rrbgth3ot667c@192.168.200.232:5060; lr>
Max-Forwards: 70
```

From Registrar to P-CSCF

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.200.232:5060;branch=z9hG4bK0g6tnu200o10tfgl0641.1
Call-ID: 1-13794@192.168.1.231
CSeq: 1 REGISTER
Contact: <sip:7818888-
rrbgth3ot667c@192.168.200.232:5060;transport=udp>;expires=3600
P-Associated-URI: <sip:781!.*!@hxu.com>  -- wildcard regex
Content-Length: 0
```

From P-CSCF to UA

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.231:5060;branch=z9hG4bK-13794-1-0
From: <sip:7818888@hxu.com;tag=1
To: sut <sip:7818888@hxu.com>;tag=1
Call-ID: 1-13794@192.168.1.231
CSeq: 1 REGISTER
Contact: <sip:7818888@192.168.1.231:5060>;expires=3600
P-Associated-URI: <sip:781!.*!@hxu.com>
Content-Length: 0
```

Incoming Request INVITE from core

For the incoming request from the core to the P-CSCF, if reg-cache-route is enabled on ingress SIP Interface, P-CSCF checks the registered P-CSCF-contact-URI in the top Route P-CSCF-contact-URI, instead of with request-URI.

```
INVITE sip:7816666@192.168.200.232:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.231:5060;branch=z9hG4bK-13800-1-0
From: sipp <sip:3054252165@192.168.200.231:5060;user=phone>;tag=1
```

```
To: sut <sip:7816666@hxu.com>
Call-ID: 1-13800@192.168.200.231
Supported: 100rel,timer,resource-priority,replaces
CSeq: 1 INVITE
Contact: sip:7816666@192.168.200.231:5060
Max-Forwards: 70
Route: <sip:7818888-rrbgth3ot667c@192.168.200.232:5060;lr>
Content-Type: application/sdp
Content-Length:  141

v=0^M
o=user1 53655765 2353687637 IN IP4 192.168.200.231
s=-
c=IN IP4 192.168.200.231
t=0 0
m=audio 6000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

outgoing Request INVITE

For outgoing Request from access side, the Register Cache entry will be found by reg-via-key, when options ‘reg-via-key’ and ‘reg-via-match’ are set, P-CSCF will have wildcarded PAU checking for allow-anonymous (or registered) verification if option wildcard-puid-match is set.

```
INVITE sip:service@192.168.1.232:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.231:5060;branch=z9hG4bK-11911-1-0
From: sipp <sip:7816666@hxu.com>;tag=1
To: sut <sip:service@192.168.1.232:5060>
Call-ID: 1-11911@192.168.1.231
Supported: 100rel,timer,resource-priority,replaces
CSeq: 1 INVITE
Contact: sip:781666@192.168.1.231:5060
Max-Forwards: 70
Content-Type: application/sdp
Content-Length:  137

v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.231
s=-
c=IN IP4 192.168.1.231
t=0 0
m=audio 6000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

HA Configurations

Support for wildcarded PUIs requires checkpointing two new data points that need to be replicated in HA configurations.

- the P-Profile-Key in the sip session so that subsequent requests in a session can take the cached value and include that value in egress requests to core
- Upon SUB-NOT-200OK, the active P-CSCF replicates the subscribe session to the standby. This information is required to replicate the new PAU list of the associated SIP User if the </wildcardedIdentity> element arrives in the NOTIFY XML and has wildcarded entities.

IMS Support for Private Header Extensions for 3GPP

As part of its RFC 3455 support, the Oracle Communications Session Border Controller supports the following headers in its IMS implementation:

- P-Associated-URI
- P-Asserted-Identity
- P-Called-Party-ID
- P-Charging-Function-Address
- P-Visited-Network-ID
- P-Early-Media —The PEM header, which has applications within IMS deployments is explained in the P-Early-Media SIP Header Support section.

The procedure to enable IMS support is explained in the previous section. IMS and all related functions must be enabled on both the access-side and core-side SIP interfaces.

P-Associated-URI Header

In the SIP registration process, the registrar often returns a set of associated URIs for a registering AoR. When the Oracle Communications Session Border Controller receives the list of associated URIs, it stores them in the registration entry for the registering endpoint. The service provider allocates one or more associated URIs per user for his or her own usage. After an endpoint successfully registers, the P-Associated-URI header returned in a 200 OK message informs the UE of all URIs associated with the AoR.

When the Oracle Communications Session Border Controller receives a request from a UE, the URI in the From header is matched against the registration cache for that endpoint. If the registering endpoint matches an associated-URI already in the registration table, the Service-Route associated with this endpoint is used to create the route for originating transactions associated with the endpoint to the S-CSCF.

The inclusion or exclusion of the P-Associated-URI header is not dependent on the trust level of an ingress or egress realm.

P-Asserted-Identity Header

The Oracle Communications Session Border Controller inserts a P-Asserted-Identity header into any initial request for a dialog or standalone transaction sourced by the UE.

The inclusion or exclusion of the P-Asserted-Identity header is dependent on the trust level of an egress realm.

P-Asserted-Identity Header Handling

1. The Oracle Communications Session Border Controller inserts a P-Asserted-Identity header into all messages other than the REGISTER message.
2. When the P-Preferred-Identity header is present in an INVITE sourced by the UE, and the SIP URI contained in this header is also present in the UE's associated URI list, then this SIP URI is inserted in the P-Asserted-Identity header as the SIP message enters the core network.
3. When the P-Asserted-Identity header is present in an INVITE sourced by the UE, and the SIP URI contained in this header is also present in the UE's associated URI list, then the

original P-Asserted-Identity header and SIP URI is passed unchanged into the core network.

4. When the From header is present in an INVITE sourced by the UE, and the SIP URI contained in this header appears in the UE's Associated URI list, then this SIP URI is inserted into the P-Asserted-Identity header as the SIP message enters the core network.
5. When the P-Asserted-Identity header is present in an INVITE sourced by the UE, and the SIP URI contained in this header is not present in the Associated URI list, the Oracle Communications Session Border Controller acts like no P-Asserted-Identity was received from the UE.
6. When no P-Asserted-Identity can be derived from an INVITE sourced by the UE, the P-Asserted-Identity is based on the first URI in the Associated URI list.
7. The P-Asserted-Identity header will be removed from SIP messages sent and received from a UE if either the ingress or egress side is untrusted and the UE's Privacy header's contents is id.
8. If no P-Associated-URI exists for a registered endpoint, the Oracle Communications Session Border Controller will use the configured default P-Asserted-Identity found on the sourcing session agent. This feature works with both SIP and H.323 session agents.
9. If the session agent that originates a message does not include a P-Asserted-Identity header or the request is not originated from the session agent, and the P-CSCF has not received P-Associated-URI list from the registrar for a particular user, no P-Asserted-Identity will be created.
10. The P-Preferred-Identity header will never be passed to the S-CSCF.

    If the above steps fail to insert a P-Asserted-Identity header, you can manually configure a value to be inserted into a P-Asserted-Identity header. The sip-ims-feature parameter must still be enabled to use the P-Asserted-Identity header override.

P-Asserted-Identity Header Configuration

P-Asserted-Identity header handling is enabled with the sip-ims-feature as described in the previous section. A P-Asserted-Identity header can be manually configured for a session agent if the automatic logic, explained earlier in this section, fails.

To configure the P-Asserted-Identity header for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Type **select** and the number of the pre-configured session agent you want to configure.

```
ORACLE(session-agent)# select 1
```

5. Type **p-asserted-id** <space> <URI to use when no P-Asserted-ID has been created> and press Enter. This completes the configuration.

```
ORACLE(session-agent)# p-asserted-id sip:name@acmepacket.com
```

6. Save your work using the ACLI **done** command.

P-Called-Party-ID Header

The Oracle Communications Session Border Controller transparently passes the P-Called-Party-ID header between the S-CSCF and a UA.

IMS Charging Headers

The Oracle Communications Session Border Controller supports IMS Charging Headers. These headers include P-Charging-Vector and the P-Charging-Function-Address. IMS charging header support is configured separately from other IMS functions in order to support a variety of customer needs. Charging header information is now recorded in the CDR records.

A charging vector is defined as a collection of charging information. The charging vector may be filled in during the establishment of a dialog or standalone transaction outside a dialog. The information inside the charging vector may be filled in by multiple network entities (including SIP proxies) and retrieved by multiple network entities. There are three types of correlation information to be transferred: the IMS Charging Identity (ICID) value, the address of the SIP proxy that creates the ICID value, and the Inter Operator Identifiers (IOI).

Charging headers are inserted, deleted, or ignored for request messages. They are forwarded through the Oracle Communications Session Border Controller unmodified when embedded in response messages. If you wish to modify the charging headers in a response message, you must use the Oracle Communications Session Border Controller's header manipulation feature as a general solution.

P-Charging-Vector

You can configure the Oracle Communications Session Border Controller (SBC) to processes the P-Charging Vector header in four different ways.

- If a P-Charging-vector header is present in an incoming SIP request, the SBC can pass the header untouched, as part of the full SIP message that is forwarded out of an egress interface.
- If a P-Charging-vector header is present in an incoming SIP request, the SBC can delete the header and forward the full SIP message out of an egress interface.
- If an incoming SIP request does not contain a P-charging-vector header, the SBC can create and insert the header and forward the full SIP message out of an egress interface. Likewise, if an incoming SIP request contains an existing P-Charging-Vector header, the SBC can overwrite this header with the values generated internally.
- Determine whether to insert a P-Charging-Vector header based on whether the header is present when the SBC receives it.

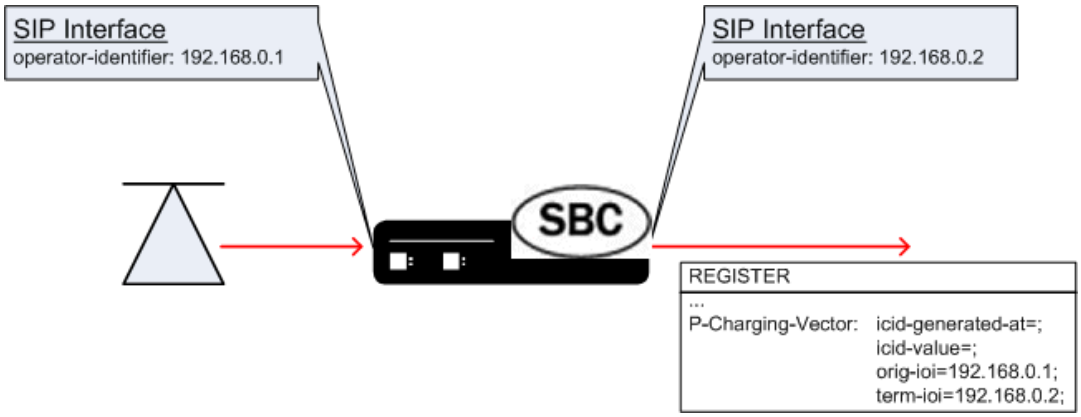
The P-Charging-Vector header is composed of four parameters: icid-value, icid-gen-addr, orig-ioi, term-ioi. See RFC 3455, Section 4.6 for more information.

1. The SBC constructs the icid-value in the following format:

   ```
   icid-value=<unique string>
   ```

   The unique string is a value created by the SBC and is based on the realm, local IP port, time, and a sequence number.
2. The icid-gen-addr parameter's value is the IP address of the egress SIP interface. This value is generated by the SBC.
3. The orig-ioi parameter's value is set manually using the operator-identifier field located in the SIP interface configuration element.
4. The term-ioi parameter's value is set manually using the operator-identifier field located in the SIP interface configuration element.

   You configure charging vector handling on the SBC interface that receives the SIP request by turning on the switches that enable charging vector processing on the ingress interface for the call. Based on the direction of the call, the SBC will insert the operator-identifier configuration parameter into the orig-ioi and the term-ioi parameters. The orig-ioi parameter takes the value of the operator-identifier configuration parameter of the SIP interface that receives the SIP request. The term-ioi parameter takes the value of the operator-identifier configuration parameter of the SIP interface that sends the SIP request to its next hop.



P-Charging-Vector Header Example

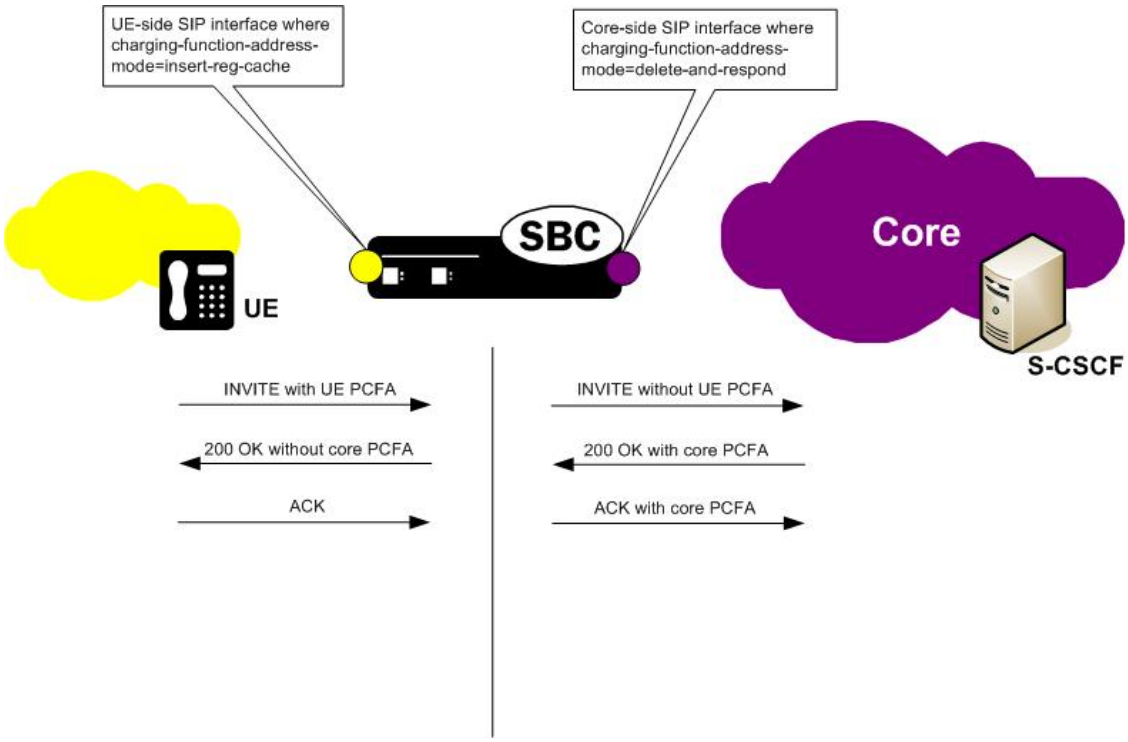
```
P-Charging-Vector: icid-
value=1ate6g46n1823s8719ck3ps6gbt46m5d3bci3po5hhdg3n86g1csio47g9c43@192.168.0.
2;
icid-generated-at=192.168.0.2;
orig-ioi=192.168.0.1;
term-ioi=192.168.0.2;
```

P-Charging-Function-Address

The P-Charging-Function-Address header is composed of two configurable parameters: ccf, ecf. You can configure the Oracle Communications Session Border Controller (SBC) to processes the P-Charging-Function-Address header in five different ways.

- If a P-Charging-Function-Address header is present in an incoming SIP request, the SBC can be set to pass the header, untouched, as the full SIP request is forwarded out of an egress interface.
- If a P-Charging-Function-Address header is present in an incoming SIP request, the SBC can be set to delete the header and forward the SIP request out of an egress interface.
- If an incoming SIP request does not contain a P-Charging-Function-Address header, the SBC can be set to create and insert the header and forward the SIP message out of an egress interface.
- If an incoming SIP request contains a P-Charging-Function-Address header, and the SBC is set to insert a configured P-Charging-Function-Address header, the new parameters will be appended before the existing parameters in the header. The SBC will then forward the SIP request out of an egress interface.
- Determine whether to insert a P-Charging-Function-Address header based on whether the header is present when the SBC receives it.

In addition, the SBC can also be configured to perform insertion and caching for the PCFA header in dialog-creating or stand-alone messages. The following diagram illustrates how this works:



For this scenario, there are two main functions, PCFA insertion and PCFA caching:

- PCFA insertion—Using the insert-reg-cache and delete-and-respond configuration values, the SBC adds the PCFA to all SIP requests and to the response on the P-CSCF facing the SIP interface. However, only dialog-creating and standalone requests, and responses to each of those, update the SBC and accounting information. Such requests do not have a To tag, and responses do not appear in established dialogs. The SBC inserts the PCFA into provisional (1XX) and success (2XX) responses, with the exception of the 100 Trying response.
  You can use SIP header manipulation rules (HMR) to remove any unwanted headers.

- PCFA caching—When you use either of the insert-reg-cache and delete-and-respond configuration values, the SBC uses the latest cached copy of a PCFA header to insert into requests and responses. The SBC does not cache any PCFA headers it receives on SIP interfaces using the none, insert, or insert-reg-cache modes because this type of SIP interface faces the UE making its replacement headers ones from the core.
  Though there can be various sources for the latest cached copy, the PCFA header received as part of a dialog-creating or standalone request has highest precedence. This PCFA header is then stored as the latest cached value for that dialog. That is, for each specific dialog, the SBC the PCFA is cached separately so it can add the most specific PCFA to the message—and is added to any message for the dialog.

  When there is no cache PCFA for a specific dialog, the SBC uses the registration cache entry as the latest cached copy. And when there is no entry in the registration, the PCFA uses the **ccf-address** and **ecf-address** values from the SIP interface.

  The latest cached copy or the **ccf-address** is the value reported in the RADIUS VSA Acme-Session-Charging-Function-Address; this VSA is used for both of the new modes.

> **Note:**
>
> Only the **ccf-address** is reported in RADIUS records; the **ecf-address** is not.

P-Charging-Function-Address Header Example

```
P-Charging-Function-Address: ccf=192.168.0.20 ; ecf=192.168.0.21;
```

RADIUS Accounting of Charging Headers

When the Oracle Communications Session Border Controller creates either the P-Charging-Vector header or the P-Charging-Function-Address header, it inserts an entry in the RADIUS record to record the charging header data.

For a P-Charging-Vector header, the icid-value is saved to the P-Charging-Vector attribute in the radius record. If the Oracle Communications Session Border Controller does not create a P-Charging-Vector header, but it receives a SIP message that already has the P-Charging-Vector header with an icid-value, the existing icid-value is written to the RADIUS record.

For a P-Charging-Function-Address header, the first CCF value is saved to the P-Charging-Function-Address attribute. When the Oracle Communications Session Border Controller creates the P-Charging-Function-Address, the CCF value it inserts into the header is saved to the radius record. If the Oracle Communications Session Border Controller does not create a P-Charging-Function-Address header, but it receives a SIP message that already has the P-Charging-Function-Address with a CCF value, the existing CCF value is written to the RADIUS record.

Name	Value	Value Type
Acme-Session-Charging-Vector	54	string
Acme-Session-Charging-Function-Address	55	string

P-Charging-Vector Processing for SIP Interfaces Configuration

P-Charging-Vector header handling is enabled in the SIP interface.

To configure P-Charging-Vector processing in a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

If you are adding support to an existing SIP interface, then you will need to select the interface you want to edit using the ACLI **select** command.

```
ORACLE(sip-interface)# select
```

4. **charging-vector-mode**—Sets the Oracle Communications Session Border Controller to create the P-Charging-Vector header. The default value is **pass**. The valid values are:
   - **none**—Pass the P-Charging-Vector header received in an incoming SIP message untouched as the message is forwarded out of the Oracle Communications Session Border Controller , but does not include icid-value in accounting records.
   - **pass**—Pass the P-Charging-Vector header received in an incoming SIP message untouched as the message is forwarded out of the Oracle Communications Session Border Controller , includes icid-value in accounting records.
   - **delete**—Delete the P-Charging-Vector header received in an incoming SIP message before it is forwarded out of the Oracle Communications Session Border Controller .
   - **insert**—Insert the P-Charging-Vector-Mode header in an incoming SIP message that does not contain the Charging-Vector header. If the incoming message contains the Charging-Vector header, the Oracle Communications Session Border Controller will overwrite the Charging-Vector-Mode header with its values. This option always uses the **ccf-address** and **ecf-address** static values.
   - **delete-and-respond**—To be configured on the SIP interface facing the P-CSCF, configures the Oracle Communications Session Border Controller to strip out the latest cached PCFA from the core side. The Oracle Communications Session Border Controller then remembers this PCFA and uses it in communications sent to the core.
   - **conditional-insert**—Configures the Oracle Communications Session Border Controller to perform the following functions, based on whether or not the header is present in the message when the Oracle Communications Session Border Controller receives it:
     – For incoming messages that have the associated header, the SBC uses **pass** mode.
     – For those that do not have the associated header, the SBC uses **insert** mode.

     An incoming message received on a realm that does not have a **sip-interface** associated with it refers to the egress **sip-interface** configuration to determine the behavior. Such messages include those with a PCV header. Under these conditions

and with the egress **sip-interface** configured for **conditional-insert**, the system forwards the PCV header unmodified. The system would also save accounting/charging information normally.

5. **operator-identifier—**Set the operator identifier value to be inserted into a P-Charging-Vector header. The direction of the call determines whether this value is inserted into the orig-ioi or the term-ioi parameter in the P-Charging-Vector header. This string MUST begin with an alphabetical character.
6. **charging-function-address-mode—**Set the charging function address mode you want to use. The default value is **pass**. The valid values are:
   - **none**—Pass the P-Charging-Function-Address header received in an incoming SIP message untouched as the message is forwarded out of the Oracle Communications Session Border Controller , but does not include icid-value in accounting records.
   - **pass**—Pass the P-Charging-Function-Address header received in an incoming SIP message untouched as the message is forwarded out of the Oracle Communications Session Border Controller , includes icid-value in accounting records.
   - **delete**—Delete the P-Charging-Function-Address header received in an incoming SIP message before it is forwarded out of the Oracle Communications Session Border Controller .
   - **insert**—Insert the P-Charging-Function-Address header in an incoming SIP message that does not contain the P-Charging-Function-Address header. If the incoming message contains the P-Charging-Function-Address header, the Oracle Communications Session Border Controller will overwrite the P-Charging-Function-Address header with its values. This option always uses the **ccf-address** and **ecf-address** static values.
   - **insert-reg-cache**—To be configured on the SIP interface facing the UE, configures the Oracle Communications Session Border Controller to replace the PCFA with the most recently cached value rather than the **ccf-address** and **ecf-address** you set to be static in your configuration. The cached values come from one of the following that the Oracle Communications Session Border Controller has received most recently: request, response, registration, or local configuration.
   - **delete-and-respond**—To be configured on the SIP interface facing the P-CSCF, configures the Oracle Communications Session Border Controller to strip out the latest cached PCFA from the core side. The Oracle Communications Session Border Controller then remembers this PCFA and uses it in communications sent to the core.
   - **conditional-insert**—Configures the Oracle Communications Session Border Controller to perform the following functions, based on whether or not the header is present in the message when the Oracle Communications Session Border Controller receives it:
     – For incoming messages that have the associated header, the SBC uses **pass** mode.
     – For those that do not have the associated header, the SBC uses **insert** mode.

> **Note:**
>
> The default settings for this parameter and for **charging-vector-mode** are **pass** for new SIP interface configurations. If you are upgrading and there are pre-existing SIP interfaces in your configuration, the defaults become **none**.

7. **ccf-address—Set the CCF address value that will be inserted into the P-Charging-Function-Address header.**
8. **ecf-address—Set the ECF address value that will be inserted into the P-Charging-Function-Address header.**
9. Save your work using the ACLI **done** command.

Charging Correlation

IMS is based on a distributed architecture that can result in multiple network entities becoming involved in providing access and services. Operators require the ability to charge for the access and services as they provide. This necessitates coordination among the network entities (for example, SIP proxies), which includes correlating charging records generated from different entities that are supporting the same session.

Charging correlation is supported in Release SC-X7.1.2 and later.

During initial provisioning of session information, the Oracle Communications Session Border Controller, functioning as a P-CSCF, indicates its support for charging correlation by subscribing to access network charging information from the PCRF (Policy Charging Rule Function). In a typical exchange, the subscription is requested with an AA-Request (AA-R) command that contains the Specific-Action (Type 513) and AF-ChargingIdentifier (Type 505) AVPs. The Specific-Action AVP is set to 1, (CHARGING_CORRELATION_EXCHANGE), requesting that the PCRF provide new and updated charging information as such information becomes available.

The PCRF responds with an AA-Answer (AA-A) that usually contains one or more associated Access-Network-Charging-Identifier (Type 502) and an Access-Network-Charging-Address (Type 501) AVPs. The Access-Network-Charging-Address AVP contains the IP address of a network entity that is charging for session services. The Access-Network-Charging-Identifier associated with the entity identified by the Access-Network-Charging-Address AVP. This identifier is applied to all traffic/media flows within the current session.

The AA-A can contain an IP-CAN-Type (Type 1027) AVP that indicates the type of Connectivity Access Network and an associated RAT-Type AVP that provides more specific information as to the access technology. For IMS charging correlation, the IP-CAN-Type AVP can contain one or two values: 0 (3GPP), or 5 (3GPP-EPS).

In some cases, the PCRF may not have the requested charging information at the time of the initial AA-R. When information becomes available, the PCRF sends the data to the P-CSCF via a Re-Auth-Request (RAR) command, which is then acknowledged by the P-CSCF with a Re-Auth-Answer (RAA) command.

Upon receiving charging information from the PCRF, the P-CSCF saves the data in the context of the associated SIP session. Saved data is used to populate the P-Charging-Vector SIP header.

Charging Correlation Configuration

To enable support for charging correlation:

1. Navigate to ext-policy-server configuration mode.

```
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# ext-policy-server
ACMEPACKET(ext-policy-server)#
```

2. **specific-action-subscription**—Enables the system to subscribe to IMS action-specific notification services. By default, no subscriptions are enabled. Available subscriptions are:
   - charging-correlation-exchange
   - loss-of-bearer
   - recovery-of-bearer
   - release-of-bearer
   - out-of-credit
   - successful-resource-allocation
   - failed-resource-allocation

   ```
   ACMEPACKET(ext-policy-server)# specific-action-subscription changing-
   correlation-exchange
   ```

   adds a single subscription

   ```
   ACMEPACKET(ext-policy-server)# specific-action-subscription (changing-
   correlation-exchange loss-of-bearer recovery-of-bearer)
   ```

   adds multiple subscriptions
3. Use done, exit, and verify-config to complete configuration.

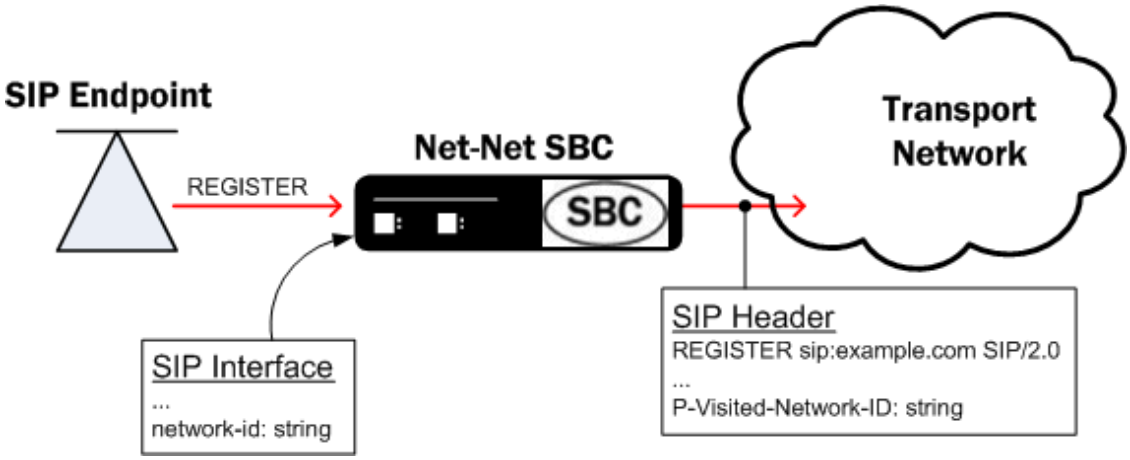
P-Early-Media SIP Header Support

The P-Early-Media (OEM) SIP Header provides a means of managing and authorizing the use of early media. This header applies to multiple contexts, including within IMS. Documentation on the PEM header is consolidated in the SIP chapter.

See Early Media Support for PEM documentation.

P-Visited-Network-ID Header

The Oracle Communications Session Border Controller's IMS support also includes the insertion of a P-Visited-Network-ID header into SIP messages when applicable. When a UE sends a dialog-initiating request (e.g., REGISTER or INVITE message) or a standalone request outside of a dialog (e.g., OPTIONS) to the P-CSCF, the Oracle Communications Session Border Controller inserts the P-Visited-Network-ID header into the SIP message as it enters into the destination network.



The P-Visited-Network ID header will be stripped from SIP messages forwarded into untrusted networks as expected. The content of a P-Visited-Network-ID header is a text string that identifies the originating UE's home network. This string is user-configurable.

P-Visited-Network-ID Header Handling for SIP Interfaces Configuration

P-Visited-Network-ID header handling is enabled with the sip-ims-feature as described earlier. The actual P-Visited-Network-ID string must be configured on the access-side SIP interface. The Oracle Communications Session Border Controller must consider the egress device trusted or it does not add that the P-Visited-Network-ID header to the forwarded request.

To configure the P-Visited-Network-ID string in a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure.

   ```
   ORACLE(sip-interface)# select 1
   ```

5. Type **network-id** <space> <network ID string> and press Enter. This completes the configuration of the P-Visited-Network-ID string for a given SIP interface.

   ```
   ORACLE(sip-interface)# network-id examplenetworkid
   ```

6. Save your work using the ACLI **done** command.

Second P-Asserted-Identity Header for Emergency Calls

The Oracle Communications Session Border Controller can add a second P-Asserted-Identity header when forwarding an emergency message into the network.

When the UE registers with an S-CSCF, the S-CSCF returns a set of associated URIs, the implicit registration set (IRS,) for the AoR in the 200 OK response. The Oracle Communications Session Border Controller caches the IRS. The user identities that comprise the cached IRS are used for validation later.

As the Oracle Communications Session Border Controller receive a UE's INVITE, the value in the P-Preferred-Identity header is validated against the public user identities in the cached IRS. If the inbound P-Preferred-Identity matches any items in the IRS, the Oracle Communications Session Border Controller inserts that value into a P-Asserted-Identity header in the egress message. The P-Preferred- Identity header is stripped from the outbound message.

Inbound INVITE Contains:	Validate Against Implicit Registration Set	Outbound Invite Created With: (all P-Preferred-Identity headers are removed)
P-Preferred-Identity: value-1	value-1 present	P-Asserted-Identity: value-1

The Oracle Communications Session Border Controller can create a second P-Asserted-Identity header by configuring the add-second-pai-for-emergency-calls option in the SIP config. Once a combination of SIP URIs and/or TEL URIs in the inbound P-Preferred-Identity header are validated against the IRS, the Oracle Communications Session Border Controller forwards the emergency INVITEs with two P-Asserted-Identity headers to the E-CSCF. The behavior is based upon the URI type received in the P-Preferred-Identity header and whether those values validate against the IRS list.

Inbound INVITE Contains:	Validate Against Implicit Registration Set	Outbound Invite Created With: (all P-Preferred-Identity headers are removed)
P-Preferred-Identity: SIP-URI-1	SIP-URI-1 present TEL-URI-1 present (TEL-URI-n present)	P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: TEL-URI-1
P-Preferred-Identity: TEL-URI-1	TEL-URI-1 present SIP-URI-1 present (SIP-URI-n present)	P-Asserted-Identity: TEL-URI-1 P-Asserted-Identity: SIP-URI-1
NO P-Preferred-Identity:	(SIP | TEL) URI -1 present (SIP | TEL) URI -n present	P-Asserted-Identity: 1st public identity from IRS
P-Preferred-Identity: SIP-URI-1	TEL-URI-1 present	P-Asserted-ID: SIP-URI-1
P-Preferred-Identity: TEL-URI-1	SIP-URI-1 present	P-Asserted-ID: TEL-URI-1
P-Preferred-Identity: SIP-URI-1	SIP-URI-1 present	P-Asserted-Identity: SIP-URI-1
P-Preferred-Identity: SIP-URI-2	SIP-URI-2 present TEL-URI-n present	P-Asserted-Identity: 1st TEL-URI from IRS
P-Preferred-Identity: TEL-URI-1	TEL-URI-1 present	P-Asserted-Identity: TEL-URI-1
P-Preferred-Identity: TEL-URI-2	TEL-URI-2 present SIP-URI-n present	P-Asserted-Identity: 1st SIP-URI from IRS

If the INVITE does not include a P-Preferred-Identity header and does not include a P-Asserted-Identity header, or the value in the original P-Preferred-Identity or P-Asserted-

Identity header is not contained in the IRS, or the URI from the From: header is not the in the IRS, then default public user identity is inserted into a P-Asserted- Identity header in the egress message. (The default public user identity is the first on the list of URIs in the P-Associated-URI header.)

If the strict-3gpp-pai-compliance option is configured in the outbound SIP interface, the first P-Asserted-Identity header also includes the display name.

Two Incoming P-Asserted-Identity Headers

When the inbound INVITE contains 2 P-Asserted-Identity headers, the Oracle Communications Session Border Controller ensures that both outbound P-Asserted-Identity headers contain public user identities from the IRS according to the following:

Inbound Invite Contains	Validate Against Implicit Registration Set:	Outbound Invite Created With:
P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: TEL-URI-1	SIP-URI-1 and TEL-URI-1 present	P-Asserted-Identity: SIP-URI-1 or TEL-URI-1 P-Asserted-Identity: 1st public identity from IRS other than value in 1st P-Asserted-Identity
P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: SIP-URI-2	SIP-URI-1 or SIP-URI-2 present TEL-URI-1 present	P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: TEL-URI-1
P-Asserted-Identity: TEL-URI-1 P-Asserted-Identity: TEL-URI-2	TEL-URI-1 or TEL-URI-2 present SIP-URI-1 present	P-Asserted-Identity: TEL-URI-1 or TEL-URI-2 P-Asserted-Identity: SIP-URI-1

1. Navigate to the sip-config configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sip-config
ACMEPACKET(sip-config)#
```

2. Type select to begin configuring this object.

```
ACMEPACKET(sip-config)# select
ACMEPACKET(sip-config)#
```

3. options—Configure the add-second-pai-for-emergency-calls option:

```
ACMEPACKET(sip-config)# options +add-second-pai-for-emergency-calls
ACMEPACKET(sip-config)#
```

4. Save and activate your configuration.

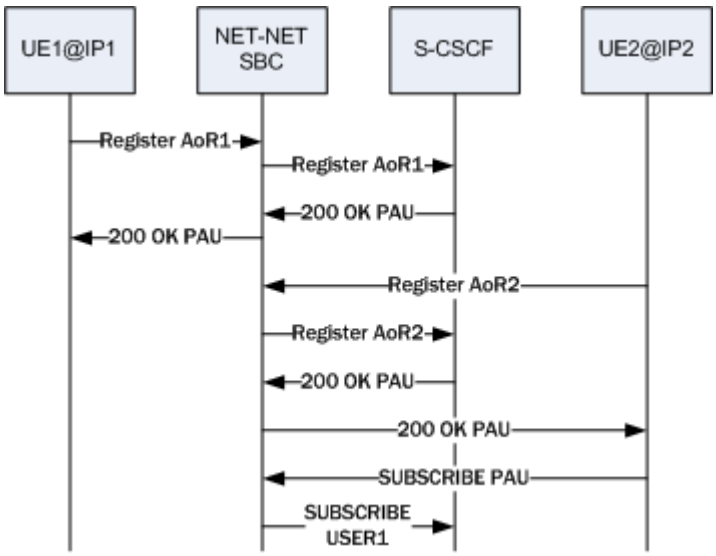
Temporary Public User Identities and Multi-SIM Scenarios

The Oracle Communications Session Border Controller's SIP interface supports multiple registered users for the same P-Asserted-Uri (PAU), a useful ability for multi-SIM scenarios. The call flow for this type of scenario differs depending on whether or not you configure the SIP interface facing the UE with the **reg-via-key** and **reg-via-match** options.

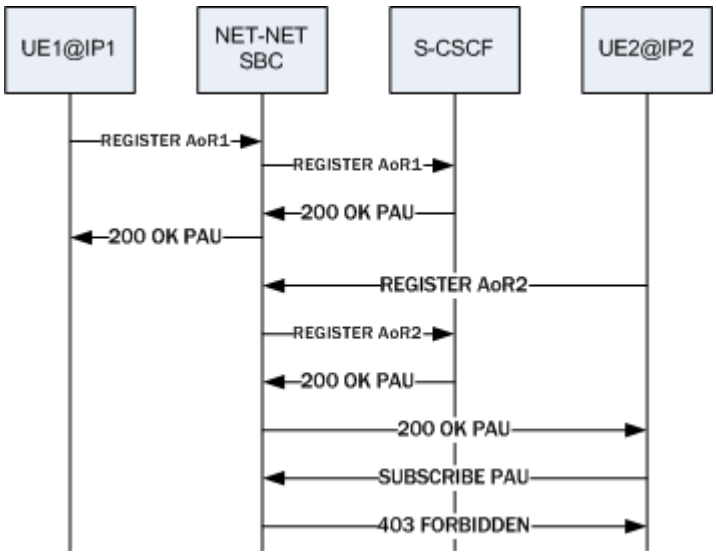
In a multi-SIM scenario, the UE derives a temporary IMS public identity (IMPU); that UE then registers with the Oracle Communications Session Border Controller using the IMPU as the address of record (AoR) from a unique IP. The S-CSCF returns a PAU in the 200 OK, which the Oracle Communications Session Border Controller caches. The UE then derives another IMPU; it registers with the Oracle Communications Session Border Controller using that IMPU as the AoR from another unique IP. The S-CSCF again returns the same PAU in the 200OK.

Old Behavior

Before the introduction of this change, the Oracle Communications Session Border Controller (SBC) associated the PAU only with the first IMPU request. The SBC considered any request made from that PAU to be a request from the first user, regardless of the request's originating IP.

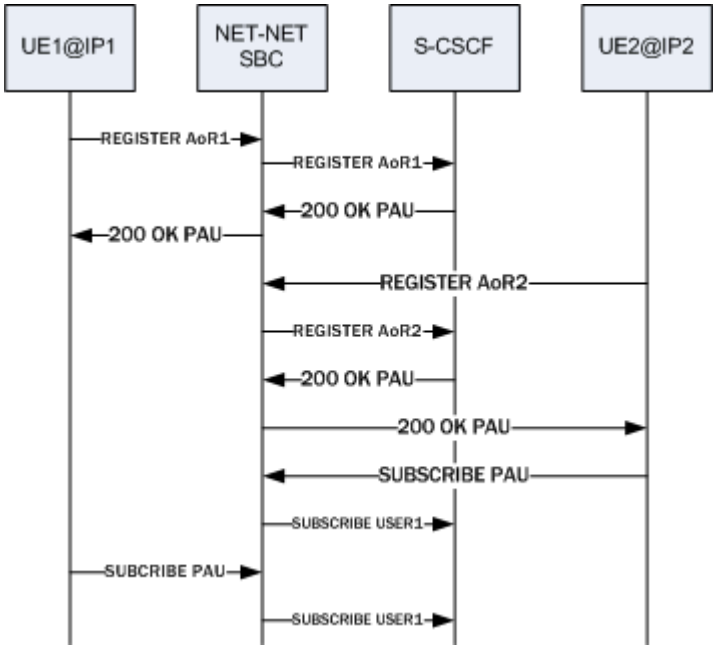


When the SIP interface facing the UE had the **reg-via-key** and **reg-via-match** options configured, the SBC rejected the request from the second user with the PAU as the From or the PPI. Because it only associates the PAU with the first user, the SBC issued a 403 message.

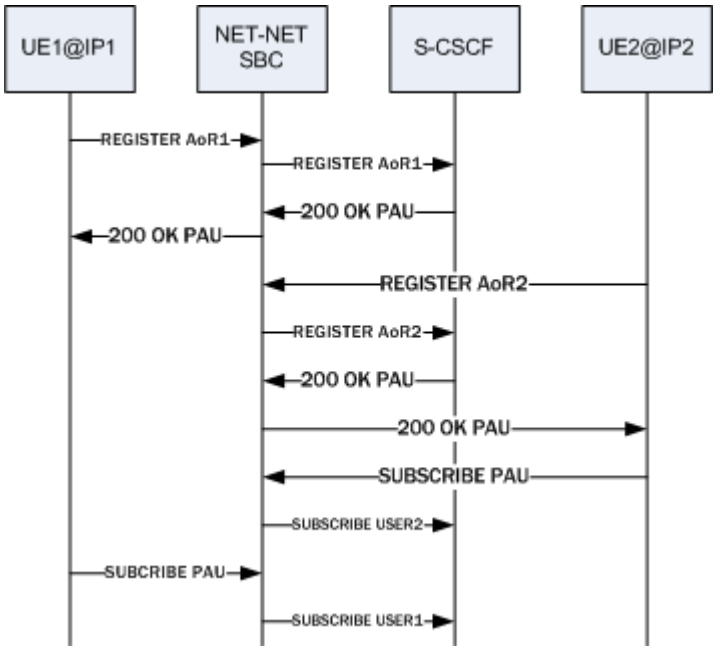


New Behavior

Your Oracle Communications Session Border Controller (SBC) now associates the PAU with both the first and second IMPU. The SBC considers any request made from that PAU to be a request from the user at the top of its registration cache table irrespective of where the request originated (the IP).



When the SIP interface facing the UE has the **reg-via-key** and **reg-via-match** options configured, and the request from the user with the PAU as the From or with PPI, the SBC matches to the proper user based on the source IP.



Configuring SIP Interface with reg-via-key and reg-via-match

If you do not want the use the call scenario associated with the SIP interface options in the New Behavior section, you do not need to make any change to your configuration.

If you want your call scenarios to resemble the one associated with the SIP interface options in the New Behavior section, then you need to configure **reg-via-key** and **reg-via-match** options on the SIP interface facing the UE.

To configure a SIP interface with the reg-via-key and reg-via-match options:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-interface)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name. Then press Enter.

   ```
   ORACLE(sip-interface)# options +reg-via-key
   ORACLE(sip-interface)# options +reg-via-match
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save your work.

Surrogate Registration

The Oracle Communications Session Border Controller surrogate registration feature lets the Oracle Communications Session Border Controller explicitly register on behalf of a Internet Protocol Private Branch Exchange (IP-PBX). After you configure a surrogate agent, the Oracle Communications Session Border Controller periodically generates a REGISTER request and authenticates itself using a locally configured username and password, with the Oracle Communications Session Border Controller as the contact address. Surrogate registration also manages the routing of class from the IP-PBX to the core and from the core to the IP-PBX.

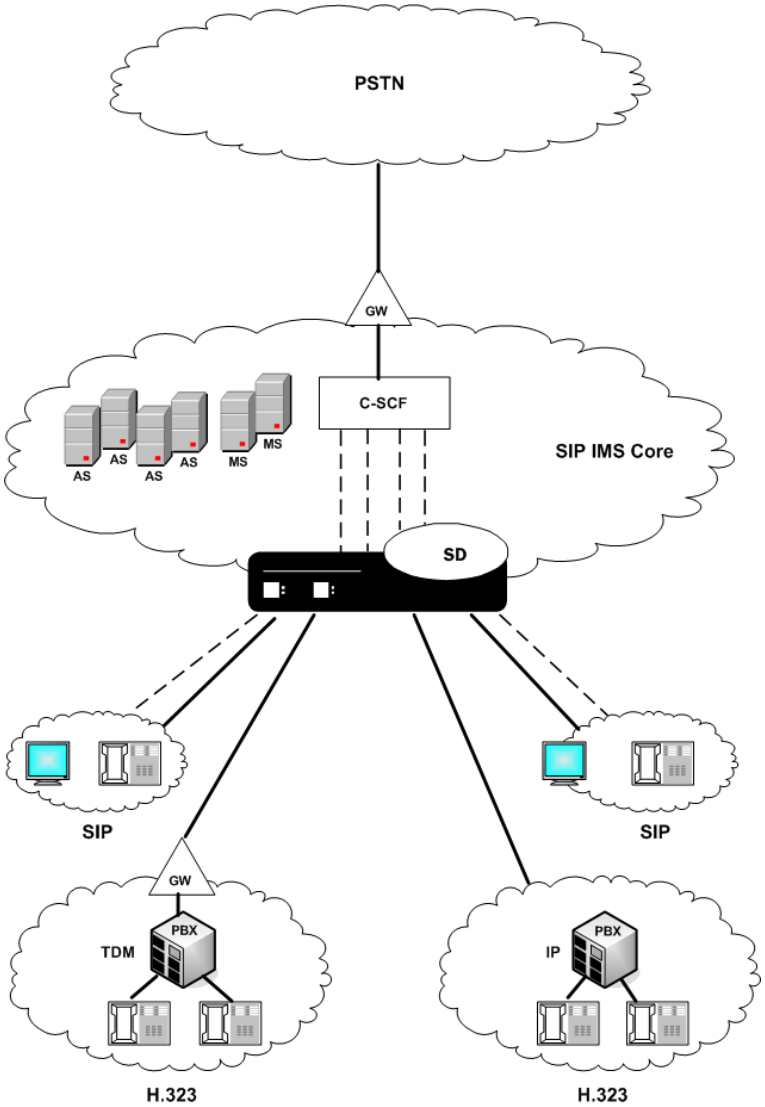
Integrating with IMS

With surrogate registration, the Oracle Communications Session Border Controller (SBC) lets IP-PBXes integrate with the IP Multimedia Subsystem (IMS) architecture. The IP-PBX registers

itself as if it were user equipment (UE), which triggers the implicit registration of all phone numbers associated with the IP-PBX.

Implicit registration means the explicit registration of one address of record (AoR) triggers the implicit registration of all the other AoRs associated with that UE. The implicitly registered AoRs are passed back to the UE as P-Associated-URIs in the registration's 200 (OK).

IMS assumes that each SIP endpoint can register itself with its Serving-CSCF (S-CSCF). However, phones can be connected to SIP Integrated Access Devices (IADs) or SIP or H.323 IP-PBXes. The SBC performs SIP registration on behalf of the IP-PBX and IADs.



The SBC registers on behalf of the IP-PBXes and then stores the associated URIs returned by the Serving Call Session Control Function (S-CSCF). The calls from the phones behind the IP-PBX can be routed based on the cache entry the SBC creates after it receives each phone's associated URI. Calls are routed using the service route, local policy or any other routing mechanism based on the associated session agent or session agent group. The SBC also supports multiple registrations on behalf of a IP-PBX because the IP-PBX can support thousands of phones, but the registrar might only be able to send 10 to 20 associated URIs in response to a single registration.

The SBC replaces the Contact URI for requests from the IP-PBX to the core to match the registered value. For calls from the IMS core to the IP-PBX, the SBC replaces the Request-

URI username with P-Called-Party-ID/To-URI username. The IMS cores sends INVITES for the phones behind the IP-PBX with the registered Contact URI as the Request-URI instead of the AoR of the phones. The IP-PBX needs to see the phone's AoR in the Request-URI.

Registration

The Oracle Communications Session Border Controller uses the configuration information of the surrogate agent that corresponds to a specific IP-PBX. After the surrogate agents are loaded, the Oracle Communications Session Border Controller starts sending the REGISTER requests on their behalf. (You can configure how many requests are sent.)

The SIP surrogate agent supports the ability to cache authorization or proxy-authorization header values from a REGISTER 401, 407, and 200 OK messages and reuse it in subsequent requests, such as in an INVITE. This allows the Oracle Communications Session Delivery Manager to support authorization of subsequent requests in cases such as, when a customer PBX doesn't support registration and authentication. It also supports the generation of authorization/proxy-authorization header if subsequent requests get challenged with a 401/407 response.

If the Oracle Communications Session Border Controller receives 401 or 407 responses to REGISTER, requests, it will use the Message Digest algorithm 5 (MD5) digest authentication to generate the authentication information. You need to specify the password. The Oracle Communications Session Border Controller also supports authentication challenge responses with the quality of protection code set to auth (qop=auth), by supporting the client nonce (cnonce) and nonce count parameters.

The Oracle Communications Session Border Controller creates a registration cache entry for each of the AoRs for which it is sending the REGISTER requests. When the Oracle Communications Session Border Controller receives the associated URIs, it checks whether the customer host parameter is configured. If it is configured, the Oracle Communications Session Border Controller changes the host in the received Associated-URI to the customer host. If it is not configured, the Oracle Communications Session Border Controller does not change the Associated-URI. It makes the registration cache entries that correspond to each of the Associated-URIs. The From header in the INVITE for calls coming from the IP-PBX should have one of the Associated-URIs (URI for a specific phone). If the Oracle Communications Session Border Controller receives a Service-Route in the 200 (OK) response, it stores that as well.

The Oracle Communications Session Border Controller uses the expire value configured for the REGISTER requests. When it receives a different expire value in the 200 OK response to the registration, it stores the value and continues sending the REGISTER requests once half the expiry time has elapsed.

REGISTER requests are routed to the registrar based on the configuration. The Oracle Communications Session Border Controller can use the local policy, registrar host and the SIP configuration's registrar port for routing.

If the Oracle Communications Session Border Controller is generating more than one register on behalf of the IP-PBX, the user part of the AoR is incremented by 1 and the register contact-user parameter will also be incremented by 1. For example, if you configure the register-user parameter as caller, the Oracle Communications Session Border Controller uses caller, caller1, caller2 and so on as the AoR user.

Routing Calls from the IMS Core

The calls coming from the core will have the Oracle Communications Session Border Controller's Contact-URI (which is sent in the REGISTER request) as the Request-URI. The Oracle Communications Session Border Controller looks for a registration entry that corresponds to this URI. After finding the registration entry and the corresponding surrogate agent, the Oracle Communications Session Border Controller looks for the routing mechanism it should use to route this INVITE to the IP-PBX. It uses the customer-next-hop configuration parameter to determine if it routes this call to the session agent, the session agent group, or directly to a particular IP address.

SIP

If the customer-next-hop parameter points to a SIP session agent or the SIP session agent group, the Oracle Communications Session Border Controller creates a Route header using the session agent and modifies the Request-URI. It changes the user portion of the Request-URI to either the user portion of the P-Called-Party-ID header, if present, or to the user portion of the To header. The Oracle Communications Session Border Controller also changes the host portion of the Request-URI to the hostname configured in the customer-host configuration parameter. It makes the change because the domain name on the core side can be different than the domain name on the access IP-PBX side. The Oracle Communications Session Border Controller then uses the added Route header to properly route the call.

H.323

If the session agent or the session agent group configured for the customer-next-hop parameter references an H.323 device, the Oracle Communications Session Border Controller sends the INVITE to its interworking task. If a session agent group is being used, the parameter containing the session agent group name is added to the Request-URI. The host portion of the Request-URI will point to the interworking IP address and the port is changed to 1720.

If a session agent is used, the Oracle Communications Session Border Controller uses it to route the call properly to the interworking task to take care of the H.323 call setup.

> **Note:**
>
> Even though the customer-next-hop field allows specification of a SAG or FQDN, the functionality will only support these values if they resolve to a single IP address. Multiple IP addresses, via SAG, NAPTR, SRV, or DNS record lookup, are not allowed.

Routing Calls from an IP-PBX

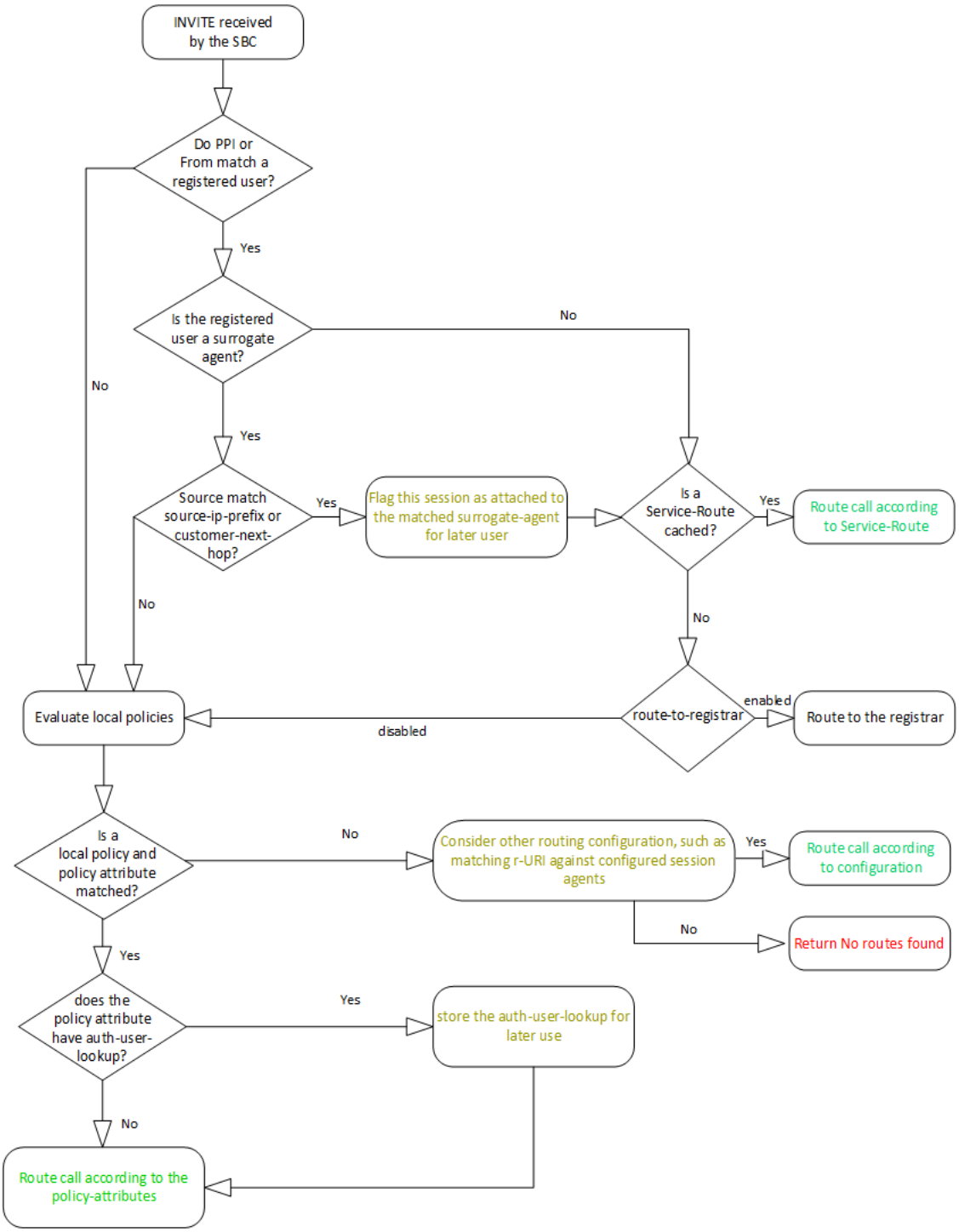
When it receives a call from an IP-PBX configured as a surrogate agent, the SBC attempts to route, and if challenged, perform authentication on behalf of the IP-PBX to authorize the call. It does this by validating the request with your configuration. After routing the call to its destination, the callee may challenge the call, in which case the SBC has an opportunity to authenticate the call. If the SBC cannot authenticate the call, it leaves authentication procedures to other devices, such as the IP-PBX itself.

The process of routing a call from an IP-PBX using a Surrogate Agent fits within the overall routing process, as shown in the following diagram. While determining a route, the SBC also determines whether the call may be authenticated using the SBC as a surrogate. The call may be routed by multiple processes, each of which can include a **surrogate-agent** match and specifying that the SBC can trust the caller.

To route the call, the SBC looks for a service route. After finding the corresponding registration Service-Route entry, the SBC uses the Service-Route for this endpoint to route the call, if it exists.

If no Service-Route exists, but the **route-to-registrar** parameter is enabled on the **sip-interface**, the SBC tries to route the call to the registrar. If **route-to-registrar** is disabled, the SBC refers to **local-policy** for routing.

At this point, the SBC routes the call, and is prepared if it needs to respond to an authentication challenge.



Having received a challenge, the SBC matches the challenge with source and SIP information presented by the caller and stored by the SBC. This matching process is explained below. If the match is successful, the SBC authenticates the call on behalf of the IP-PBX. After authentication succeeds, media between the caller and callee may proceed.

> **Note:**
>
> You can configure the **surrogate-agent** to override the **sip-interface**, **route-to-register** setting. If the surrogate agent's **route-to-registrar** parameter is set to disable, it takes precedence over the **sip-interface** setting. In this case, the SBC does not try to route the call to the registrar.

Configure Surrogate Agents

A surrogate agent is a SBC configuration object that corresponds to an IP-PBX. You configure surrogate agents so that the SBC can perform surrogate registration and routing calls to and from the IP-PBX.

Set the system to Super User mode.

Configure a surrogate agent for each IP-PBX proxy that you want the SBC to register.

> **Note:**
>
> To view all surrogate agent configuration parameters, enter a **?** at the surrogate-agent prompt.

1. Access the **surrogate-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# surrogate-agent
ORACLE(surrogate-agent)#
```

2. On the Add Surrogate Agent page, do the following:
   - Register host—Enter the registrar's hostname to be used in the Request-URI of the REGISTER request. This name is also used as the host portion of the AoR To and From headers.
   - Register user—Enter the user portion of the AoR (Address of Record).
   - Description—Optional. Enter a description of this surrogate agent.
   - Realm ID—Enter the name of realm where the surrogate agent resides (where the IP-PBX proxy resides). There is no default.
   - State—Set the state of the surrogate agent to indicate whether the surrogate agent is used by the application. The default value is **enabled**.
   - Customer host—Optional. Enter the domain or IP address of the IP-PBX, which is used to determine whether it is different than the one used by the registrar.
   - Customer next hop—Enter the next hop to this surrogate agent:
     - session agent group: <session agent group name>
     - session agent: <hostname> or <IPV4>
   - Register contact host—Enter the hostname to be used in the Contact-URI sent in the REGISTER request. This should always point to the SBC. If specifying a IP address,

use the egress interface's address. If there is a SIP NAT on the registrar's side, use the home address in the SIP NAT.

- Register contact user—Enter the user part of the Contact-URI that the SBC generates.
- Password—If you are configuring the auth-user parameter, you need to enter the password used when the registrar sends the 401 or 407 response to the REGISTER request.
- Register expires—Enter the expires in seconds for the REGISTER requests. The default value is **600,000** (1 week). The valid range is 0-999999999.
- Replace contact—This specifies whether the SBC needs to replace the Contact in the requests coming from the surrogate agent. If this is enabled, Contact will be replaced with the Contact-URI the SBC sent in the REGISTER request. The default value is **disabled**. The valid values are enabled and disabled.
- Options—Optional. Enter non-standard options or features.
- Route to registrar—This indicates whether requests coming from the surrogate agent should be routed to the registrar if they are not explicitly addressed to the SBC. The default value is **enabled**. The valid values are enabled and disabled.
- AoR count—Enter the number of registrations to do on behalf of this IP-PBX. If you enter a value greater than **1**, the SBC increments the register-user and the register-contact-user values by that number. For example, if this count is 3 and register-user is john then users for three different register messages will be john, john1, john2. It does the same for the register-contact-user values. The default value is **1**. The valid range is 0-999999999.
- Auth user—Enter the authentication user name you want to use for the surrogate agent. This name is used when the SBC receives a 401or 407 response to the REGISTER request and has to send the REGISTER request again with the Authorization or Proxy-Authorization header. The name you enter here is used in the Digest username parameter. If you do not enter a name, the SBC uses the value of the register-user parameter.
- Max register attempts—Enter the total number of times to attempt registration until success. Range 1-10
- Register retry time—Enter the time to wait after an unsuccessful registration before re-attempting. Range 30-3600
- Count start—Enter the starting value for numbering when performing multiple registrations. Range 0-9999999999
- Register mode—Select automatic (default) or triggered (upon trigger from PBX).
- Triggered inactivity interval—Enter the maximum time with no traffic from the corresponding PBX. (Valid only with Triggered inactivity interval.) Range 5 -300
- Triggered OoS response—503 (Default. Send 503 response for core network failure) or drop response (Do not respond to PBX or core network failure
- Source IP Prefix—Contains a list of IP address/prefixes that specify the source addressing of endpoints the system can authenticate using this surrogate-agent. Valid entries include any number of IP addresses and IP address prefixes in the format <ip>/<subnet>. If you set multiple values, separate them with a space and enclose them with parenthesis (). Addressing can be IPv4, IPv6 or a combination of both. The default configuration is null (no entry).
-
- Auth User Lookup—If you intend to authenticate register requests using a realm configuration, enter the name of the target Auth Attribute configuration in that realm.

This name must match an Auth User Lookup name in the realm's Auth Attribute list. When configured, the SBC uses those credentials to authenticate challenged register requests.

- Proxy Name—If you have configured the Registrar that validates this surrogate agent's register requests as a session agent, enter the name of that session agent here.
- Un-register—Enable this parameter to cause the register requests from this surrogate agent to specify Expires:0 and to remove each of this surrogate agents entries from the registration cache.

3. Save and activate your configuration.

Example

The following example shows the surrogate agent configuration.

```
surrogate-agent
register-host    acmepacket.com
register-user    234567
state    enabled
realm-id    public
description
customer-host    acmepacket.com
customer-next-hop    111.222.33.44
register-contact-host    111.222.5.68
register-contact-user    eng
password
register-expires    600000
replace-contact    disabled
route-to-registrar    enabled
aor-count    1
source-ip-prefix
options
auth-user
max-register-attempts   10
register-retry-time   30
count-start   1
register-mode   automatic
triggered-inactivity-interval   30
triggered-oos-response   503
auth-user-lookup
proxy-name charlie
un-register disabled
last-modified-date    2006-05-04 16:01:35
```

SIP Surrogate Registration Enhancements

For IMS-E networks, enhancements to the Oracle Communications Session Border Controller's SIP surrogate registration capabilities enable it to register a series of endpoints on behalf of a set of devices that are unable to register themselves. In addition, the Oracle Communications Session Border Controller retries failed registrations, prevents authentication loops, and sends an SNMP trap for failed retransmissions. The automatic incrementing of register-user and register-contact-user values are also now more flexible.

Without Enhancements

Without the enhancements configured, the Oracle Communications Session Border Controller's surrogate agent performs a series of registrations based on count when the system boots or when its configuration changes. It only attempts to register each user once. Although the surrogate agent uses the same retry mechanism used for SIP client transactions, it does not attempt further if it receives a failure response until the entry expires. When it receives 401, 403, or 407 responses to requests that include authentication, the surrogate agent's automatic incrementing mechanism appends a number to the end of each registered username. Always starting at one, this number cannot appear in any other position in the username.

With Enhancements

With the enhancements configured, the Oracle Communications Session Border Controller supports:

- Registration retry—You can configure the surrogate agent to retry registration when after a failure, timeout, or transport error. You can set how many times the Oracle Communications Session Border Controller will attempt to register each user; a setting of zero means retries are umlimited. You can also define the number of seconds to wait before initiating a retry. The Oracle Communications Session Border Controller tracks each registration retry count and timers, and sends an SNMP trap when it reaches the maximum number of retries, which signifies failed registration.
- Authentication loop prevention—Authentication loops can occur in previous releases when the Oracle Communications Session Border Controller resends a registration request with authentication in response to 401, 403, or 407 responses (indicating, for example, that there might be a password error). Using the new enhancements, the Oracle Communications Session Border Controller only allows permits the retransmission of one request. It now considers further 401, 403, or 407 responses to be errors and initiates the retry mechanism.
- Automatic increment enhancements—Now, the automatic increment works with the caret (^) in the register-user and register-contact-user fields. These carets define where the automatically generated incrementing number is inserted in the username. You can also use multiple carets to define leading zeroes to insert; for example, the entry user^^^^ will become user0001. You can also define the starting integer for the incrementing registrations. For example, setting the AoR count to 20, the count start to 5, and using the value user^^^^ for register-user and register-contact-user results in the incremented user registrations user0005 through user0025.

Configuring the Retry Mechanism

To set the retry mechanism:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **surrogate-agent** and press Enter.

```
ORACLE(session-router)# surrogate-agent
ORACLE(surrogate-agent)#
```

4. **max-register-attempts**—Using a value from 0 (meaning registration attempts are unlimited) to **10**, enter the number of times you want to attempt registration. The default value is **3**. The valid range is:
   - Minimum—0
   - Maximum—10
5. **register-retry-time**—Enter the amount of time in seconds, between 30 and 3600 seconds, you want the Oracle Communications Session Border Controller to wait before reattempting registration. The default value is **300**. The valid range is:
   - Minimum—10
   - Maximum—3600
6. Save and activate your configuration.

Configuring the Count Start

To set the value where automatic incrementing will start:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **surrogate-agent** and press Enter.

```
ORACLE(session-router)# surrogate-agent
ORACLE(surrogate-agent)#
```

4. **count-start**—Change this parameter from its default of 1 if you want the automatic increment count to start at any other number when the Oracle Communications Session Border Controller performs multiple registrations. The default value is **1**. The valid range is:
   - Minimum—0
   - Maximum—999999999

SIP-IMS Surrogate Registration Proxy Authorization Header for Non-Register Requests

The Oracle Communications Session Delivery Manager's IMS functionality helps customers who use SIP IP PBX or SIP gateways that can only peer with carriers connected to IMS via a P-CSCF. As part of this function, the Oracle Communications Session Delivery Manager provides for generating a Proxy-Authorization or Authorization header for REGISTER requests that are challenged. This feature extends the Oracle Communications Session Delivery

Manager's capabilities by also allowing you to configure the generation of Proxy-Authorization and Authorization headers for non-REGISTER requests.

When you configured it to do so, the Oracle Communications Session Delivery Manager caches Proxy-Authorization or Authorization headers from the most recent (last-sent) messages in the following exchange: REGISTER--407 Proxy Authentication Required--REGISTER--200. Then the system uses these values in the subsequent requests. The following methods are supported:

- INVITE
- ACK
- BYE
- CANCEL
- UPDATE
- INFO
- PRACK
- OPTIONS

The Oracle Communications Session Delivery Manager updates the following parameters when it generates the header:

- nonce-count—Incremented for every new request the Oracle Communications Session Delivery Manager receives
- response—Contains the digest-request, newly generated using the cnonce, nonce, and other fields as input

In addition, the system supports the nonce text parameter in the Authentication-Info header. And for surrogate registration, it recognizes the Authentication-Info header in 200 OK responses received from the UAS and updates its cached nonce value accordingly; in this case, the system resets the nonce count to 1 for the subsequent request.

SIP-IMS Surrogate Registration Proxy Authorization Header Configuration

You configure the SIP-IMS surrogate registration proxy authorization header for non-register requests by setting the **options** parameter in the surrogate agent configuration. You set two types of options: **auth-methods** and **auth-info**.

> **Note:**
>
> If authentication of any SIP requests other than REGISTER is required, then the surrogate-agent option auth-methods MUST be configured. Supported methods are INVITE, ACK, BYE, CANCEL, UPDATE, INFO, PRACK, OPTIONS. For example:
>
> ```
> ORACLE(surrogate-agent)# options +auth-methods='INVITE,OPTIONS"
> ```

To enable SIP-IMS Surrogate Registration Proxy Authorization Header for Non-Register Requests:

1. In Superuser mode, type configure terminal and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ACMEPACKET(session-router)#
```

3. Type **surrogate-agent** and press Enter.

```
ORACLE(session-router)# surrogate-agent
ORACLE(surrogate-agent)#
```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name. Then type the equal sign (=), open quotation mark (“), the list of methods you want supported separated by commas, and closed quotation mark (”). Then press Enter. Valid values are INVITE, ACK, BYE, CANCEL, UPDATE, INFO, PRACK, OPTIONS. Default is blank.

```
ORACLE(surrogate-agent)# options +auth-methods=’INVITE,CANCEL,ACK,BYE”
ORACLE(surrogate-agent)# options +auth-info=refresh
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration’s options list, you must prepend the new option with a plus sign as shown in the previous example.

SIP Surrogate Agent Registration Re-initialization

Surrogate agents in the IMS network integrate IP Private Branch Exchange (PBX) that cannot register itself to the registrar. The Oracle Communications Session Border Controller supports registration retry attempts to the IMS core up to the configured maximum register attempts before failing. When the Oracle Communications Session Border Controller identifies an unregistered surrogate agent, the registration cache can cleared as a normal SIP user and the registration process re-started.

Surrogate Agents may not successfully register before the **max-register-attempts,** causing an unresponsive agent. Commands identify these unregistered agents and clear them from cache. New registration attempts are then available.

Command line option **show registration sipd surrogate-agent <realm-id/unregistered>** with no additional arguments displays all surrogate agents and their state.The system displays the last time of registration for each agent.

- realm id—displays all surrogate agents present in the selected realm.
- unregistered— displays all unregistered surrogate agents.

The preceding commands have the **to-file** option to redirect the output to a file

Possible values for the **Last Registered at** and **State** fields

User Contacts
Displays the registered SIP URI for the surrogate agent. In case of surrogate agents configured with block of numbers, one user contact (SIP NUI) for each number within the block of numbers.

Last Registered at
Displays the last time when the agent registered with the Registrar. This time will change with every renewal registry at expiration. If the surrogate user has not successfully registered, this field will say **never registered**

State
Displays the current state of the surrogate agent. This **State** field confirms the state of two timers used when a surrogate agent is being configured: 1. **register-expires** and 2. **register-retry-time**. When the **register-retry-time** expires, the state will be set to **unregistered** as it continues to try to register. When the **register-expires** time expires and **max-register-attempts** is exhausted, the state is set to **failed**.

Last Registered At	State	Description
<time>	registered	Surrogate agent valid and registered
<time>	failed	Surrogate agent registered, but failed to register refresh and is in an invalid state
<time>	unregistered	Surrogate agent registered, now server is not responding but continues to send REGISTER
Never registered	failed	Surrogate agent failed to register.

When there are no surrogate agents configured on the Oracle Communications Session Border Controller , the output for **show registration sipd surrogate-agents, show registration sipd surrogate-agents by-realm <realm-id>** and **show registration sipd surrogate-agents unregistered** shows "No matching entries found!".

When all the surrogate agents are registered and none in an unregistered state, the output for **show registration sipd surrogate-agents unregistered** shows "No matching entries found!".

Existing ACLI command **clear-cache registration sipd** restarts the registration process for the surrogate agent.

Surrogate Agent Reregistration to SAG Member Handling

In a round-robin environment sometimes requests need to go to the same server as the initial request. The Oracle Communications Session Border Controller will direct the reREGISTER request to a 401/407 response for a surrogate agent to the same Proxy Call Session Control Function(P-CSCF) as the prior REGISTER request for security purposes.

Round robin environments are commonly implemented to balance traffic on multiple Proxy Call Session Control Functions(P-CSCFs). While this strategy is effective, it does also introduce complexity to scenarios in which some consecutive requests need to be directed to the same target. When an initial reREGISTER is challenged, the Oracle Communications Session Border Controller directs the reREGISTER request for a surrogate agent to the same Proxy Call Session Control Function(P-CSCF) as the prior REGISTER request. If the reREGISTER were to go to another server, it would never be accepted. The routing decision for this reREGISTER is made independent of local policy.

No configuration is needed to enable this feature. However, the **cache-challenges** parameter in the SIP Config must be enabled (default).

IMS Implicit Service Route

The Oracle Communications Session Border Controller provides implicit service route support in situations where it is deployed between user equipment (UE) and the P-CSCF, and where the IMS core network does not support the Service-Route header.

When this feature is enabled, the Oracle Communications Session Border Controller sends requests to the P-CSCF and does not include the Service-Route header received in the 200 OK response (to a REGISTER message) as Route headers in subsequent requests. The Oracle Communications Session Border Controller also includes a Route header of the P-CSCF address in subsequent requests, and includes the loose-route parameter in the Route header. Because inclusion of the loose-route parameter is not needed in all cases, you can set this feature to strict in the SIP interface configuration.

Recent enhancements address the following issues:

- Even when IMS is disabled for a SIP interface, the Oracle Communications Session Border Controller caches and uses the Service-Route headers from SIP REGISTER responses received from the REGISTER. Therefore, you must use SIP HMR to remove the Service-Route headers from the response, while having no mechanism to replace the Service-Routes from the REGISTER response with an implicit Service Route.
- In the Oracle Communications Session Border Controller global SIP configuration, the presence of the option route-registrar-no-service-route sets the behavior for using the Service-Route header in the REGISTER request. The new enhancements greatly simplify the process of determining proper use of the header in both IMS and non-IMS environments.
- You can configure the Oracle Communications Session Border Controller with an option to keep it from using the Service-Route header for REGISTER requests when sent to an out-of-service session agent. The enhancements make this behavior the default—because otherwise these REGISTER requests fail.
- If the initial re-registration is challenged, the subsequent REGISTER will be sent to the same target as its prior REGISTER.

When implicit service route support is enabled, the Oracle Communications Session Border Controller stores the Service Route URIs from the Service-Route headers that are included in 200 OK responses to REGISTER messages. The Service Route URIs are included in the Route headers in subsequent Request messages, except for REGISTER messages.

The Oracle Communications Session Border Controller also supports the ability to keep the loose-route parameter from being included in the implicit Route URI that the Oracle Communications Session Border Controller generates and includes as a Route header in the Request messages.

Once an endpoint registers successfully, the Oracle Communications Session Border Controller caches the Service-Route header (if any) to use for routing all subsequent requests from the endpoint—with the exception of any subsequent REGSITER requests.

You can set whether or not you want the Oracle Communications Session Border Controller to route subsequent REGISTER requests using the cached Service Route, and whether the endpoint is engaged in an active session through the Oracle Communications Session Border Controller. If you decide not to use the Service Route for endpoints engaged in active sessions, then the Oracle Communications Session Border Controlleruses the local policy to make routing decisions.

For endpoints not in found in the Oracle Communications Session Border Controller's registration cache, the Oracle Communications Session Border Controller again uses the local policy to make routing decisions.

IMS Implicit Service Route Configuration

To configure implicit service route support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. **implicit-service-route**—To enabled implicit service route support, change this parameter from **disabled** to **enabled**. The default value is **disabled**. Supported values are:

   - absent | disabled (the default) | enabled | replace | strict
   - absent—An implicit service route to the session agent to which the REGISTER request was sent is constructed when the successful REGISTER response contains no Service-Route headers.
   - disabled (default)—Turns off the implicit service route feature; Oracle Communications Session Border Controller constructs service route the Service-Route headers in a successful REGSITER response.
   - enabled—Turns on this feature, meaning that an implicit service route to the session agent to which the REGISTER request was sent is inserted in front of Service-Route header in a successful REGISTER response; the inserted URI includes the ;lr parameter if the session agent has loose routing enabled.
   - replace—An implicit service route to the session agent to which the REGISTER request was sent is used to construct the service route. The Oracle Communications Session Border Controller ignores Service-Route headers in successful REGISTER responses.
   - strict—An implicit service route to the session agent to which the REGISTER request was sent is inserted in front of Service-Route header in a successful REGISTER response; the inserted URI does not the ;lr parameter if the session agent has loose routing enabled, overriding the loose routing behavior configured for the session agent.

     Save and activate your configuration.

IMS Service Route Configuration

To configure how you want the Service Route used:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **register-service-route**—Enter the way you want the Oracle Communications Session Border Controller to use the service route:
   - always—The Oracle Communications Session Border Controller always uses the cached service route for an endpoint for routing REGISTER requests.
   - never—The Oracle Communications Session Border Controller never uses the service route, and makes routing decisions based on local policies instead.
   - removal—The Oracle Communications Session Border Controller uses the cached service route for an endpoint when routing REGISTER requests that remove the endpoint's contact. It uses the local policy for refresh and query REGISTER requests.
   - session—The Oracle Communications Session Border Controller uses the cached service route when routing REGISTER requests that appear while an endpoint has an active session traversing it. When an endpoint does not have an active session, the Oracle Communications Session Border Controller uses the local policy to make routing decisions.
   - session+removal—Combining the session and removal values, the Oracle Communications Session Border Controller uses the cached service route: when routing REGISTER requests that remove the endpoint's contact and when REGISTER requests appear while while an endpoint has an active session traversing the system. Otherwise, the Oracle Communications Session Border Controller uses the local policy to make routing decisions.
5. Save and activate your configuration.

Notes About Upgrading

There are Oracle Communications Session Border Controllers currently deployed that use the route-registrar-no-service-route option, and these enhancements provide for backward compatibility.

When you upgrade to a release that has the new register-service-route parameter in the SIP configuration, the system checks for the presence of the route-registrar-no-service-route option. If the system finds the option, then it translates the value configured for the option like this:

Old route-registrar-no-service-route value	New register-service-route value
<empty> or all	never
refresh	removal
all, idle	session
refresh; idle	session+removal

You must save your configuration for these changes to take place, allowing you to fall back to the previous software image.

IMS Charging Vector Mode Adaptation

This adaptation to the Oracle Communications Session Border Controller's IMS functionality provides the ability to remove the P-Charging-Vector from incoming requests for a session and store it. Then the Oracle Communications Session Border Controller inserts it into outbound responses related to that session in a P-Charging-Vector header.

IMS Charging Vector Mode Configuration

Typically, the ACLI **charging-vector-mode** parameter is set to **delete-and-respond** (which supports removing and storing the P-Charging-Vector for later insertion in outbound response) on the core, trusted interface. On the access, untrusted side, this same parameter is set to **insert**.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing SIP interface, you need to select and edit that configuration.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **charging-vector-mode**—Change this parameter to **delete-and-respond** to remove the P-Charging-Vector from incoming requests for a session and store it. Then the Oracle Communications Session Border Controller inserts it into outbound responses related to that session in a P-Charging-Vector header.
5. Save and activate your configuration.

IMS P-CSCF Endpoint Identification Using Address and Port

You can configure the Oracle Communications Session Border Controller, acting as a P-CSCF, to match a Request it receives to a registration cache entry based only on the IP address and port from which the Request came. When you enable this behavior, the Oracle Communications Session Border Controller will perform this kind of endpoint identification even when there nothing in the message matches the cache entry.

IMS P-CSCF Endpoint Identification Configuration

For this behavior to work as designed, you must also have the **reg-via-key** option enabled for the SIP interface to which you are adding the **reg-via-match** option.

To configure P-CSCF endpoint identification using address and port:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter. If you are editing an existing configuration, select the one on which you want to enable this feature.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **reg-via-match** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-interface)# options +reg-via-match
```

   If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to this** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

5. Save and activate your configuration.

IMS-AKA

The Oracle Communications Session Border Controller supports IP Media Subsystem-Authentication and Key Agreement (IMS-AKA).

Defined in 3GPPr7 (specifications in TS 33.203 and call flows in TS 24.228), IMS-AKA can be used as a framework for authentication and for securing the signaling path between a UE and the Oracle Communications Session Border Controller (when the Oracle Communications Session Border Controller is acting as a P-CSCF or as a B2BUA) across the Gm interface.

In addition, the Oracle Communications Session Border Controller's serving as an IMS-AKA termination point is valuable because it allows IMS-AKA use behind by multiple endpoints sitting behind a NAT device. IMS-AKA support also works when there are no NAT devices between endpoints and the Oracle Communications Session Border Controller acting as a P-CSCF, and when the Oracle Communications Session Border Controller sits behind a third-party P-CSCF. In addition, you can use IMS-AKA when the endpoint uses SIP UDP.

Requirements

IMS-AKA use assumes that you have installed the appropriate IPSec module on your SBC, or that it has come from Oracle with those modules pre-installed. IMS-AKA will not work without this hardware.

IMS-AKA deployments require an activated **network-parameters** element configured with the options shown below.

```
options                          atcp-rxmt-count=2
                                 atcp-rxmt-interval=2
                                 atcp-syn-rxmt-interval=2
                                 atcp-syn-rxmt-maxtime=6
                                 atcp-idle-timer=3700
```

In addition, your configuration must have SIP registration caching enabled.

IMS-AKA Socket Cleanup

To ensure that the SBC properly removes idle IMS-AKA sockets, you can set the **cleanup-inactive-imsaka-tcp-socket** option. This option generates the cleanup logic when you also set the **inactivity-conn-timer** on the access side **sip-interface**. When you configure this option, the SBC:

- Increments idle connection timer of the core side registration expiry value for sip service sockets that are TCP, are created with an IMS-AKA profile, and are not expecting more data.
  Sets the idle connection timer for the 5060 unsecured TCP socket, the secure inbound TCP socket, and the secure outbound TCP socket for IMS-AKA to the **core side reg-expiry** value plus the value you configured in the **inactivity-conn-timer** parameter.
- Resets this inactivity time every time it has a Send or Recv event on the SipService socket.
- Disconnects the service socket when it detects no activity for that amount of time.

The refreshRegForward Option

The Oracle Communications Session Border Controller provides a the user with a means of ignoring its registration refresh half-life timer, and send all applicable registration refreshes received via IMS-AKA to the core for authentication.

By default, the Oracle Communications Session Border Controller uses its half-life function and attempts to manage registration refreshes prior to half-life expiry without forwarding the refresh to the core. The Oracle Communications Session Border Controller sends registration refreshes that arrive after the half-life expiry to the core.

The user changes this behavior by setting the **refreshRegForward** in the applicable IMS-AKA profile to as follows.

```
ORACLE(ims-aka-profile)# options +refreshRegForward
```

When this option is set, the system forwards every refresh registration to the IMS core regardless of the half-life timer's status.

Monitoring

The ACLI **show sipd endpoint-ip** command is updated to show the IMS-AKA parameters corresponding to each endpoint. The display shows the algorithms used, the ports used, and the security parameter indexes (SPIs) used.

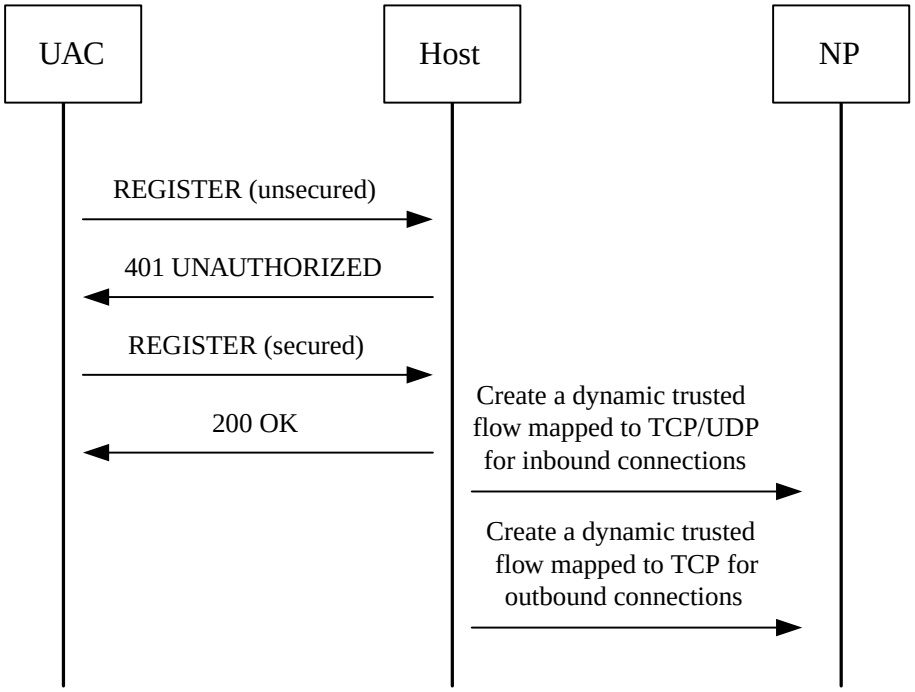
In addition, the **show sa stats** command now shows the security associations information for IMS-AKA.

DDoS for IMS-AKA

The Oracle Communications Session Border Controller (SBC) supports DDoS protection for IMS-AKA. This can be enabled on the realm interface for the access network when the **access-control-trust-level** configuration element is set to **low** or **medium**.

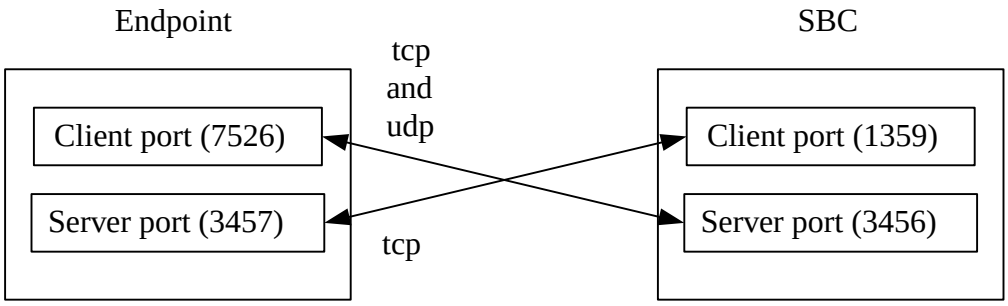
The SBC's DDoS protection for IMS-AKA is identical to regular DDoS protection except for the following:

- The SBC installs two dynamic trusted flows in its network processor (NP) as soon as the user agent client (UAC) completes registration with a 200 OK.



Because both flows are trusted, this ensures that the signaling from authenticated IMS-AKA endpoints will not be dropped even during a DDoS attack.

- Rather than installing multiple flows for different protocols, the SBC installs two protocol-aware flows. One flow covers the TCP and UDP traffic from or to the endpoint client port; the other flow covers the TCP traffic from or to the endpoint server port. This allows the SBC to avoid size limitations in the NAT endpoint tables.



As with other trusted flows, the SBC enforces the configured thresholds:

- invalid-signal-threshold
- maximum-signal-threshold
- deny-period
- cac-failure-threshold
- untrust-cac-failure-threshold
- wait-time-for-invalid-threshold (if the IDS feature is enabled)

ACLI Instructions and Examples

You enable IMS-AKA by configuring the following:

- An IMS-AKA profile
- Certain parameters in the global IPSec configuration
- Certain parameters in the SIP interface, and in the SIP interface's SIP port

Setting Up an IMS-AKA Profile

An IMS-AKA profile establishes the client and server ports to be protected, and it defines lists of encryption and authentication algorithms the profile supports. You can configure multiple IMS-AKA profiles, which are uniquely identified by their names.

You apply an IMS-AKA profile to a SIP port configuration using the name.

To configure an IMS-AKA profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **ims-aka-profile** and press Enter.

   ```
   ORACLE(system)# ims-aka-profile
   ORACLE(ims-aka-profile)#
   ```

4. **name**—Enter the name you want to give this IMS-AKA profile. This is the value you will use to apply the profile to a SIP port configuration. This parameter is required, and it has no default value.
5. **protected-server-port**—Enter the port number of the protected server port, which is the port on which the Oracle Communications Session Border Controller receives protected messages. The protected server port should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.

6. **protected-client-port**—Enter the port number of the protected client port, which is the port on which the Oracle Communications Session Border Controller sends out protected messages. Like the protected server port, the protected client port should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

   This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.

7. **encr-alg-list**—Enter the list of encryption algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   ```
   ORACLE(ims-aka-profile)# encr-alg-list "aes-cbc null"
   ```

   This parameter defaults to the following three values: **aes-cbc**, **des-ede3-cbc**, and **null**.

8. **auth-alg-list**—Enter the list of authentication algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   ```
   ORACLE(ims-aka-profile)# auth-alg-list "hmac-sha-1-96 hmac-md5-96"
   ```

   This parameter defaults to hmac-sha-1-96.

Setting Up an IPSec Profile for IMS-AKA Use

Using the global IPSec configuration, you establish the parameters governing system-wide IPSec functions and behavior. This configuration also contains parameters required for IMS-AKA support. The IPSec global configuration is a single instance element, meaning there is one for the whole system.

To configure the global IPSec parameters that apply to IMS-AKA:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **ipsec** and press Enter.

   ```
   ORACLE(system)# ipsec
   ORACLE(ipsec)#
   ```

4. Type **ipsec-global-config** and press Enter. If you are editing a pre-existing IPsec global configuration, then you need to select the configuration before attempting to edit it.

   ```
   ORACLE(system)# ipsec-global-config
   ORACLE(ipsec-global-config)#
   ```

5. **red-ipsec-port**—Specify the port on which the Oracle Communications Session Border Controller should listen for redundancy IPSec synchronization messages. The default is 0, and valid values are either 0 or 1994.
6. **red-max-trans**—Enter the maximum number of redundancy transactions to retain on the active. The default is 10000, and valid values range up to a 999999999 maximum.
7. **red-sync-start-time**—Enter the time in milliseconds before the system starts to send redundancy synchronization requests. The default is 5000, and valid values range up to a 999999999 maximum.
8. **red-sync-comp-time**—Enter the time in milliseconds to define the timeout for subsequent synchronization requests once redundancy synchronization has completed. The default is 1000, and valid values range up to a 999999999 maximum.

Enabling IMS-AKA Support for a SIP Interface

To enable IMS-AKA for a SIP interface, you must set the **sec-agree-feature** parameter to enabled.

To enable IMS-AKA for a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing SIP interface, you need to select and edit that configuration.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **sec-agree-feature**—Change this parameter to **enabled** if you want to use IMS-AKA on this SIP interface. By default, this parameter is **disabled**.

Applying an IMS-AKA Profile to a SIP Port

The final step in setting up IMS-AKA support is to apply an IMS-AKA profile to a SIP port. Enter the **name** value from the IMS-AKA profile you want to apply in the SIP port's **ims-aka-profile** parameter.

To apply an IMS-AKA profile to a SIP port:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing SIP port, you need to select and edit that configuration.

```
ORACLE(session-interface)# sip-ports
ORACLE(sip-port)#
```

5. **ims-aka-profile**—Enter the **name** value for the IMS-AKA profile configuration you want applied to this SIP port. This parameter has no default.
6. Save and activate your configuration.

IPSec IMS-AKA

Compliance with the VoLTE specification (GSMA PRD IR.92) requires cluster member support for IPsec IMS-AKA (IP Multimedia Services Authentication and Key Agreement) as defined in 3GPP TS 24.299, *IP Multimedia Call Control Protocol Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)*: Stage 3, and TS 33.203, *3G Security: Access Security for IP-based Services*.

Support for IMS-AKA requires no new additional configuration elements.

Sample IMS-AKA Configuration

The following formatted extract from **show running-config** ACLI output shows a sample IMS-AKA profile configuration.

```
ims-aka-profile
name                            dut2.test
protected-client-port           4060
protected-server-port           4060
encr-alg-list                   aes-cbc des-ede3-cbc
auth-alg-list                   hmac-sha-1-96
last-modified-by                admin@172.30.11.18
last-modified-date              2012-01-10 17:31:59
```

Sample Security Policy Configuration

The following formatted extracts from **show running-config** ACLI output shows three associated security policies.

The first policy, and the one with the highest priority, opens Port 5060 for SIP traffic.

```
security-policy
name                            pol1
```

```
network-interface              M10:0.6
priority                       0
local-ip-addr-match            3fff:c0ac::c0ac:ce12
remote-ip-addr-match           ::
local-port-match               5060
local-port-match-max           5060
remote-port-match              0
trans-protocol-match           ALL
direction                      both
local-ip-mask                  ::
remote-ip-mask                 ::
action                         allow
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask                  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
local-port-mask                65535
remote-port-mask               65535
trans-protocol-mask            0
valid                          enabled
vlan-mask                      0xFFF
last-modified-by               admin@console
last-modified-date             2012-01-10 17:48:59
```

The second policy opens Port 4444 for CCP traffic.

```
security-policy
name                           pol2
network-interface              M10:0.6
priority                       2
local-ip-addr-match            3fff:b623::b623:ce02
remote-ip-addr-match           3fff:b623::b623:ce01
local-port-match               4444
local-port-match-max           4444
remote-port-match              4444
remote-port-match-max          4444
trans-protocol-match           ALL
direction                      both
local-ip-mask                  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
action                         allow
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask                  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
local-port-mask                65535
remote-port-mask               65535
trans-protocol-mask            0
valid                          enabled
vlan-mask                      0xFFF
last-modified-by               admin@console
last-modified-date             2012-01-10 17:49:15
```

The third policy, the policy with the least priority, and, consequently, the last policy applied, requires IPsec on all ports.

```
security-policy
name                            pol3
network-interface               M10:0.6
priority                        10
local-ip-addr-match             3fff:c0ac::c0ac:ce12
remote-ip-addr-match            ::
local-port-match                0
remote-port-match               0
trans-protocol-match            ALL
direction                       both
local-ip-mask                   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                  ::
action                          ipsec
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask                   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
local-port-mask                 65535
remote-port-mask                65535
trans-protocol-mask             0
valid                           enabled
vlan-mask                       0xFFF
last-modified-by                admin@console
last-modified-date              2012-01-10 17:50:42
```

Sec-Agree

The SBC supports RFC 3329, Security Mechanism Agreement for the Session Initiation Protocol, commonly referred to as Sec-Agree. The RFC defines three SIP headers, Security-Client, Security-Server, and Security-Verify that provide the ability for SIP UAs and other SIP entities (servers, proxies, and registrars) to negotiate next-hop security mechanisms. Note that this initial implementation does not provide support for server-initiated security negotiation, nor does it support media-plane security. That is, support is limited to client-initiated negotiation during initial registration, and to signalling security.

Currently the P-CSCF functionality includes support for IMS-AKA feature for VoLTE deployments. In order to support RCS clients along with VoLTE P-CSCF functionality needs to be enhanced to support RFC 3329, Security Mechanism Agreement for the Session Initiation Protocol (commonly referred to as Sec-Agree), which includes support for TLS as security mechanism.

Sec-Agree defines three SIP headers, Security-Client, Security-Server and Security-Verify, to negotiate security agreements during initial REGISTER transactions. Header definitions are as follows:

```
security-client  = "Security-Client" HCOLON
                           sec-mechanism *(COMMA sec-mechanism)
security-server  = "Security-Server" HCOLON
                   sec-mechanism *(COMMA sec-mechanism)
security-verify  = "Security-Verify" HCOLON
                   sec-mechanism *(COMMA sec-mechanism)
sec-mechanism    = mechanism-name *(SEMI mech-parameters)
```

```
mechanism-name   = ( "digest" / "tls" / "ipsec-ike" /
                     "ipsec-man" / token )
mech-parameters  = ( preference / digest-algorithm /
                     digest-qop / digest-verify / extension )
preference       = "q" EQUAL qvalue
qvalue           = ( "0" [ "." 0*3DIGIT ] )/ ( "1" [ "." 0*3("0") ] )
digest-algorithm = "d-alg" EQUAL token
digest-qop       = "d-qop" EQUAL token
digest-verify    = "d-ver" EQUAL LDQUOT 32LHEX RDQUOT
extension        = generic-param
```

The Security-Client header contains one or more security mechanisms and associated parameters proposed by the initiating client. This initial implementation supports two security mechanisms: TLS and ipsec-3gpp.

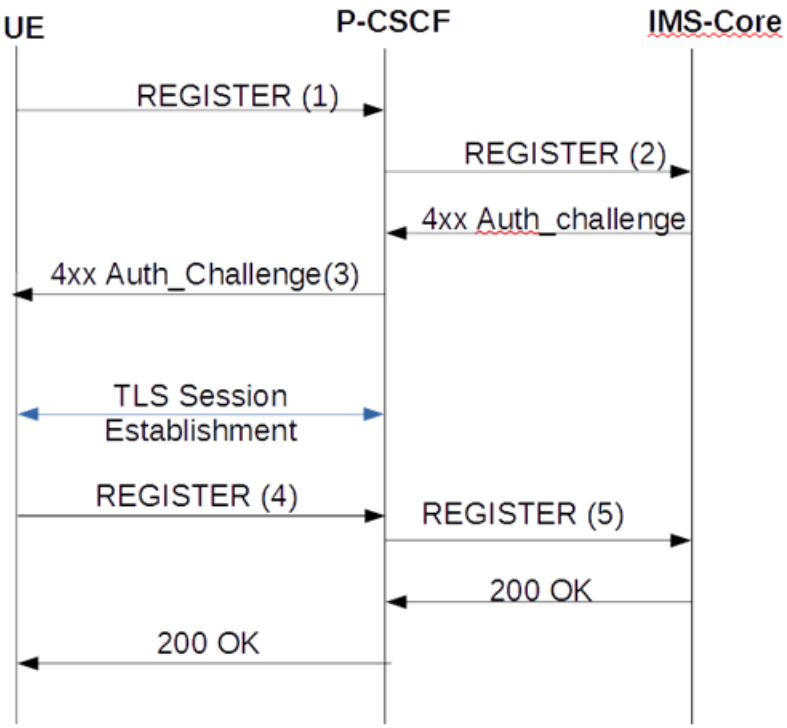
The Security-Server header contains the security mechanism chosen by the server from those mechanisms proposed by the client.

The Security-Verify header contains the contents of the Security-Server header.

Two additional header fields, Require and Proxy-Require, are also used in support of Sec-Agree negotiations. Both headers are required in client transmissions.

TLS Session Setup During Registration

This call flow depicts a TLS session setup during the registration procedure. Only relevant header fields are noted.



(1) REGISTER

```
Proxy-Require:sec-agree
Security-Client: ipsec-3gpp;alg=sha2-512;ealg=aes-cbc;prot=esp;mod=trans;spi-
```

```
c=8765423;port-c=7524;spi-s=1234563;port-s=1358, ipsec-3gpp;alg= hmac-
sha-1-96;ealg=aes-cbc;prot=esp;mod=trans;spi-c=8765423;port-c=7524;spi-
s=1234563;port-s=1358, tls
```

(2) REGISTER

```
Authorization: Digest
uri="sip:ims.mnc007.mcc262.3gppnetwork.org",username="262073900320132@ims.mnc0
07.mcc262.3gppnetwork.org",response="",realm="ims.mnc007.mcc262.3gppnetwork.or
g",nonce=""
(No integrity-protected field will be present if TLS is selected as the
security mechanism)
```

(3) 401

```
Security-Server: tls
```

(4) REGISTER

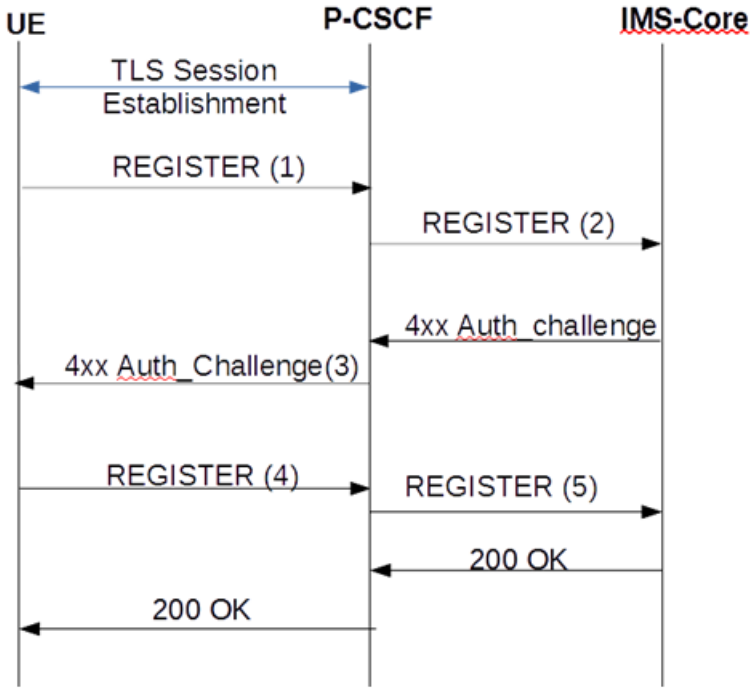
```
Proxy-Require: sec-agree
Security-Verify: tls
```

(5) REGISTER

```
Authorization: Digest
username="262073900320132@ims.mnc007.mcc262.3gppnetwork.org",realm="imstest1.t
elefonica.de",uri="sip:ims.mnc007.mcc262.3gppnetwork.org",qop=auth,nonce="Ms8l
2xeF3l4lb0VO8fK3KFMSKLv1sQAATdN2NpFUCgU=",nc=00000001,cnonce="3063397945",algo
rithm=AKAv1-MD5,response="3779ff40a057f999a2f8288bbfafc10d", integrity-
protected=tls-pending
```

TLS Session Setup Prior to Registration

This call flow depicts a TLS session setup prior to the registration procedure.



(1) REGISTER

```
(No Security-Client or Proxy-Require header present)
```

(2) REGISTER

```
Authorization: Digest
uri="sip:ims.mnc007.mcc262.3gppnetwork.org",username="262073900320132@ims.mnc0
07.mcc262.3gppnetwork.org",response="",realm="ims.mnc007.mcc262.3gppnetwork.or
g",nonce=""
(No integrity-protected field will be present)
```

(3) 401

```
(No Security-Server header present)
```

(4) REGISTER

```
(No Security-Client, or Security-Verify or Proxy-Require header present)
```

(5) REGISTER

```
Authorization: Digest
username="262073900320132@ims.mnc007.mcc262.3gppnetwork.org",realm="imstest1.t
elefonica.de",uri="sip:ims.mnc007.mcc262.3gppnetwork.org",qop=auth,nonce="Ms8l
2xeF3l4lb0VO8fK3KFMSKLv1sQAATdN2NpFUCgU=",nc=00000001,cnonce="3063397945",algo
rithm=AKAv1-MD5,response="3779ff40a057f999a2f8288bbfafc10d", integrity-
protected=tls-pending
```

Regardless of TLS Session setup procedure, if the newly added configurable item sec-agree feature is enabled, any messages on unprotected port will be rejected except REGISTER messages or messages related to emergency services.

For refresh registration, if the Sec_Agree occurred during Registration, it verifies for the presence or change of Security-Client & Security-Verify headers, if they differ it will be rejected with 4xx response and also Authorization header fields are verified irrespective of the above methods and if they differ with previous association it will be rejected with 403 (Forbidden) response. Also when the refresh REGISTER is being forward to the core, it will set the integrity-protected field to "tls-yes".

SEC-agree Configuration

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **sec-agree-feature**—Set this parameter to enable or disable for Sec-Agree support. By default, support is disabled.
4. **sec-agree-pref**—Configure this parameter to specify security protocol preferences.
   - **ipsec3gpp** — support only IMS-AKA protocol
   - **tls** — support only TLS protocol
   - **ipsec3gpp-tls** — support both IMS-AKA and TLS, preferred protocol is IMS-AKA
   - **tls-ipsec3gpp** — support both TLS and IMS-AKA, preferred protocol is TLS
5. Type **done** to save your configuration.

IMS AKA over TCP

IMS-AKA registration is conducted over UDP or TCP protocol only. The Oracle Communications Session Border Controller supports both transport protocols.

Within mobile IMS VoLTE/RCSe deployments, IP packets carrying SIP messages can be large due to IPv6 headers, IMS-AKA specific headers, extensive codec policies, and other 3GPP related headers. Because of this, IPv6 VoLTE signaling messages using IMS-AKA frequently exceed 1300 bytes and require TCP according to RFC3261 section 18.1.1.

IMS-AKA Secure Call Registration over TCP

To register and place a call into the network, a UE creates 3 TCP connections. The first insecure connection is established to the port (usually 5060) specified in the sip-port for the

first registration request. You should create a sip-ports configuration element with port 5060 and an ims-aka-profile parameter that references an ims-aka-profile configuration element. The ims-aka-profile configuration element initiates the process that creates secure connections. For example:

sip-ports

```
address                         sd-ip-address
port                            5060
transport-protocol              TCP
tls-profile
multi-home-addrs
allow-anonymous                 registered
ims-aka-profile                 profile
```

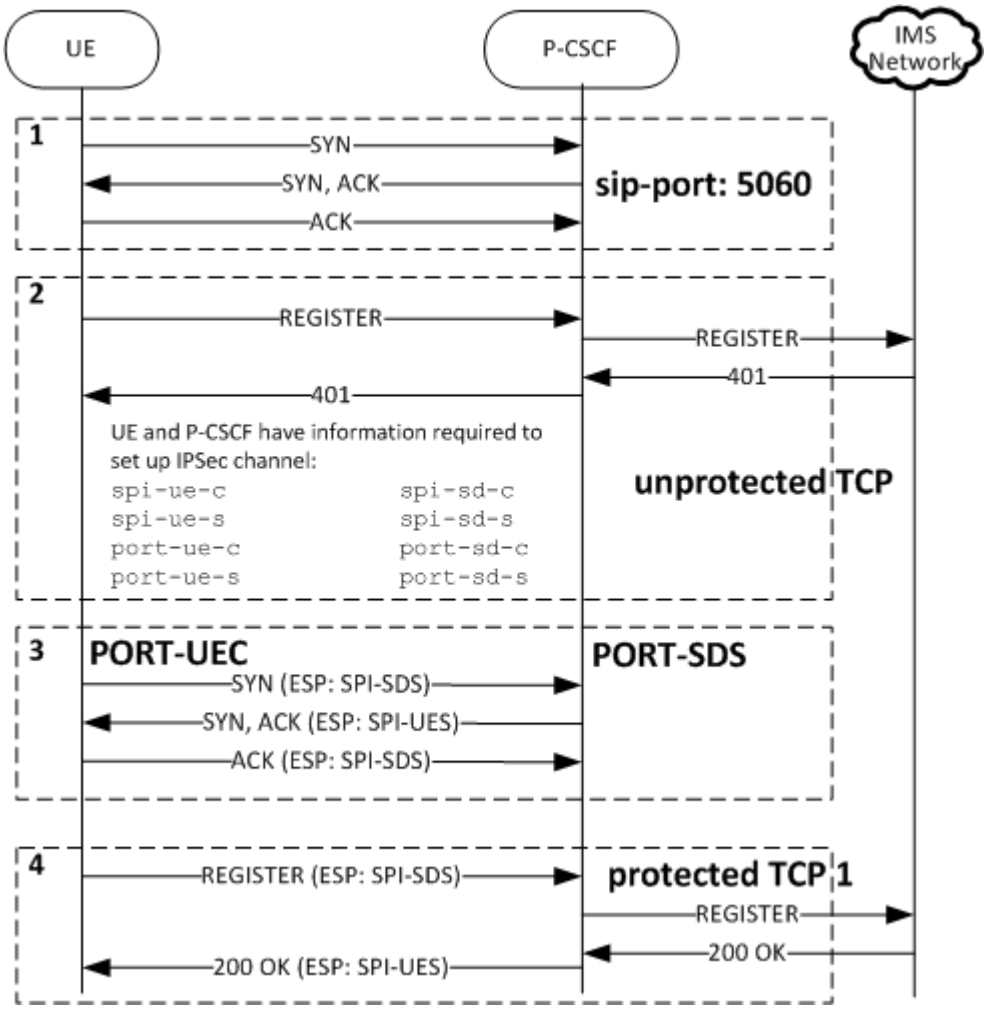
The ims-aka-profile configuration element defines the protected-server-port (PORT-SDS) and protected-client-port (PORT-SDC). The protected-server-port is opened for both inbound TCP and UDP traffic. For example:

ims-aka-profile

```
name                            profile
protected-client-port           PORT-SDC
protected-server-port           PORT-SDS
encr-alg-list                   aes-cbc des-ede3-cbc null
auth-alg-list                   hmac-sha-1-96
```

When the UE receives the 401Unauthorized challenge from the Oracle Communications Session Border Controller acting as P-CSCF, both devices have the information to set up security association for two IPSec channels. The UE establishes the second TCP connection via IPSec channel from the UE's PORT-UEC to the P-CSCF's PORT-SDS, and the registration process continues.

Hereafter, the UE uses the IPSec channels from communication.

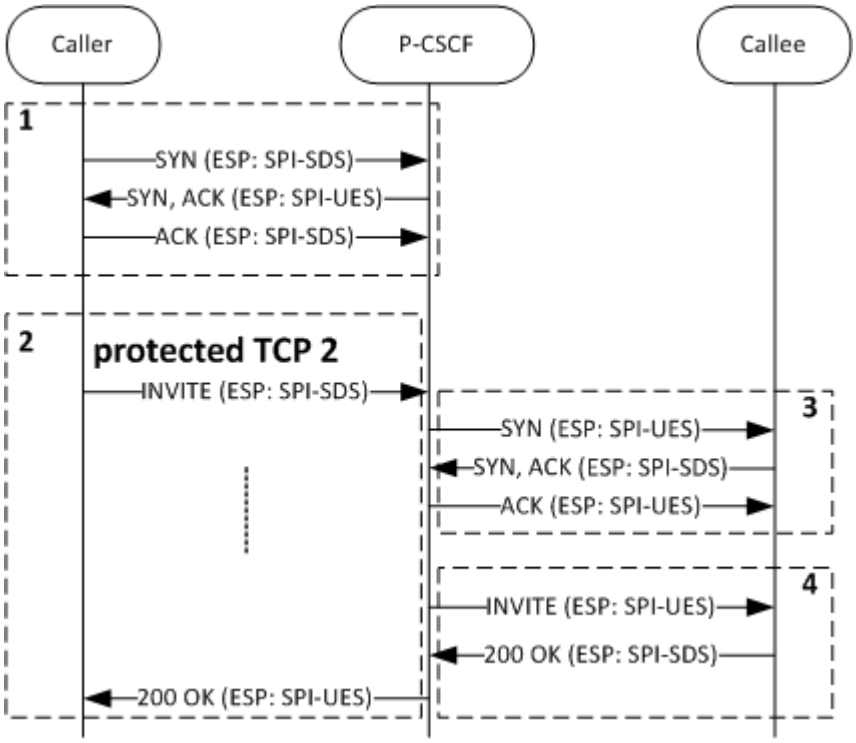


1. The UE and P-CSCF set up the TCP connection.
2. The UE sends an unauthenticated SIP Register message to the P-CSCF's unprotected server port (usually 5060). The Register message is forwarded to the UE's Home S-CSCF. The S-CSCF then replies with a the SIP 401 Authentication Required response back to P-CSCF. This message contains encryption keys and authentication information.

   The P-CSCF modifies the 401 message back to UE. At this point, both UE and P-CSCF should have all the information need to establish secure IPSec channels.
3. The UE and P-CSCF create a TCP connection over a secure channel from port-ue-c to port-sd-s.
4. The UE sends an authenticated REGISTER over the secure channel via the P-CSCF to the S-CSCF. If the authentication is valid, the P-CSCF will forward the 200 OK response from the S-CSCF to the UE. The 200 OK response will be sent in the same secure TCP connection.

IMS-AKA Call Establishment over TCP

For the UE to send a new request into the network and establish a call, a third TCP connection is created using the information configured/generated prior to creating the first secure connection.



1. If the TCP connection over a secure channel does not exist, the Caller will establish it.
2. The Caller initiates the call by sending SIP INVITE from its PORT-UEC to the P-CSCF's PORT-SDS.
3. The P-CSCF forwards the INVITE to the Caller. If not present, it will create a secure TCP connection from its PORT-SDC to the callee's PORT-UES.
4. The INVITE is then forwarded to the Callee securely.

SIP SUBSCRIBE and NOTIFY over TCP IMS-AKA

SUBSCRIBE and NOTIFY messages are exchanged between a UE and the P-CSCF in a manner similar to the previous INVITE example whereby the secure channel is first created and then the SIP messages are exchanged securely.

IMS-AKA Change Client Port

The Oracle Communications Session Border Controller is now in compliance with 3GPP TS 33.203, *Access Security for IP-Based Services*. Previous releases did not comply with requirements specified in Section 7.4, Authenticated re-registration, which reads in part:

Every registration that includes a user authentication attempt produces new security associations. If the authentication is successful, then these new security associations shall replace the previous ones. This clause describes how the UE and P CSCF handle this replacement and which SAs to apply to which message.

When security associations are changed in an authenticated re-registration then the protected server ports at the UE (port_us) and the P-CSCF (port_ps) shall remain unchanged, while the protected client ports at the UE (port_uc) and the P-CSCF (port_pc) shall change.

If the UE has an already active pair of security associations, then it shall use this to protect the REGISTER message. If the S-CSCF is notified by the P-CSCF that the REGISTER message from the UE was integrity-protected it may decide not to authenticate the user by means of the AKA protocol. However, the UE may send unprotected REGISTER messages at any time. In this case, the S-CSCF shall authenticate the user by means of the AKA protocol. In particular, if the UE considers the SAs no longer active at the P-CSCF, e.g., after receiving no response to several protected messages, then the UE should send an unprotected REGISTER message."

Prior releases failed to change the protected client ports after a successful re-registration.

Protected Ports

Within IMS networks, the P-CSCF provides the network access point and serves as the outbound proxy server for user equipment -- smart phones, tablets, and similar devices. The UE must connect to the P-CSCF prior to registration and initiation of SIP sessions. Connection to the P-CSCF, which can be in the user's home network, or in a visited network if the UE is roaming, is accomplished using Dynamic Host Control Protocol (DHCP) P-CSCF discovery procedures.

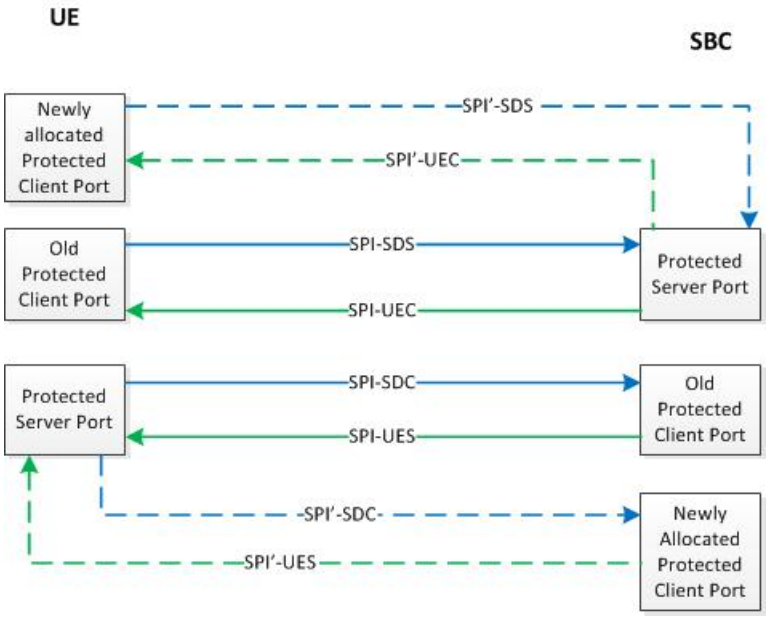
After successful discovery, the P-CSCF and UE negotiate IPSec security associations (SAs) which are used to establish four protected (authenticated and encrypted using Encapsulating Security Payload protocol) ports between the UE and the P-CSCF.

The four protected ports are shown in the following illustration:

As required by Section 7.4 of 3GPP TS 33.203, the protected client ports, one on the UE and the other on the Oracle Communications Session Border Controller, must be changed after each successful re-registration.

To fulfill this requirement, this release adds a new attribute to the existing ims-aka-profile configuration object. This attribute (**end-protected-client-port**) works in conjunction with **start-protected-client-port** (**protected-client-port** in previous releases) to enable the identification of a pool of protected client ports, which will be used for re-registration scenarios where the Oracle Communications Session Border Controller is required to change the client port.

The Oracle Communications Session Border Controller creates new protected client ports, one on the UE and the other on the Oracle Communications Session Border Controller, after every re-registration. Old protected client ports, along with their associated SAs, are maintained for 30 seconds after re-registration to ensure correct handling or any pending responses to previously transmitted messages.



After successful re-registration, the Oracle Communications Session Border Controller updates the registration cache with updated port information and checkpoint with the HA peer, if present.

IMS-AKA Change Client Port Configuration

An IMS-AKA profile establishes the client and server ports to be protected, and it defines lists of encryption and authentication algorithms the profile supports. You can configure multiple IMS-AKA profiles, which are uniquely identified by their names.

You apply an IMS-AKA profile to a SIP port configuration using the name.

To configure an IMS-AKA profile:

1. From Superuser mode, use the following command sequence to navigate to ims-aka-profile configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ims-aka-profile
   ORACLE(ims-aka-profile)#
   ```

2. **name**—Enter the name you want to give this IMS-AKA profile. This is the value you will use to apply the profile to a SIP port configuration. This parameter is required, and it has no default value.
3. **protected-server-port**—Enter the port number of the protected server port, which is the port on which the Oracle Communications Session Border Controller receives protected messages. The protected server port should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

   This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.
4. **start-protected-client-port (protected-client-port in Release S-CX6.3.3M2 and earlier releases)**—Enter the start value for the pool of port numbers available following a

successful re-authentication. Like the protected server port, the protected client port pool should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

Any existing configuration for **protected-client-port** will be mapped to both **start-protected-client-port** and **end-protected-client-port** parameter values.

This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.

5. **end-protected-client-port**—Enter the end value for the pool of port numbers available following a successful re-authentication. Ensure that this value is greater than the value assigned to **start-protected-client-port**. Note that the maximum supported pool contains 5 entries. Like the protected server port, the protected client port pool should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

   This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.

6. **encr-alg-list**—Enter the list of encryption algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   This parameter defaults to the following three values: **aes-cbc**, **des-ede3-cbc**, and **null**.

7. **auth-alg-list**—Enter the list of authentication algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   This parameter defaults to hmac-sha-1-96.

Sample IMS-AKA Configuration

The following formatted extract from **show running-config** ACLI output shows a sample IMS-AKA profile configuration.

```
ims-aka-profile
    name                                TS33.203
    start-protected-client-port         4060
    end-protected-client-port           4064
    protected-server-port               4070
    auth-alg-list                       hmac-sha-1-96
    encr-alg-list                       aes-cbc
```

SIP IMS P-CSCF P-Asserted Identity in Responses

In releases earlier than Release S-C6.1.0, the Oracle Communications Session Border Controller—operating as a P-CSCF—removes the P-Preferred-Identity header (if present) on receipt of a 1xx or 2xx response. It also inserts a P-Asserted-Identity header with the value received in the P-Preferred-Identity header.

Release S-C6.1.0 changes this behavior. Now the Oracle Communications Session Border Controller:

- Caches a copy of the P-Called-Party-ID header when it receives one of the following destined for a UE prior to forwarding the request:
  - An initial request for dialog

  - A request for a standalone transaction
  - A request for an unknown method that does not related to an existing dialog
    The SIP interface receiving the request should have the SIP IMS feature enabled.
- Removes the P-Preferred-Identity header (if present) and inserts a P-Asserted-Identity header with the value saved from the P-Called-Party-ID header on receipt of a 1xx or 2xx response.

Important Notes

Note the following:

- The endpoint to which the response is being sent must be a trusted endpoint. The option **disable-ppi-to-pai** should not be configured in the global SIP configuration's **options** list.
- If the P-Preferred-Identity header is present in the response, the Oracle Communications Session Border Controller will delete the header.
- If the P-Asserted-Identity header is present in the response, the Oracle Communications Session Border Controller will overwrite that -Asserted-Identity.

SIP IMS P-CSCF P-Asserted Identity in Responses Configuration

This behavior is enabled automatically. You do not need to perform any configuration steps.

SIP IMS P-CSCF S-CSCF Target Caching and Invalidation

In IMS architectures, the Oracle Communications Session Border Controller can form these roles:

- An access session border controller, acting as the media front end to a third-party proxy CSCF (P-CSCF)
- Combined access session border controller and P-CSCF

In both, the Oracle Communications Session Border Controller needs to resolve the next-hop signaling element with DNS using these methods: NAPTR resource record, DNS SRV, and DNS address query (A-query).

In addition to this use of DNS, the Oracle Communications Session Border Controller can also now be configured to use DNS for the purposes of load balancing toward the core network elements and resiliently tracking failures. Using DNS for core load balancing simplifies initial provisioning and support more graceful failover when upstream elements fall out of service. Although the DNS-based service route can appear active despite an IP address having failed, the Oracle Communications Session Border Controller provides mechanisms to learn targets and their subsequent invalidation via DNS.

You use session agents to set much of the Oracle Communications Session Border Controller's behavior for S-CSCF target caching and invalidation. The following table describes special ways or change to session agent configuration you must make in order to achieve the desired Oracle Communications Session Border Controller behavior.

Behavior	Configuration
You can configure the SBC to resolve the P-CSCF for a registering UE using a DNS NAPTR query.	Set the applicable session agent's port to 0 and its transport-method to ANY.
You can configure the system to resolve the P-CSCF for a registering UE using a DNS SRV query.	Set the applicable session agent's port to 0 and its transport-method to a value other than ANY.

Behavior	Configuration
You can configure the system to resolve the P-CSCF for a registering UE using a DNS A-query.	Set the applicable session agent's port to value greater than 0.
When you have configured the SBC to resolve the P-CSCF for a registering UE using a DNS NAPTR query, you can establish the way it learns the next hop that will handle the UE's signaling. If multiple NAPTR RRs result or if the order values is the same, the one with the lowest order value becomes the target for registration. If there is no in-service destination available, the NAPTR RR with lowest preference of the remaining options becomes the target.	No special configuration is required.
When you can configure the Oracle Communications Session Border Controller to resolve the P-CSCF for a registering UE using a DNS SRV query, you can establish the way it learns the next hop that will handle the UE's signaling. If the SRV target is an FQDN host that resolves to multiple A RRs, the Oracle Communications Session Border Controller either selects a single A RR by hunting or round robins across multiple A RRs.	To support the round-robin method, you can set the applicable session agent's load-balance-dns-query to round robin. Or, you can leave that parameter set to hunt (its default). Note that the round robin method does not work then the session agent has the dns-load-balance option configured. That option distributed requests to IP address resolved from SRV responses with the same weight and priority. Note that if the ping is enable for the session agent and load-balance-dns-query is set to round robin, the Oracle Communications Session Border Controller pings all the IP addresses resolved from the DNS query.

Acting as a B2BUA Front End to Third-Party P-CSCF

This section describes the support DNS load balancing provides for an Oracle Communications Session Border Controller acting as a B2BUA front end to a third-party P-CSCF.

NAPTR

The Oracle Communications Session Border Controller performs resolution of the P-CSCF for a registering UE using a DNS NAPTR query. This happens when a session agent is defined with its port set to 0 and its transport method set either to any or *, or when there is no session agent defined. When no session agent is defined, the system performs the NAPTR query according to the registering UE's URI domain. And if no transport method is defined for the session agent, the default query type is DNS NAPTR. Note that the configuration of how to resolve the P-CSCF is defined for a specific session agent.

When it receives a UE'S initial registration message, the system performs an NAPTR DNS query for the next hop that will handle the UE's signalling. The Oracle Communications Session Border Controller selects the NAPTR resource record (RR) with the lowest order value as the target for the registration when multiple RRs are returned. If the order value is the same, the system uses the lowest preference RR AS THE target. And if no in-service destination is available, the system selects the RR with the lowest preference from the remaining RRS as the target registration.

DNS SRV

The Oracle Communications Session Border Controller performs resolution of the P-CSCF for a registering UE, starting with a DNS SRV query for a session agent with its port set to 0 and transport method given a setting other than any or *. Note the system provides this support with DNS SRV-based session agents.

When it receives a UE'S initial registration message, the Oracle Communications Session Border Controller performs a DNS SRV query for the next hop target that will handle the UE's signalling. If the SRV target is an FQDN host that then resolves to multiple A RRs, the system then:

- Selects a single A RR through hunting if the **load-balance-dns-query** parameter is set to **hunt** (i.e., the first of several A RRs will always be selected unless it goes out of service)
- Round-robins across multiple A RRs for a given SRV target (if the **load-balance-dns-query** parameter is set to **round-robin**)

By default, the system hunts for and selects a single A RR. If the **dns-load-balance** option is configured, then round-robin will not work. This option distributes the requests to IP addresses that are resolved from SRV responses with the same weight and priority. Also, if you enable the session agent ping and set the **load-balance-dns-query** to **round-robin**, then all IP addresses resolved through the DNS query will be pinged.

Note that when no session agent exists, standard SRV ordering and A record hunting are used.

DNS

The Oracle Communications Session Border Controller generates a DNS query to resolve the P-CSCF for a registering UE if the session is with a port greater than zero (0).

Acting as a P-CSCF

This section describes the support DNS load balancing provides for an Oracle Communications Session Border Controller acting as a P-CSCF.

For each UE registering through the system as a P-CSCF and using any of the three defined methods, the top service route resolved for the UE using DNS is used for subsequent refreshed and INVITE messages without querying DNS or using any selection method. DNS is used in these instances:

- The DNS result TTL expires and the new results are different or changed.
- The cached target associated with the UE is out of service.

For SIP messages the UE initiates other than a REGISTER, the system routes the messages according to the top service route. The DNS-resolved target is used for such messaging.

NAPTR

The Oracle Communications Session Border Controller performs resolution of the I-CSCF for a registering UE with an NAPTR query for a session agent with its port set to 0 and transport method given a setting other than any or *. When no session agent is defined, the system performs the NAPTR query according to the registering UE's URI domain. And if no transport method is defined for the session agent, the default query type is DNS NAPTR. Note that the configuration of how to resolve the P-CSCF is defined for a specific session agent.

When it receives a UE'S initial registration message, the Oracle Communications Session Border Controller performs an NAPTR DNS query for the next hop that will handle the UE's signalling. The system selects the NAPTR resource record (RR) with the lowest order value as the target for the registration when multiple RRs are returned. If the order value is the same, the system uses the lowest preference RR AS THE target. And if no in-service destination is available, the system selects the RR with the lowest preference from the remaining RRS as the target registration.

DNS SRV

The Oracle Communications Session Border Controller performs resolution of the I-CSCF for a registering UE, starting with a DNS SRV query for a session agent with its port set to 0 and transport method given a setting other than any or * or *. Note that the system provides this support with DNS SRV-based session agents.

When it receives a UE'S initial registration message, the system performs a DNS SRV query for the next hop target that will handle the UE's signaling. If the SRV target is an FQDN host that then resolves to multiple A RRs, the Oracle Communications Session Border Controller then:

- Selects a single A RR through hunting if the **load-balance-dns-query** parameter is set to **hunt** (i.e., the first of several A RRs will always be selected unless it goes out of service)
- Round-robins across multiple A RRs for a given SRV target (if the **load-balance-dns-query** parameter is set to **round-robin**)

By default, the system hunts for and selects a single A RR. If the **dns-load-balance** option is configured, then round-robin will not work. This option distributes the requests to IP addresses that are resolved from SRV responses with the same weight and priority. Also, if you enable the session agent ping and set the **load-balance-dns-query** to **round-robin**, then all IP addresses resolved through the DNS query will be pinged.

Note that when no session agent exists, standard SRV ordering and A record hunting are used.

DNS

The system generates a DNS query to resolve the I-CSCF for a registering UE if the session is with a port greater than zero (0).

S-CSCF Configuration

To set the port, transport-method, and load-balance-query parameters for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type and **session-router** press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. If you are editing a pre-existing configuration, you need to select the configuration before making changes.

```
ORACLE((session-router)# session-agent
ORACLE((session-agent)#
```

4. **port**—Change this parameter to the port value you want, or leave it set to 5060 (default).
5. **transport-method**—Leave this parameter set to UDP (default) or enter ANY to support all transport methods.
6. **load-balance-dns-query**—To perform load balancing across multiple A RRs, change this value from **hunt** (default) to **round-robin**.
7. Save and activate your configuration.

E-CSCF Support

An Emergency Call Session Control Function (E-CSCF) is an IMS core element that aids in routing emergency calls to an appropriate destination, such as a PSAP. E-CSCF functionality can be performed by the Oracle Communications Session Border Controller with appropriate local policy and network management control configuration.

The E-CSCF feature let the Oracle Communications Session Border Controller internally prioritize and route emergency calls to the corresponding Emergency Service Center, based either on the calling party's request URI, or based on location information retrieved from a CLF (Connectivity Location Function) for wireline/TISPAN networks.

By integrating E-CSCF functionality into the P-CSCF (Oracle Communications Session Border Controller), networks can satisfy the common local requirement that certain telephony elements be deployed locally, rather than use single, centralized elements. Funcitons like the E-CSCF likely fall into this category.

Service URN Support

To enable E-CSCF functionality, the Oracle Communications Session Border Controller can parse service URNs for local policy lookup keys, and as desitnation identifiers in network management controls (NMC). Ensure that the match-URN is entered correctly as: urn:service:sos" or "urn:service:sos.type or the Oracle Communications Session Border Controller will interpret the URN as a hostname. Please see RFC 5031 for more information on compliant URN construction.

E-CSCF Configuration Architecture

There are four elements which comprise and enable E-CSCF support on the Oracle Communications Session Border Controller :

- CLF Connectivity
- NMC Emergency Call Control
- Local Policy
- Emergency Local Route Table

CLF Connectivity

The Oracle Communications Session Border Controller must be configured with Diameter-based CLF support. This is accomplished by creating an appropriate external policy server configuration.

When the Oracle Communications Session Border Controller requests authorization from the CLF server, a Line-Identifier AVP which includes a location string is expected to be returned for the call. The returned location string will be used later for an LRT query.

NMC Emergency Call Control

By configuring a Network Management Control (NMC), the Oracle Communications Session Border Controller can flag a call for special priority early after it is received and validated by the system. The **destination identifier** must be configured in the NMC with the service URN of an incoming emergency call. Also, the NMC configuration must have its **next hop** parameter left blank. This lets the Oracle Communications Session Border Controller route the emergency call with local policies.

For example, if **urn:service:sos** is the configured value in the NMC's **destination identifier**, and an INVITE arrives on the Oracle Communications Session Border Controller with urn:service:sos in the request URI, the call will be flagged for emergency handling. The next step in call processing is for the INVITE to be evaluated by local policy.

Local Policy

Local policies must be configured to match and then route an incoming emergency call. Once a local policy match is made, the Oracle Communications Session Border Controller looks to the configured policy attributes for where to forward the INVITE. A matching policy attribute's next hop should be configured to point to an emergency LRT that contains specific destinations for emergency calls. In addition, the **elec str lkup** parameter must be set to enabled so the Oracle Communications Session Border Controller will perform an LRT lookup based on the location string returned in the CLF response.

The **eloc str match** parameter identifies the attribute, whose value in the location string will be used as the lookup key in the emergency LRT. For example, if the returned location string is:

```
loc=xxx;noc=yyyy;line-code=zzzz
```

and the **eloc str match** parameter is set to **noc**, then when the Oracle Communications Session Border Controller performs a local policy route search, it will search the LRT for yyyy. If the **eloc str match** parameter left empty or if there is no match when **eloc str lkup** is enabled, the entire location string is used as the lookup key.

Emergency LRT

The Oracle Communications Session Border Controller needs to be configured with an emergency LRT to route emergency calls to their destination.

As stated in the previous section, when searching an emergency LRT, any user defined parameter within a Location String may be used as the key to look up next-hop routing information.

LRT files support <user type = string> which enables the Oracle Communications Session Border Controller to perform searches on free form attributes that may appear in the returned

location-string. The <user type = string> value for an entry in the emergency LRT should be set to a part or whole value returned in the CLF's location string. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<localRoutes>
    <route>
        <user type="string">1234</user>
        <next type="regex">!^.*$!sip:911@192.168.200.140:5060!</next>
    </route>
    <route>
        <user type="string">loc=xxx;noc=yyyy;line-code=zzzz</user>
        <next type="regex">!^.*$!sip:911@192.168.1.139:5060!</next>
    </route>
</localRoutes>
```

> **Note:**
>
> Given that the Location String is not a well-defined string, care should be taken when defining and configuring the LRT tables.

LRTs must be individually uploaded to both the active and standby systems in an HA node; LRTs are not automatically replicated across nodes.

CLF Response Failure

If there is no location string in a CLF's repsone or the CLF rejects the call, the Oracle Communications Session Border Controller uses the **default location string** parameter from the ingress SIP interface to populate the PANI header. The emergency call proceeds normally using this location string's information for emergency LRT lookups.

E-CSCF Configuration

This procedure assumes that the Oracle Communications Session Border Controller is configured to communicate with a CLF. In addition, this procedure assumes an the Oracle Communications Session Border Controller is configured and loaded with an appropriate LRT for E-CSCF Use.

To configure an NMC for E-CSCF use (baseline parameters are not mentioned):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **net-management-control** and press Enter.

   ```
   ORACLE(session-router)# net-management-control
   ```

4. **name**—Enter the name of this network management control rule; this value uniquely identifies the control rule. There is no default for this parameter.

5. **state**—Enable or disable this network management control rule. The default value is **enabled**. The valid values are:
   - enabled | disabled
6. **type—**Set this parameter to **priority** so that the Oracle Communications Session Border Controller will flag incoming calls with a matching destination identifier as a priority calls.
7. **treatment**—Set this parameter to **divert**.
8. **next-hop**—Leave this parameter blank so that the call's processing will go directly to local policy.
9. **destination-identifier**—Enter the service URN that endpoints in your network include in their request URIs to identify themselves as emergency calls.
10. Save your configuration.

    To configure local policy for E-CSCF use (baseline parameters are not mentioned):
11. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```
12. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ORACLE(session-router)#
    ```
13. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

    ```
    ORACLE(session-router)# local-policy
    ORACLE(local-policy)#
    ```
14. **to-address**—Set this parameter to the lookup key for matching emergency calls. You can now use a service URN as lookup key criteria.
15. Save your configuration.

    To configure policy attributes for E-CSCF use (baseline parameters are not mentioned):
16. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

    ```
    ORACLE(local-policy))# policy-attributes
    ORACLE(policy-attributes)#
    ```
17. **next-hop**—Set this parameter to **lrt: name-of-elrt-file.gz** for this policy attribute to lookup routes in the named lrt file.
18. **eloc-str-lkup**—Set this parameter to **enabled** for the Oracle Communications Session Border Controller to parse the emergency location string, as received in a CLF Line Identifyier AVP, for emergency LRT lookup.
19. **eloc-str-match**—Set this parameter to the attribute name found in the location string whose value will be used as a lookup key in the LRT named in the next-hop parameter. Common values include "loc" or noc.
20. Save and activate your configuration.

Maintenance and Troubleshooting

The **show lrt route-entry** command displays two entries, if the username 1234 has a "string" type and "E164" type entries.

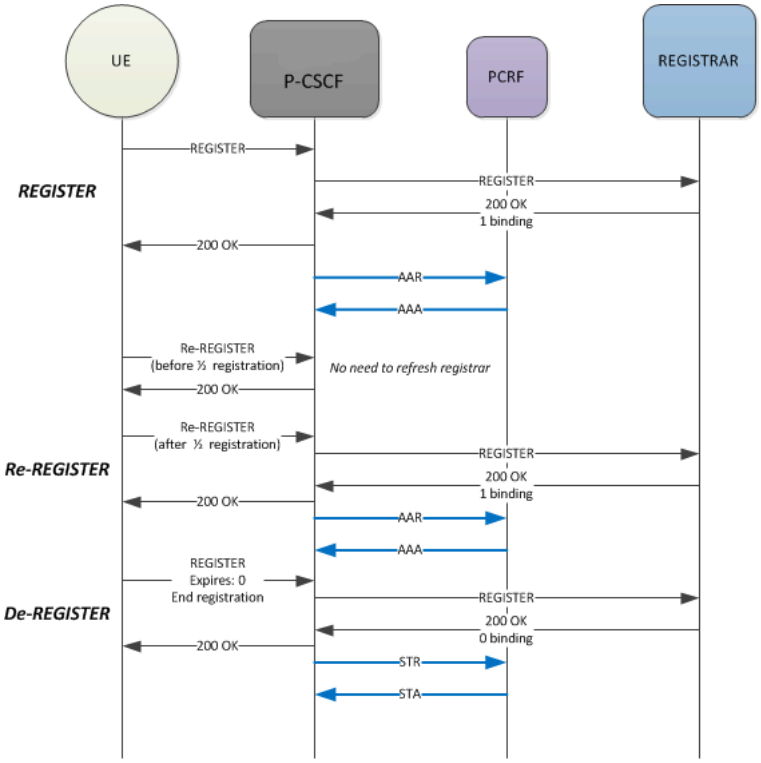
```
ACMEPACKET# show lrt route-entry emergency_lrt 1234
UserName <1234>
 User Type= E164
 NextHop= !^.*$!sip:911@192.168.200.139:5060!
 NextHop Type= regexp
UserName <1234>
 User Type= string
 NextHop= !^.*$!sip:911@192.168.200.140:5060!
 NextHop Type= regexp
```

2774 - Provisioning of SIP Signaling Flow Information

This feature supports of Provisioning of SIP Signaling Flow Information as described in 3GPP TS 29.213 section B1b [1], and the procedures specified in TS 29.214 section 4.4.5a.

The feature applies to the scenario when the Oracle Communications Session Border Controller (A-SBC / P-CSCF) uses its external policy server functionality and connects to a PCRF via Rx interface in bandwidth-management mode.

This feature deals with the information that the Oracle Communications Session Border Controller includes in an AAR message it sends to the PCRF when an endpoint registers, re-registers, and de-registers. The following diagram shows the typical scenario.



Configuration Conflict

The **npli-upon-register** component of an **npli-profile** invalidates the **provision-signaling-flow** configuration when they overlap. Exercise caution when applying a **provision-signaling-flow** to ensure it does not conflict with any **npli-profile** configuration. The SBC provides a configuration verification error when it detects this conflict.

Initial Registration

When an endpoint registers, the Oracle Communications Session Border Controller creates and sends an AAR message to a PCRF which includes the following information:

<AA-Request>::= < Diameter Header: 265, REQ, PXY >

AVP	AVP Contents
Session-ID	OriginHost;0;0;<Key to Registration Cache> Key is usually the endpoint's AoR.
Auth-Application-ID	Application-ID of Rx (16777236)
Origin-Host	<ext-policy-server_name>.<ext-policy-realm>.<domain-name-suffix>
Origin-Realm	<ext-policy-realm>.<domain-name-suffix>
Destination-Realm	<IP Address of endpoint>@<destination realm of this AAR>
Framed-IP-Address AVP - v4 address	Layer 3 Endpoint-IP-address v4 - Framed-IP-Addressor
Framed-IPv6-Prefix AVP - v6 address	v6 - Framed-IPv6-Prefix
Media-Component	Grouped AVP description follows

Media-Component-Description AVP ::= < AVP Header: 517 >

AVP	AVP Contents
Media-Component-Number	0
Media-Sub-Component	Grouped AVP description follows

Media-Sub-Component ::= < AVP Header: 519 >

AVP	AVP Contents
Flow-Number	1
Flow-Description	Permit in <ip> from <Endpoint IP:Port> to <SBC Sip Interface IP:Port> Permit out <ip> from <SBC SIP Interface IP:Port> to <Endpoint IP Port> Where <ip> is (UDP: 17, TCP: 6) if wildcard-trans-protocol = disabled.
Flow-Status	Set to: ENABLED (2)
Flow-Usage	Set to: AF_SIGNALLING (2)
AF-Signalling-Protocol	Set to: SIP (1)

Register Refresh

When a registration Refresh is received before the half time of the registration expiry, the registration cache is not updated and the Oracle Communications Session Border Controller responds with 200OK to the UE. This is standard registration cache functionality. No additional AAR is sent to the PCRF.

If a registration Refresh is received after the half time of the registration expiry, or if any registration information changes, the Oracle Communications Session Border Controller sends an AAR to the PCRF after it receives and forwards a 200 OK response from registrar. The AAR includes the same session-id as the initial AAR that was to PCRF. This lets PCRF correlate the AAR with the earlier AAR that it received.

De-Registration

When an contact de-registers with expires=0, it means that the endpoint is removing all of its contacts from registration. When this occurs, the Oracle Communications Session Border Controller sends an STR message to the PCRF to terminate the session. If the de-registration message does not reflect a complete de-registration, the Oracle Communications Session Border Controller does not send the STR message to the PCRF. The STR message includes the following information:

<ST-Request> ::= < Diameter Header: 275, REQ, PXY >

AVP	AVP Contents
Session-ID	OriginHost;0;0;<Key to Registration Cache> Key is usually the endpoint's AoR.
Auth-Application-ID	Application-ID of Rx (16777236)
Origin-Host	<ext-policy-server_name>.<ext-policy-realm>.<domain-name-suffix>
Origin-Realm	<ext-policy-realm>.<domain-name-suffix>
Destination-Realm	<IP Address of endpoint>@<destination realm of this AAR>
Destination-Host	N/A
Termination-Cause	DIAMETER LOGOUT (1) - User initiated a disconnect

Failure Response to Re-Register

Upon reception of a failure response from the REGISTRAR for a subsequent Registration refresh from endpoint, the Oracle Communications Session Border Controller performs de-registration actions, i.e. an STR message to the PCRF.

Provisioning SIP Signaling Flows Configuration

To enable correct provisioning of signaling flows upon registration:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **ext-policy-server** and press Enter.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI select command) the external policy server that you want to edit.

4. **provision-signaling-flow**—Set this to enabled for the Oracle Communications Session Border Controller to send AAR messages to a PCRF on endpoint registrations.
5. Save and activate your work.

Troubleshooting

Upon receipt of AAA response from PCRF for the AAR sent, the Oracle Communications Session Border Controller logs the status of the AAA received. Successful AAAs are noted by session-ID in EBMD logs. The log will look like:

```
Provision of SIP Signaling Flow (<session-ID>) via Diameter Rx on realm
<realm-id> successful.
```

Failed AAA response are noted at NOTICE level EBMD logs citing the failure response to AAR request with the session-ID sent. An a MINOR non-health-affecting alarm is also raised with the following text:

```
Provision of SIP Signaling Flow (<session-ID>) via Diameter Rx on realm
<realm-id> failed.
```

show ext-band-mgr

The show ext-band-mgr command has been augmented to include the Register and DeRegister counts. For example:

```
ORACLE# show ext-band-mgr
13:55:49-166
EBM Status                  -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          7       4       1
Server Trans           0       0       0          0       0       0
Sockets                1       1       0          1       1       1
Connections            0       0       0          1       1       1
                          ---- Lifetime ----
                     Recent      Total  PerMax
Reserve                   0          0       0
Modify                    0          0       0
Commit                    0          4       2
Remove                    0          2       1
Register                  1          1       1
DeRegister                1          1       1
EBM Requests              0          6       3
EBM Installs              0          6       3
EBM Req. Errors           0          0       0
EBM Rejects               0          0       0
EBM Expires               0          0       0
EBMD Errors               0          0       0
```

Subscription for Notification of Signaling Path Status

The Oracle Communications Session Border Controllercan explicitly open a flow for the signaling and through a subscription to this flow, provide status change notifications.

The Oracle Communications Session Border Controller supports provisioning of signaling flows to send a Authentication-Authorization Request (AAR) message to the Policy and Charging Rule Function (PCRF) when an endipoint registers, reregisters and deregisters. Service Providers want status information of the signaling flows to be able to react to state changes. By enabling the **specific-action-sig-flow-subscription** parameter, the Oracle Communications Session Border Controller subscribes for signaling flow status change notifications. The Policy and Charging Rule Function (PCRF) informs the Proxy Call Session Control Function (P-CSCF) when signaling flow state changes have taken place so that the P-CSCF can then take action, e.g. de-registering the User Equipment in the Serving Call Session Control Function (S-CSCF).

The Oracle Communications Session Border ControllerP-CSCF supports the Specific Action attribute value pair (AVP) in conjunction with the AAR being sent to provision a signaling flow. The request to subscribe to this flow status change notifications will enable the handling of either INDICATION_OF_LOSS_OF_BEARER and/or INDICATION_OF_RELEASE_OF_BEARER, the two values of the specific action AVP.

Once enabled, the Oracle Communications Session Border Controller P-CSCF accepts the Re-Authentication Request (RAR) from the PCRF and its notification. The Oracle Communications Session Border Controller P-CSCF replies with a Re-Authentication Authorization acknowledging the RAR as per usual.

When the notification in the RAR is INDICATION_OF_RELEASE_OF_BEARER, the P-CSCF cancels the registration of that UE. The Oracle Communications Session Border Controller deletes the contact on INDICATION_OF_RELEASE_OF_BEARER only if this action is specified in the **specific-action-sig-flow-subscription** parameter settings. Further action is not taken when the RAR notification is "INDICATION_OF_LOSS_OF_BEARER" as the bearer can be re-established. This notification service is not available for VoLTE.

Subscription for Notification of Signaling Path Status Configuration

To enable **specific-action-sig-flow-subscription** to receive notifications regarding signal state change information.

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. **specific-action-sig-flow-subscription**—Signaling path status changes information available for notification
    - **loss of bearer**— Within a Re-authorization Request (RAR), this value shall be used when the server reports a loss of a bearer (e.g. in the case of General Packet Radio Service - Packet Data Protocol (GPRS PDP) context bandwidth modification to 0 kbit) to the Application Function (AF). The Service Data Flows (SDFs) that are deactivated as a consequence of this loss of bearer shall be provided within the Flows Attribute Value Pair (AVP). In the Authentication-Authorixation Request (AAR), this value indicates that the AF requests the server to provide a notification at the loss of a bearer.
    - **release of bearer**— Within a RAR, this value shall be used when the server reports the release of a bearer (e.g. PDP context removal for GPRS) to the AF. The SDFs that are deactivated as a consequence of this release of bearer shall be provided within the Flows AVP. In the AAR, this value indicates that the AF requests the server to provide a notification at the removal of a bearer.tate where the bearer connection has been released.
4. Type **done** to save your configuration.

RTP and RTCP Bandwidth Calculation and Reporting

The Oracle Communications Session Border Controller supports changing bandwidth requirements in an ad-hoc mult-party conference by tracking reduced bandwidth needs as parties are placed on hold during the initiation of a multi-party call. The combination of the 5 AVPs are considered by network elements for this functionality.

This section is applicable to the Oracle Communications Session Border Controller's Rx implementation when acting as a P-CSCF and connecting with a PCRF. The 5 AVPs considered in this section are created and sent in AAR messages.

Max-Requested-Bandwidth-UL & Max-Requested-Bandwidth-DL AVPs

These AVPs reflect RTP bandwidth requirements for a call. The default behavior (i.e., no options are configured) that dictates how these AVPs are created follows.

The Oracle Communications Session Border Controller reads b=AS parameter from the media line attributes in the SDP and not from the session level attributes. The average rate limit scenario is when no b=AS: parameter in the SDP and the **media-profile**, **average-rate-limit** parameter is configured to a value greater than 0. The Oracle Communications Session Border Controller inserts the **media-profile**, **average-rate-limit** parameter ×8 into both Max-Requested-Bandwidth-UL & Max-Requested-Bandwidth-DL AVPs.

The SDP bandwidth scenario is when there is a b=AS: parameter in the SDP and the media-profile configuration element has the following configurations:

- average-rate-limit = 0
- sdp-rate-limit-headroom > 0
- sdp-bandwidth = ENABLED

When these conditions are met, the Max-Requested-Bandwidth-UL & Max-Requested-Bandwidth-DL AVPs are populated with the value in the b=AS: parameter + **sdp-rate-limit-headroom** ACLI parameter.

Optional AVP Creation

When the **get-bw-from-sdp** option is configured in the sip-config, the following occurs:

When SDP contains the

```
b=AS:
```

parameter with valid value, the Oracle Communications Session Border Controller multiplies that value ×1000 and inserts the result in the Max-Requested-Bandwidth-UL and Max-Requested-Bandwidth-DL AVPs in an AAR message.

If there is no b=AS: line in the SDP, the value is taken from the **media-profile**, **average-rate-limit** parameter multiplied ×8 to get a bps value then inserted into the Max-Requested-Bandwidth-UL and Max-Requested-Bandwidth-DL AVPs in an AAR message.

The Oracle Communications Session Border Controller uses the b=AS: line (or chosen codec via the media-profile) from the final answer SDP for the session as the value for creation of these two AVPs.

RR-Bandwidth & RS-Bandwidth AVPs

These AVPs reflect RTCP bandwidth requirements for a call. When SDP contains:

```
b=RR:
b=RS:
```

parameters and values, the Oracle Communications Session Border Controller inserts the values (after the :) into the respective RR-Bandwidth (521) and RS-Bandwith (522) AVPs. When these parameters are not present in the SDP, the RR-Bandwidth and RS-Bandwidth AVPs are not created in AAR messages sent to the PCRF. The **sip-config**, **get-bw-from-sdp** option must be configured to enable creation of these AVPs.

Flow-status AVP

The Flow-status AVP (511) is based on SDP direction. Tests for SDP are performed in the following order:

SDP State	Flow-Status AVP (511) Set to
port in the SDP m-line is 0	REMOVED (4)
Transport in m-line is "TCP" or " TCP/MSRP"	ENABLED (2)
a=recvonly and <SDP direction> = UE originated	ENABLED-DOWNLINK (1)
a=recvonly and <SDP direction> = UE terminated	ENABLED-UPLINK (0)
a=sendonly and <SDP direction> = UE originated	ENABLED-UPLINK (0)
a=sendonly and <SDP direction> = UE terminated	ENABLED-DOWNLINK (1)
a=inactive	DISABLED (3)
a=sendrecv or no direction attribute	ENABLED (2)

UE originated - Call originator creates the SDP being considered.

UE terminated - Call terminator creates the SDP being considered.

Flow-status AVP is always created according and requires no configuration.

The P-Early-Media Header and the Rx Interface

As defined in 3GPP TS 24.229, IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 both the P-CSCF and IBCF may add, remove, or modify, the P-Early-Media header field within forwarded SIP requests and responses according to procedures in RFC 5009.

The P-CSCF can use the P-Early-Media header field for the gate control procedures, as described in 3GPP TS 29.214. In the current implementation, if you set the configuration option **early-media-allow** to **none**, the SBC sends the Flow-Status AVP (511) in any AAR request set to disable until there is a final response.

Although you can configure the SBC to provide PEM header information within the Flow-Status AVP (511) to provide early media status over the Rx interface, there are two conditions that must be in place before it can populate this AVP with PEM header flags.

The SBC does not include PEM header feature status in the Rx Flow-Status AVP unless:

1. You enable the **get-flow-status-from-sdp** option in the **sip-config**.

   ```
   ORACLEACMEPACKET#(media-manager) options +get-flow-status-from-sdp
   ```

2. There has been an SDP exchange. Note that provisional responses without SDP, which occur prior any SDP exchange, do not generate update data within the Rx flow-status AVP even if they include PEM status.

2629 - IR.92 Compliance via SIP 380 Response

This feature furthers the Oracle Communications Session Border Controller's compliance with GSMA's Voice over LTE specification (IR.92) to redirect VoLTE originated emergency calls to a circuit switched network.

When the Oracle Communications Session Border Controller receives an emergency call, which it can not complete, it returns a 380 (Alternative Service) response to the sender. Some examples of when the Oracle Communications Session Border Controller can not forward such an emergency call are:

- The Next-hop does not exist to route the call via NMC means
- The NSEP calls are rejected due to the Oracle Communications Session Border Controller hitting a load limit
- The NSEP calls are rejected due to the target session agent exceeding constraints
- The NMC treatment for the call is set to Reject

380 Response Format

The Oracle Communications Session Border Controller's 380 SIP response to the sender includes:

- Content-Type header field set to application/3gpp-ims+xml
- P-Asserted-Identity header field set to the value of the SIP URI of the last entry on the Path header field value received during registration; It is the value of the SIP URI of the P-CSCF.

- 3GPP IM CN subsystem XML body containing an <ims-3gpp> element with the "version" attribute will be set to "1" and with an <alternative-service> child element set to "alternative service"
  - a <type> child element set to emergency
  - a <reason> child element set to the value of configuration element send-380-response
  - an <action> child element set to emergency-registration

380 Response Example

```
SIP/2.0 380 Alternative Service
Via: SIP/2.0/UDP 192.168.15.2:5060;branch=z9hG4bK-23615-1-0
From: sipp <sip:911@192.168.15.2:5060>;tag=1
To: sut <sip:911@192.168.101.11:5060>
Call-ID: 1-23615@192.168.15.2
CSeq: 1 INVITE
Content-Type: application/3gpp-ims+xml
Content-Length: 209
P-Asserted-Identity: sip:911-44e2etbufgibf@172.16.101.11:5060
Reason: Q.850; cause=63
<?xml version='1.0' encoding="UTF-8"?>
<ims-3gpp version="1.0">
<alternative-service>
<type>emergency</type>
<action>emergency-registration</action>
<reason>sample reason</reason>
</alternative-service>
</ims-3gpp>
```

IR.92 Compliance Configuration

To configure the 380 SIP response for IR.92 compliance:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. If configuring an existing interface, enter the select command to select the interface.
5. **send-380-response**—Enter a reason phrase enclosed in quotes to place in the reason tag of the <ims-3gpp> element in a 380 response for a failed-to-route emergency call.

6. Type **done** and continue.

IR.92 Multiple Emergency Numbers

The Oracle Communications Session Border Controller expands compliance with the IR.92 standard by including an alternative service and message for emergency traffic.

The Oracle Communications Session Border Controller can be configured with multiple emergency services based on the dialed number match. Custom messages can be configured through the **sip-380-reason** parameter.

When an emergency call is received at the Oracle Communications Session Border Controller Proxy Call Session Control Function, the system checks any Network Management Control (NMC) rules that are enabled for the ingress realm. If the call matches a rule's **destination-identifier** parameter and fails with a 380 response, then that rule's **sip-380-reason** will be used in the response. Conversely, if the matching rule's **sip-380-reason** is empty or no matching NMC rule is found, then the value of the **sip-interface**'s **send-380-response** will be used as the reason. If the **send-380-response** is also empty, the default reason "priority calls not allowed" will be used. This logic also applies if the call fails and no matching network management control rule is found. The Oracle Communications Session Border Controller will use the value of the **sip-interface**'s **send-380-response** as the reason or the default message if the value is empty. The valid values for **send-380-response** are any string. It is not required.

This functionality is configured through the **sip-380-reason** parameter in the **net-management-ctrl** object.

IR.92 Multiple Emergency Numbers Configuration

To configure multiple emergency numbers and custom messages.

1. Access the **net-management-control** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# net-management-control
ORACLE(net-management-control)
```

2. Select the **net-management-control** object to edit.

```
ORACLE(net-management-control)# select
<name>:
1: NMC01 (type=priority)
2: NMC02 (type=priority)

selection: 1
ORACLE(net-management-control)#
```

3. **sip-380-reason** — Enter a reason phrase enclosed in quotes.

4. Type **done** to save your configuration.

Emergency Calls from Unregistered Users

Version S-CZ7.2.0 provides compliance with Section 5.1.6.2 of 3GPP TS24.229. This section mandates that unregistered, or roaming User Endpoints (UE) must perform an emergency

registration before sending any SIP request (for example, an INVITE) related to the emergency situation.

Version S-CZ7.2.0 adds a new parameter (options reject-unreg-priority-calls) that enforces compliance with Section 5.1.6.2, Initial Emergency Registration, of 3GPP TS24.229, IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP);Stage 3. Section 5.1.6.2 mandates that unregistered, or roaming User Endpoints (UE) must perform an emergency registration before sending any SIP request (for example, an INVITE) related to the emergency situation.

Prior to Version S-CZ7.2.0, the default SBC behavior (acting as a P-CSCF) was not compliant with Section 5.1.6.2 of 3GPP 24.229. Rather, an emergency call from an unregistered or roaming UE was processed and routed in exactly the same way as a similar call from a currently registered UE.

Processing of emergency calls received from an unregistered or roaming UE as specified by 3GPP TS24.229 can be summarized as follows.

First, the emergency call (for example an INVITE) is rejected by the SBC with a 380, Alternative Service response message, which includes

1. a Content-Type header field set to

   ```
   application/3gpp-ims+xml
   ```

2. a P-Asserted-Identity header set to the value of the SIP URI of the last entry on the path header field value received during registration -- the SIP URI of the P-CSCF. This is the UE-side Service Route that the SBC sends out when sip-ims-feature is enabled.
3. an Alternative Service child element containing a Type element set to "emergency" ;

   ```
   emergency-value = %x65.6D.65.72.67.65.6E.63.79 ; "emergency"
   ```

   an Action element set to “emergency-registration”

   ```
   %x65.6D.65.72.67.65.6E.63.79.2D.72.65.67.69.73.74.72.61.74.69.6F.6E ;
   "emergency-registration"
   ```

   a Reason element set to

   ```
   "priority calls not allowed from unregistered UEs"
   ```

Second, upon receipt of the 380, Alternative Response, the UE responds with an emergency registration.

Finally, upon receipt of the emergency registration, the SBC processes that registration as it would any other registration request.

Emergency Calls from Unregistered Users Configuration

1. Access the **sip-interface** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **options +reject-unreg-priority-calls**—Add this option to require compliance with Section 5.1.6.2 of 3GPP TS24.229 . If this option is enabled, emergency calls from unregistered or roaming UEs are rejected with a 380, Alternative Service response, and the UE is required to perform an emergency registration prior to sending an emergency message. If this option is not present, emergency calls, by default, are processed and routed regardless of registration status.

```
ORACLE(sip-interface)# options +reject-unreg-priority-calls
```

4. **disallow-priority-calls**—ensure that this option has not been enabled on the current SIP interface because it summarily rejects all emergency calls on the current SIP interface. Ensure that this option is NOT present to ensure compliance with 3GPP TS24.229.

```
ORACLE(sip-interface)# options -disallow-priority-calls
```

5. **options send-380-response**—ensure that this option has not been enabled on the current SIP interface. The **send-380-response** option provides the text for the Reason header. The correct text is supplied by the 380 Alternative Service message. Ensure that this option is NOT present to ensure compliance with 3GPP TS24.229.

```
ORACLE(sip-interface)# options -send-380-response
```

6. Type **done** to save your configuration.

IR.94 Support

The Oracle Communications Session Border Controller supports IR.94 (IMS Profile for Conversational Video Service) for use in an IMS and LTE environment.

The Oracle Communications Session Border Controller supports both video and audio and can keep a call up by switching to audio mode when the video session is dropped. The Oracle Communications Session Border Controller receives a Specific-Action AVP in the RAR from the PCRF indicating the loss of video bearer. The Oracle Communications Session Border Controller then replies with a RAA, and finally sends a STR failing the call.

This behavior contradicts TS 29.214 that indicates the P-CSCF should wait for an ASR from the PCRF (indicating the bearer has been removed), and should not fail the call until all bearers have been removed.

The following behaviors enable IR.94 Support:

- The Oracle Communications Session Border Controller supports the IR.94 re-INVITE procedures for adding or removing video mid-call.
- The Oracle Communications Session Border Controller recognizes that a UE is capable of handling video calls when in the REGISTER message, the sip.video media feature is present in the Contact: header.

- The Oracle Communications Session Border Controller supports Audio-Video Profile with Feedback (AVPF) and SDP Capability Negotiation (RFC 5939) as documented in TS 26.114 section 6.2.1a.
- When the Oracle Communications Session Border Controller supports sessions (200 OK received after an INVITE) with a single media stream throughout the session duration, it may receive any of the following notifications in the Specific-Action AVP of the RAR from the PCRF:
  - INDICATION_OF_LOSS_OF_BEARER (2)
  - INDICATION_OF_RELEASE_OF_BEARER (4)
  - INDICATION_OF_OUT_OF_CREDIT (7)
  - INDICATION_OF_FAILED_RESOURCES_ALLOCATION (9)

  The Oracle Communications Session Border Controller then:

  1. Replies with an RAA acknowledging the RAR
  2. Sends a BYE to both the UE and the Core, then
  3. Sends an STR to the PCRF indicating the session has been terminated.
- When the Oracle Communications Session Border Controller supports sessions (200 OK received after an INVITE) with multiple simultaneous media streams sometime during the session, it may receive any of the following notifications in the Specific-Action AVP of the RAR from the PCRF:
  - INDICATION_OF_LOSS_OF_BEARER (2)
  - INDICATION_OF_RELEASE_OF_BEARER (4)
  - INDICATION_OF_OUT_OF_CREDIT (7)
  - INDICATION_OF_FAILED_RESOURCES_ALLOCATION (9)

  The Oracle Communications Session Border Controller then:

  1. Sends an RAA acknowledging the RAR, and no further action is required When an eSR-VCC handover occurs, the ATCF (Oracle Communications Session Border Controller) sends a INVITE/UPDATE without video (voice-only SDP) toward the target UE.

IR.94 Loss Of Voice Bearer

The Oracle Communications Session Border Controller provides compliance with an IMS Profile for Conversational Video Service (IR.94) requirement that specifies the termination of a multi-media session (voice and video) if the voice bearer is lost.

Version S-CZ7.2.0 adds a new parameter (options **terminate-on-voice-bearer-release**) that addresses an unlikely, but possible, corner case in which the voice bearer is lost but the video bearer remains in service. Prior to Version S-CZ7.2.0, the Oracle Communications Session Border Controller would retain the call as a video-only session. This behavior is not compliant with IR.94, which specifies that the video-only session should not be allowed to continue.

The option **terminate-on-voice-bearer-release** enables compliance with the IR.94 standard. Compliance is ensured by basing the decision to terminate or continue a multi-media session on the state of the voice bearer exclusively; the state of any other media bearer (video, for example) plays no role in the decision process. Consequently, if the voice bearer fails, the call terminates; failure of the video bearer, or other media stream, is not pertinent to call termination or continuance.

IR.94 Loss Of Voice Bearer Configuration

Use the following procedure to enable IR.94 compliance.

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. Use options terminate-on-voice-bearer-release to enable I.94 compliance.

```
ACMEPACKET(ext-policy-server)# options +terminate-on-voice-bearer-release+
    ACMEPACKET(ext-policy-server)#
```

4. Type **done** to save your configuration.

Dynamic Sessions Agents for Home-Remote S-CSCF Liveliness

For IMS applications, you can configure your Oracle Communications Session Border Controller to create session agents dynamically for remote S-CSCFs on in-coming service routes. Dynamic session agents inherit properties of the static session agents with which they are associated, and the Oracle Communications Session Border Controller takes them out of service when they are deemed no longer responsive according to the liveliness mechanism you set.

Discovery

The Oracle Communications Session Border Controller, acting as a P-CSCF, can discover remote S-CSCFs using Service-Route header that returns with a 200 OK response from the registrar for a REGISTER request from the endpoint. The system takes the top Service Route from the response and uses it as the first hop in the egress route for the endpoint. Because the creation of dynamic session agents is based on the Service Route returned in the 200 OK, there is no impact to handling AoRs with multiple contacts.

In addition, the system stores the Service Route header data with the endpoint's registration cache. If the Service Route is an FQDN, a DNS look-up is used to provide the route to the S-CSCF.

Creation

For your Oracle Communications Session Border Controller to create session agents dynamically, you must enable the **create-dynamic-sa** in the global SIP configuration. This parameter defaults to **disabled**, meaning that no dynamic session agents will be created.

When you set the parameter to **enabled**, the Oracle Communications Session Border Controller decides whether or not to create dynamic session agents in the following ways. The system will create up to five (5) dynamic session agents.

- If the Service Route is an IP address, the Oracle Communications Session Border Controller attempts to find an exact match for that IP address against pre-existing session agents. If no exact match appears, the system will not create dynamic session agents.
- If the Service Route is an FQDN and matches an existing session agent, the Oracle Communications Session Border Controller assigns the remote S-CSCF to that session agent.
  Service Routes with FQDNs might not match any session agents. However, the Service Route's FQDN can match the DNS suffix of a wildcard session agent. When this wildcard matching occurs, the Oracle Communications Session Border Controller creates a new dynamic session agent with the original wildcard session agent as its parent. The new dynamic session agent inherits all configuration properties of the parent session agent.
- If the Service Route is neither an IP address nor an FQDN and the system is unable to match any statically defined or wildcard session agents, then the Service Route is not associated with any session agents.

Property Inheritance

Because dynamic session agents are created on the basis of static or wildcarded session agents, you can think of them as children of the original--or parent--session agent. This relationship means that the child, dynamic session agent inherits the configuration properties of its parent, static or wildcard session agent.

These inherited configuration properties and their effects are:

- The FQDN's DNS resolution of the new dynamic session agent to IP addresses.
- Monitoring for liveliness via your configuration settings for the ping method and interval, transaction timeout, or OOS response. The dynamic session agent also inherits its parent's criteria for being taken out of service, with the exception that a dynamic session agent will not be taken out of service until it expires on its own. This gives the dynamic session agent time to return to service.
- The setting for **ping-all-address**, which when enabled causes new routes (internal session agents) fork and makes the dynamic session agent the parent. The system caps the limit of five routes (or internal session agents) per dynamic session agent. If the FQDN resolves to more than five IP addresses, the system only uses the first five to create routes (internal session agents), and then pings the internal session agents.
- Suppression of the heartbeat (or ping) of the IP address in the presence of traffic. If traffic to a specific IP address stops, then the Oracle Communications Session Border Controller resumes pinging within the time you set for the **ping-interval**.
- The setting for the **invalidate-registration** option. If the dynamic child session agent goes out of service, the corresponding registration cache entries of users that have the Service Route pointing to this session agent will be invalidated.
  This invalidation means that the next REGISTER request from that user will not receive a local response. Any other services this user requires will not use the Service Route information stored in its registration cache. Instead, the system will route it to the next hop as determined by other means, such as the local policy. At this time, the user would must be re-registered by the registrar, a process that might return a new Service Route to be updated in the registration cache.

Deletion

The Oracle Communications Session Border Controller needs to delete dynamic session agents no longer in use. But dynamic session agents should not be deleted too soon, in case they return to service. So, deletion occurs according to the process this section describes.

For each dynamically created session agent, the Oracle Communications Session Border Controller assigns and tracks the last registration expiry time. It determines this time by doubling the registered endpoint's core side expiry. Where X is the last registration expiry value and Y is the registered endpoint's core side expiry, the Oracle Communications Session Border Controller performs checks according to this criteria:

X > (2 * Y)

The the system updates the expiry time with the greater value. This way, whenever the dynamic session agent re-registers, it will have an updated expiry timer value.

Timeouts can also cause dynamic session agents to be deleted. Pings, status changes, transaction timeouts, DNS expiries and other system occurrences can trigger timeouts. When the Oracle Communications Session Border Controller detects that a timeout has occurred, the dynamic session agent is deleted.

How to Wildcard a Session Agent

You can create a wildcard session agent using the session agent's hostname parameter when configured with an FQDN.

To configure a wildcard session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are wildcarding and existing session agent, you have to select the configuration before you make changes.

4. **hostname**—To wildcard a session agent, you simply replace the value you want to wildcard with an asterisk (*****). Also note that your value must lead with the asterisk (*****), as in the following example.

   ```
   ORACLE(session-router)# hostname *xyz.com
   ORACLE(session-agent)#
   ```

5. Save and activate your configuration.

Enabling the Global SIP Configuration for Dynamic Session Agents

To use dynamic session agents for remote S-CSCFs on in-coming service routes, you need to set the create-dynamic-sa parameter in the global SIP configuration to enabled.

To configure the ping mode for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding this support to a pre-existing SIP configuration, you have to select the configuration before you make changes.

4. **create-dynamic-sa**—To support the creation of dynamic session agents for remote S-CSCFs on in-coming service routes, change this parameter from **disabled** (default) to **enabled**.
5. Save and activate your configuration.

SRVCC Handover Support in Alerting Phase

The SBC supports handovers between Packet-Switched (PS) and Circuit-Switched(CS) networks for calls in an alerting phase; that is, a 180 ringing response for the initial INVITE has been sent or received and the SIP final response has not been sent or received.

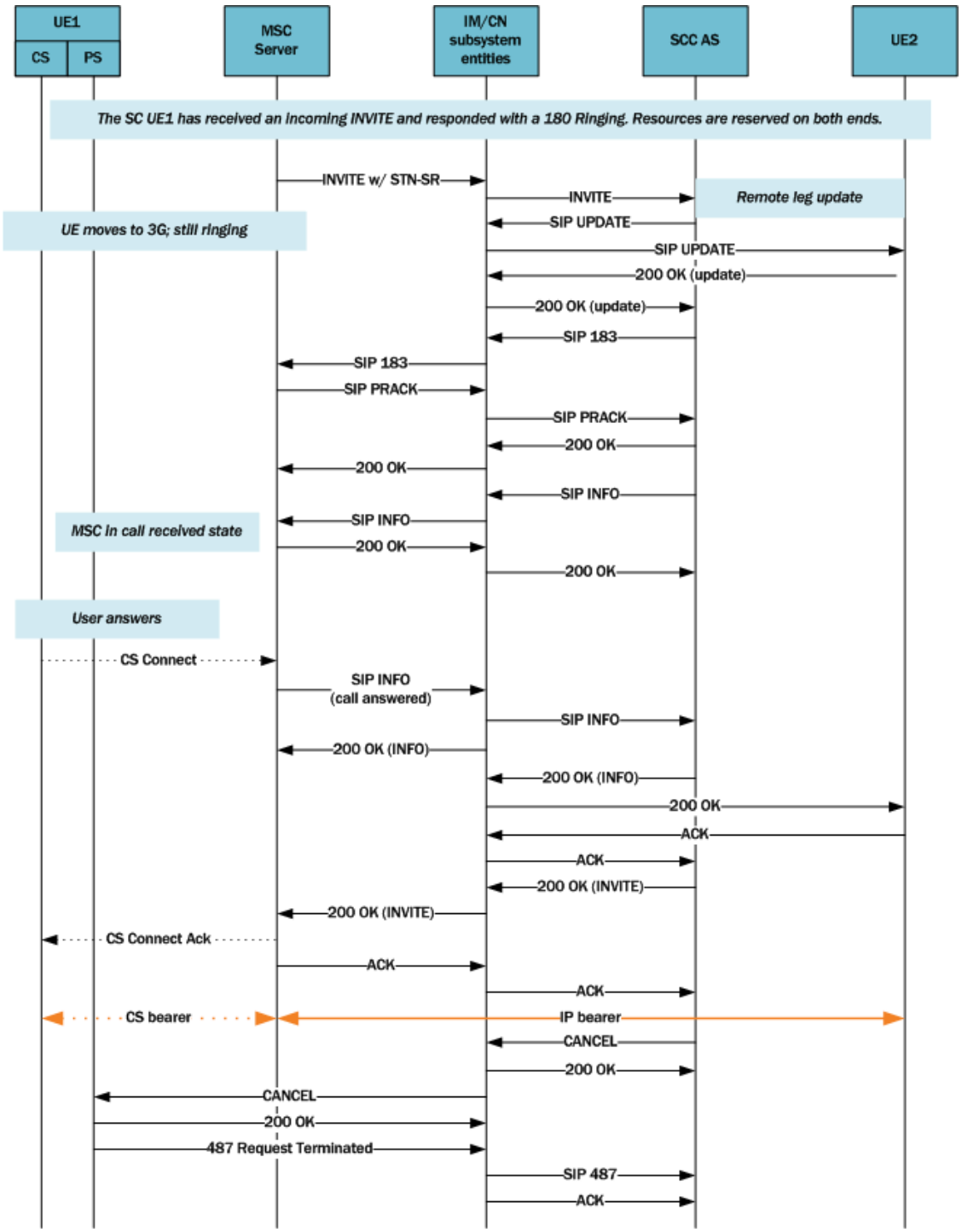
The behavior of the two anchoring points, ATCF and ATGW, is defined by 3GPP in Release 12 of Technical Specification TS 24.237. Oracle Communications developed these functional entities based on the initial version of TS 24.237 Release 10, and has added the **sip-feature-caps** configuration element to align with Release 12.

To ensure that calls in an alerting phase are transferred between PS and CS networks, set the **sip-feature-caps** value as follows:

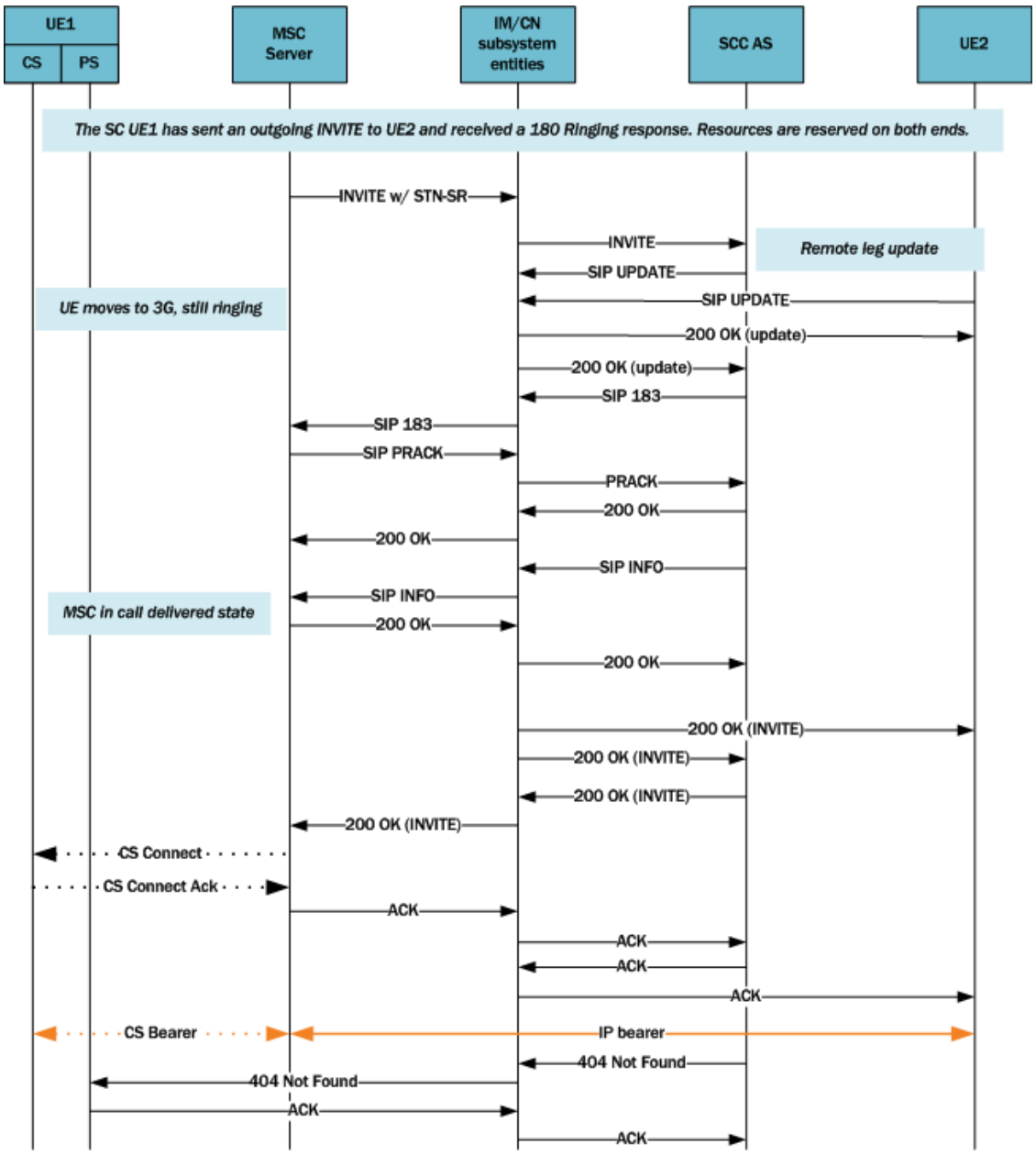
- Set **state** to “enabled”
- Set **atcf-management-uri** to "management" or "psi"
- Set **atcf-alerting** to "enabled"

For information on the results of these settings, refer to the *SIP Feature Capabilities* section.

The diagram below shows an incoming call session during the alerting phase.



This diagram shows an outgoing call session during the alerting phase.



SIP Feature Capabilities

The ATCF can announce a feature capability in a message by transporting the information in the Feature-Caps header, which is supported by the SBC.

The behavior of the two anchoring points, ATCF and ATGW, is defined by 3GPP in Release 12 of Technical Specification TS 24.237. Oracle Communications developed these functional entities based on the initial version of TS 24.237 Release 10, and has added the **sip-feature-caps** configuration element to align with Release 12.

When the **state** is enabled, the SBC includes the Feature-Caps header in applicable SIP messages. The information in the header is dependent on **sip-feature-caps** configuration.

Specifically, When you enable **sip-feature-caps**, regardless of other settings, the SBC inserts the Feature-Caps header with:

- The **g.3gpp.mid-call** capability indicator.
- The **g.3gpp.atcf-path** parameter with a value set to the ATCF URI for terminating requests

Assume the **state** is **enabled** and both **atcf-alerting pre-atcf-alerting** are **disabled**. The SBC adds the Feature-Caps header with the above and:

- The **g.3gpp.atcf** parameter with a value of the configured **atcf-stn-sr** in **sip-config**
- The **g.3gpp.atcf-mgmt-uri** parameter with a value of the configured **atcf-psi-dn** in **sip-config**

Finally, when **state** is **enabled**, **atcf-management-uri** is not disabled, and **atcf-alerting** is **enabled**, the SBC adds the Feature-Caps header with the above and also adds the **g.3gpp.srvcc-alerting** capability indicator. Similarly, the SBC adds the **g.3gpp.srvcc-pre-alerting** capability indicator when **atcf-pre-alerting** is **enabled**.

SIP Feature Capabilities Configuration

You can configure Oracle Communications Session Border Controllers to have the ATCF announce a feature capability in a message by transporting the information in the Feature-Caps header.

1. Access the **session-router** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-feature-caps
ORACLE(sip-feature-caps)#
```

2. **state** — identifies whether to enable the feature and add the Feature-Caps header to messages. Possible values are "enabled" and "disabled". The default value is "disabled".
3. **atcf-alerting** — identifies whether to turn on the alerting feature and add the alerting Feature-Caps header to messages. Possible values are "enabled" and "disabled". The default value is "disabled".
4. **atcf-pre-alerting** — identifies whether to turn on the alerting feature and add the pre-alerting Feature-Caps header to messages. Possible values are "enabled" and "disabled". The default value is "disabled".
5. **atcf-management-uri** — identifies the feature capability indicator that will be used to transport the ATCF management URI. Possible values are "management" and "psi". The default value is "management". When the value is "management" and the value of **state** is "enabled", the Feature-Caps header “g.3gpp.atcf-mgmt-uri” is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element. When the value is "psi" and the value of **state** is "enabled", the Feature-Caps header “g.3gpp.atcf-psi” is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element.
6. Type **done** to save your configuration.

ATCF INVITE ICSI Matching

The Oracle Communications Session Border Controller can check, on reception of an INVITE on an ingress sip-interface that has a configured ATCF and before applying any of the already implemented logic, whether the incoming INVITE includes the ICSI (Instantaneous Channel-State Information) of the requested service. The ATCF will be involved in the call flow when the configured ICSI value matches the ICSI value in the original INVITE; otherwise the handoff call will be rejected with code 480 (Temporarily Unavailable) or 404 (Not Found).

The system looks for the ICSI string in the following headers:

- P-Preferred-Service
- P-Asserted-Service
- Feature-Caps (within the "g.3gpp.icsi-ref" feature-capability indicator)
- Accept-Contact (within the tag-value within the g.3gpp.icsi-ref media feature tag)

An example of the ICSI string in the P-Preferred-Service or P-Asserted-Service header is "urn:urn-7:3gpp-service.ims.icsi.mmtel". Examples of the ICSI string in the Feature-Caps or Accept-Contact headers are "+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel" and "g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel". The ATCF will be involved in the call flow only when the ICSI matches. Configure the **atcf-icsi-match** parameter with the ICSI string you want to match. If **atcf-icsi-match** is blank, the check is not done and the behavior remains the same as before.

ATCF INVITE ICSI Matching Configuration

You can configure the Oracle Communications Session Border Controller to check, on reception of an INVITE on an ingress sip-interface that has a configured ATCF and before applying any of the already implemented logic, whether the incoming INVITE includes the ICSI (Instantaneous Channel-State Information) of the requested service and, if so, to involve the ATCF in the call flow.

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **atcf-icsi-match** — enter the ICSI string you want to match.
4. Type **done** to save your configuration.

Enhanced eSRVCC Call Continuity

As the LTE Evolved Packet Core (EPC) continues to expand, voice deployments (VoLTE) appear more and more, with the Oracle SBC and P-CSCF playing key roles. 3GPP standards that define how LTE communications take place also continue to evolve, identifying and solving critical issues to increase effectiveness and efficiency. One such issue is session continuity, keeping session transfers between LTE and existing 2G and 3G networks as seamless as possible. Single Radio Voice Call Continuity (eSRVCC) offers one solution for the session continuity issue.

In its role as the P-CSCF and IMS-GW, the Oracle Communications Session Border Controller can provide eSRVCC by acting as signaling and media anchor points to handover calls to 3G

networks smoothly. These anchoring points are called Access Transfer Control Function (ATCF) and the Access Transfer Gateway (ATGW), both of which are logical additions to the Oracle Communications Session Border Controller's IMS support.

The behavior of these two anchoring points, ATCF and ATGW, is defined by 3GPP in Release 12 of Technical Specification TS 24.237. Oracle Communications developed these functional entities based on the initial version of TS 24.237 Release 10. To align with Release 12, Requirement 4944 introduces the following two new elements to the functional design:

Handsets and Session Continuity

Session continuity can be effected by handsets, which can be dual-mode (3G+WiFi and 3G+WiMAX, for example). Such a handset has two receivers or radios to initiate calls simultaneously. An LTE handset has only one receiver and is able to attach to a single LTE or 3G network at a given time.

The question of session continuity appears in 3GPP standards Release 8, 9, and 10—each building on the previous. Release 8 defines Single Radio Voice Call Continuity (SRVCC) as the mechanism for moving active voice sessions between LTE and existing 2G or 3G circuit networks. In moving over sessions such as these, it is key to keep latency as low as possible to increase the possibility of successful handovers.

Release 9, because of variable signaling latencies within the core network, could not guarantee smooth handovers to circuit networks. And though IMS provides great flexibility in locating application servers remotely from the UE, that flexibility actually increases the latency in signaling media changes to the access network. Due to the high total signaling latency and the difficulty of successfully coordinating handover timing between 3G and 4G call legs, the possibility of call drops increased.

Anchors for Signaling and Media

Release 10 addresses the latency concern by proposing these two logical entities, the anchoring points called the ATCF and ATGW. The Oracle Communications Session Border Controller (SBC) can fulfill the tasks set to both entries:

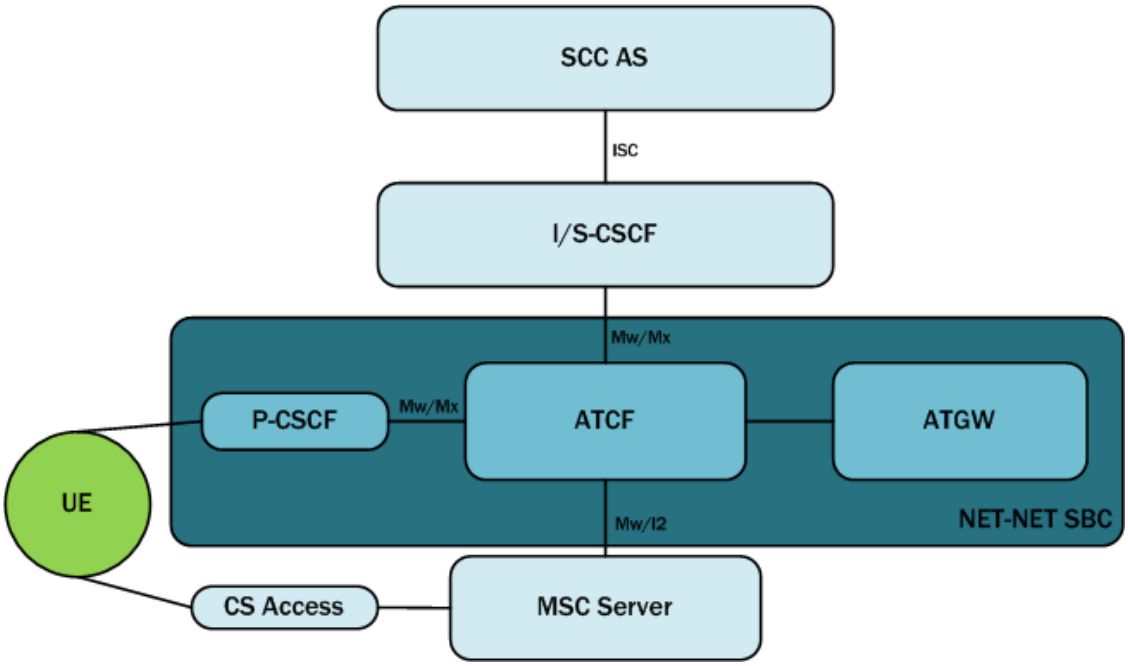
- ATCF—A signaling anchor point co-located with the UE access network. The ATCF is responsible for:
  - Allocating the Session Transfer Number for Single Radio (STN-SR).
  - Instructing the ATGW to anchor the media path for originating and terminating sessions.
  - Tracking sessions (in alerting, active, or held states) so it can perform the access transfer of the selected session. Tracking this information allows the ATCF to support transferring the first session.
  - Performing Access Transfer and updating the ATGW with the new media path for the access call leg, without requiring updating from the remote leg.
  - After the access transfer, updating the Service Centralization and Continuity Application Server (SCC AS) that the transfer has taken place, ensuring that the Terminating Access Domain Selection (T-ADS) has updated information about the access currently used.
  - Handling failures during the access transfer.
  - Handling mid-call support for the access transfer using MSC server-assisted mid-call support.

- Access Transfer Gateway (ATGW)—A media anchor point co-located with the UE access network. Controlled by the ATCF, the ATGW anchors media both for the duration of a call and after the access transfer, based on the local configuration in the serving network.

Originating and terminating sessions are anchored in the ATCF and ATGW already during session set-up. For the first transferred session and the second established session, the SCC-AS provides session state information for the alerting, held, and conference states.

When a UE makes or receives a call, signaling and media are anchored at the ATCF and ATGW. At the point call handover point, the Visited-Mobile Switching Center (V-MSC) receives the handover message from Mobility Management Entity (MME, an EPC network element). The V-MSC then sends a call request to the local ATCF rather than sending the call request home or to the SCC-AS, as defined in 3GPP Release 8. Sending the call request to the ATCF reduces the number of hops required to initiate a media stream change to a new access network.

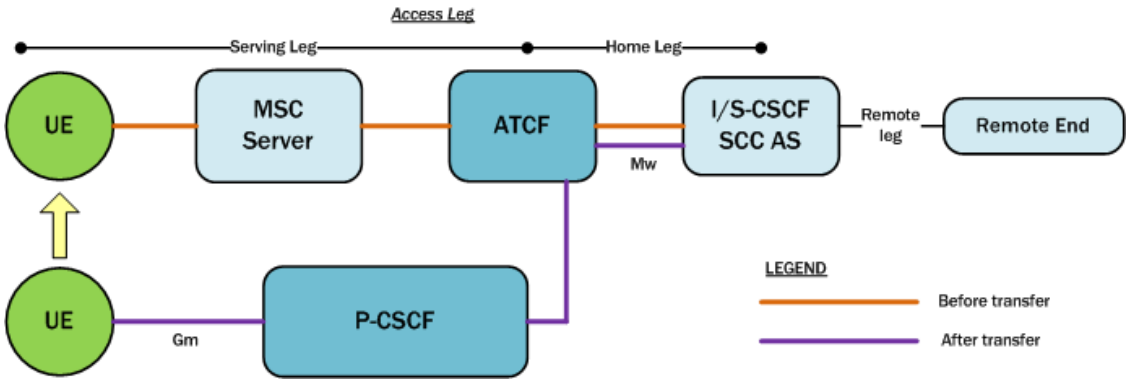
Acting as the ATGW, the SBC can immediately affect media switchovers without further core signaling. In this role, the SBC keeps RTP media continuity between endpoints using its Hide Media Update (HMU) functionality: The elements on the core side of the eSRVCC update will not see changes to SDP sources and destinations because the Synchronization Source identifiers (SSRCs) are masked. Media can thus flow directly to and from the 3G-attached MSC and onward to the UE with minimal interruption.



Note that if the MSC server-assisted mid-call feature is not supported or MSC server is not enhanced for ICS support, the interface between the MSC server and the ATCF will be Mw. If the MSC server-assisted mid-call feature is supported or the MSC server is enhanced for ICS, the interface between the MSC server and the ATCF will be I2.

Architectural View

This diagram is an architectural view of the control and user planes, depicting communication both before and after transfer. This depiction assumes the PGW and the P-CSCF are in the serving network and support IMS voice roaming (if not the home network). So, the ATCF resides in the serving networking (home network if not roaming). The access leg of the session is divided into the serving leg and home leg. In an actual network, there might be other server IMS nodes.



IMS Registration Details

This section discussed the IMS registration process when a UE attempts to register with the home network, but must do so using an ATCF/P-CSCF. Based on the operator policy and the home network's support for eSRVCC, the ATCF allocates an STN-SR to the session and includes itself in the signaling path for subsequent registration-related messaging. To understand whether or not the home network supports eSRVCC, the ATCF uses service-level agreements and can also determine if eSRVCC is activated in the SCC AS by the reception of a C-MSISDN/ATU-SI during session start-up.

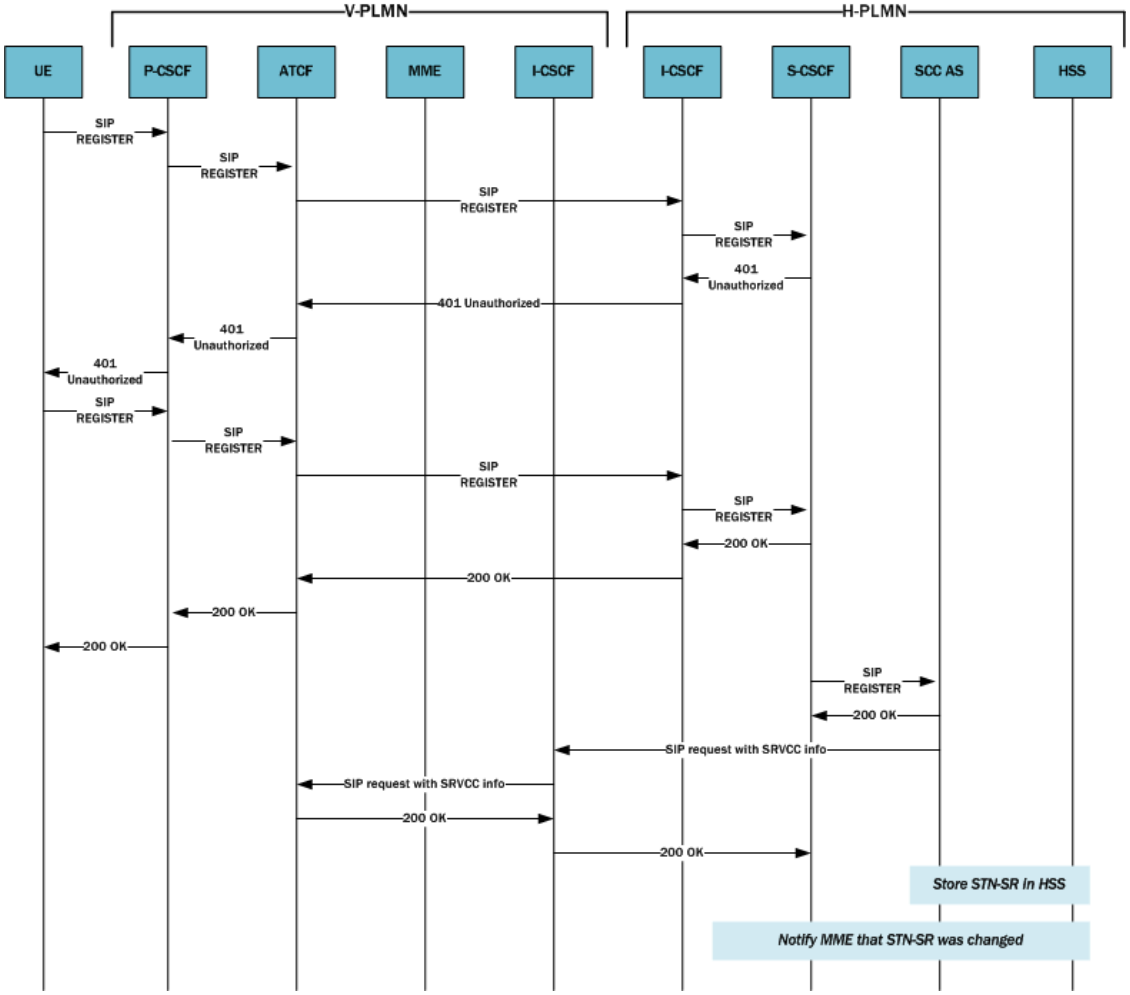
If the ATCF generates an STN-SR, it includes the STN-SR in requests it forwards to the I/S-CSCF. The path and route information for the SIP REGISTER request sent to the S-CSCF is:

- Path—ATCF URI for terminating requests (uniquely identifies registration or registration flow), followed by P-CSCF URI for terminating requests.
  The header field containing the ATCF URI for terminating requests contains the g.3gpp.atcf media feature tag (indicating this URI supports the ATCF role). This tag contains the STN-SR and the ATCF PSI, showing this URI can receive SIP message requests with SRVCC-related information.
- Route—URI of the entry point of the UE's home network.

The ATCF tracks existing registrations for the UEs it has served. Each registration is identified by the P-CSCF path URI. The following information remains in the ATCF's registration cache: the S-CSCF service route URI, the ATU-STI, and the C-MSISDN. When a UE's registration expires or it is de-registered, the ATCF can remove any SRVCC information bound to the registration.

This ladder diagram shows the IMS registration process between these entities:

- The Visited Public Land Mobile Network (V-PLMN)—The network a mobile subscriber uses when that subscriber is roaming.
- The Home Public Land Mobile Network (H-PLMN)—The mobile subscriber's home network.



SIP Register Request UE to ATCF

The following is an example of the SIP REGISTER request the UE sends to the ATCF/ P-CSCF.

```
REGISTER sip:home1.net SIP/2.0
Via: SIP/2.0/UDP
[5555::aaa:131313:ccc:eee];comp.sigcomp;branch=z9hG4bRnasiuen8
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD; utran-cell-id-3gpp=234151DOFCE11
From: <sip:user1_public1@home1.net>;tag=2hiue
To: <sip:user1_public1@home1.net>
Contact: <sip:
[5555::aaa:bbb:ccc:eee];comp=sigcomp>;+sip.instance="<urn:gsma:imei:90420156-
    025763-0>;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Call-ID: E05133BD26DD
Authorization: Digest username="user1_private@home1.net",
realm="registrar.home1.net", nonce="",
    uri="sip:home1.net", response=""
Security-Client: ipsec-3gpp; alg=hmac-sha-1-96; spi-c=23456789; spi-
s=12845678; port-c=1234;
    port-s=5678
```

```
Require: sec-agree
Proxy-Require: sec-agree
CSeq: 1 REGISTER
Supported: path, gruu
Content-Length: 0
```

SIP Register Request ATCF to S-CSCF

The following is an example of the SIP REGISTER request the ATCF sends to the S-CSCF.

```
REGISTER sip:home1.net SIP/2.0
Path: <sip:termsdgfdfwe@actf.visited2.net>;+g.3gpp.atcf="<tel:
+1-237-888-9999>";+g.3gpp.atef-
    psi="<sip:actf.visited2.net>",<sip:aga2gfgf@pcscf1.visited2.net:5070;ob>
Route: <sip:icscf.home1.net;lr>
P-Visited-Network-ID:
P-Charging-Vector:
Via: SIP/2.0/UDP actf.visited2.net:5060;branch=z9hG4bKnas5889; SIP/2.0/UDP
    pcscf1.visited2.net:5060;branch=z9hG4bKnas56565, SIP/2.0/UDP

[5555::aaa:bbb:ccc:eee];comp=sigcomp;branch=z9hG4bKnasiuen8;rport=5060;receive
d=5555::aaa:bbb:ccc:eee
Max-Forwards: 68
P-Access-Network-Info:
From:
To:
Contact:
Call-ID: Authorization:
Require:
Proxy-Require:
CSeq:
Supported:
Content-Length:
```

SIP 200 OK from S-CSCF

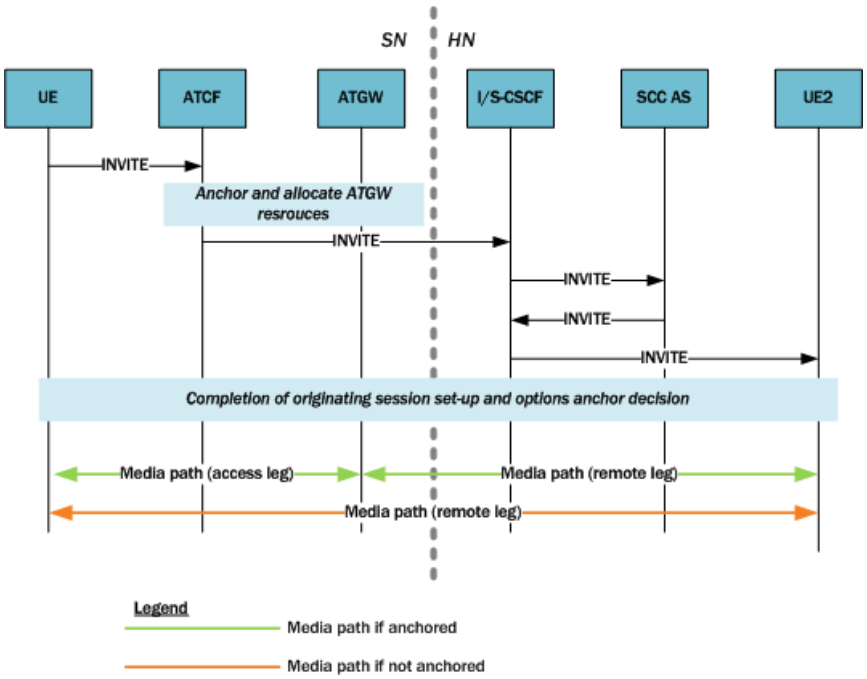
The following is an example of the SIP 200 OK from the S-CSCF.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP icscf1_p.home1.net;branch=z9hG4bK351g45.1, SIP/2.0/UDP
    pcscf1.visited1.net;branch=z9hG4bK240f34.1, SIP/2.0/UDP
    [5555::aaa:bbb:ccc:ddd]:1357;comp=sigcomp;branch=z9hG4bKnashds7
Path: <sip:term@pcscf1.visited1.net;lr;ob>
Service-Route: <sip:orig@scscf1.home1.net;lr>
From:
To:
Call-ID:
Contact: <sip:[5555::aaa:bbb:ccc:ddd]:1357;comp=sigcomp>;
   pub-gruu=" sip:user1_public1@home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
   ;temp-gruu="sip:tgruu.7hs==jd7vnzga5w7fajsc7-ajd6fabz0f8g5@example.com;gr"
   ;+sip.instance="< urn:gsma:imei:90420156-025763-0 >"+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-
   service.ims.icsi.mmtel";+g.3gpp.ics="principal";+g.3gpp.accesstype="cellular1"
   ;expires=600000;+g.3gpp.iut-controller
CSeq:
Supported: path, outbound
Require: outbound
Date: Wed, 11 July 2001 08:49:37 GMT
P-Associated-URI: <sip:user1_public2@home1.net>, <sip:user1_public3@home1.net>, <sip:+1-212-555-
   1111@home1.net;user=phone>
Content-Length:
```

Originating Sessions for SRVCC with ATCF

For initial SIP requests, the ATCF distinguishes SIP INVITE requests with the ATCF URI for originating requests in the topmost Route header field. And when receiving such an originating SIP INVITE request, the ATCF will do the following prior to forwarding it:

- Insert the Record-Route header field with its own SIP URI.
- If the latest SRVCC information received for a session contains a C-MSISDN and ATU-STI:
  - The ATCF will associate the session being established with the C-MSISDN and ATU-STI bound to the registration.
  - The ATCF will replace the SDP offer in the originating SIP INVITE with updated SDP the ATGW provides. Replacement occurs if the originating SIP INVITE contains SDP and a determination to anchor media has been made (according to the operator policy as specified in the 3GPP standard).



SIP INVITE for SRVCC Using the ATCF

The following is an example of the SIP INVITE the UE sends to the ATCF/P-CSCF.

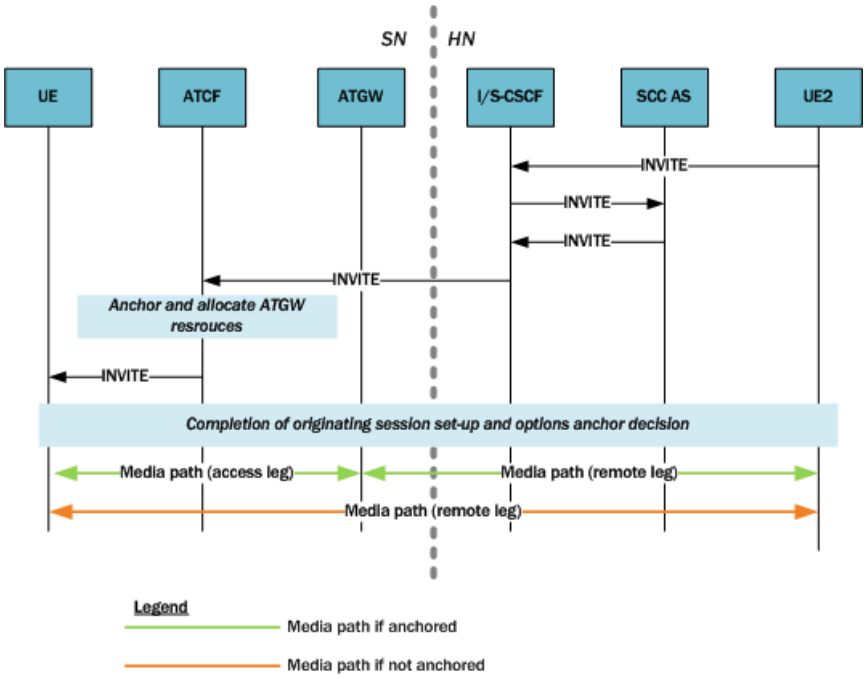
```
INVITE tel:+1-212-555-2222 SIP/2.0
Via: SIP/2.0/UDP [5555::aaa:bbb:ccc:ddd]:1357;comp=sigcomp;branch=z9hG4bKnashds7
Max-Forwards: 70
Route: <sip:pcscf1.visited2.net:7531;lr;comp=sigcomp>, <sip:orig@scscf1.home1.net;lr>
P-Preferred-Identity: "John Doe" <sip:user1_public1@home1.net>
P-Preferred-Service: urn:urn-7:3gpp-service.ims.icsi.mmtel
P-Access-Network-Info: 3GPP-UTRAN-TDD; utran-cell-id-3gpp=234151D0FCE11
Privacy: none
From: <sip:user1_public1@home1.net>;tag=171828
To: <tel:+1-212-555-2222>
Call-ID: cb03a0s09a2sdfglkj490333
Cseq: 127 INVITE
Require: sec-agree
Supported: precondition, 100rel, gruu
Proxy-Require: sec-agree
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi-c=98765432; spi-s=87654321; port-
   c=8642; port-s=7531
Contact: <sip:user1_public1@home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
   00a0c91e6bf6;comp=sigcomp>;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Accept-Contact: *;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: (…)

v=0
o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ddd
s=-
c=IN IP6 5555::aaa:bbb:ccc:ddd
t=0 0
m=audio 3456 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event
```

Terminating Sessions for SRVCC with ATCF

For initial SIP requests, the ATCF identifies SIP INVITE requests with the ATCF URI for terminating requests in the topmost Route header field. These are called terminating SIP INVITE requests.

When it receives a terminating SIP INVITE, the ATCF does the following if the INVITE has a Record-Route header field with the g.3gpp.srvcc media feature tag:

- The ATCF inserts a Record-Route header field with its own SIP URI
- If the latest SRVCC information received for a session contains a C-MSISDN and ATU-STI,
  - The ATCF associates the session being established with the C-MSISDN and ATU-STI bound to the registration, and
  - The ATCF replaces the SDP offer in the originating SIP INVITE with updated SDP provided by the ATGW. Replacement occurs if the terminating SIP INVITE contains an SDP offer and a determination to anchor media has been made (according to the operator policy as specified in the 3GPP standard).



SIP INVITE from UE2 ATCF

The following is an example of the SIP INVITE UE2 sends to the ATCF/P-CSCF.

```
INVITE <sip:user1_public1@home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-
a765-00a0c91e6bf6> SIP/2.0
Via: SIP/2.0/UDP sccas1.home1.net;branch=z9hG4bKnas34r5
Max-Forwards: 67
Route: <sip:scscf1.home1.net:lr>
P-Asserted-Identity: <tel: +1-237-555-2222>
P-Charging-Function-Addresses: ccf=[5555.1399:c88:d77:e66];
ccf=[5555::a55:1344:c33:d22];
    ecf=[5555::1ff:2ee:3dd:4ee]; ecf=[5555.6aa:7bb:8cc:9dd]
P-Charging-Vector: icid-value="AyretyU0dm+602IrT5tAFrbHLso=023551024"; orig-
ioi="type3home1.net"
P-Access-Network-Info:
Privacy: none
From: <tel: +1-237-555-2222; gr=hdg7777ad7aflzig8sf7>;tag=171828
To: <tel:+1-237-555-1111>
Call-ID: cb03a0s09a2sdfg1kj490333
Cseq: 127 INVITE
Supported: 100rel, precondition
Require: sec-agree
Proxy-Require: sec-agree
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi=87654321; port=7531
Contact: <sip:user2_public1@home2.net;gr=urn:uuid:2ad8950e-48a5-4a74-8d99-
    ad76cc7fc74>;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Accept: application/sdp, application/3gpp-ims+xml
Content-Type: application/sdp
Content-Length: (...)
```

```
v=0
o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ddd
s=
c=IN IP6 5555::aaa:bbb:ccc:ddd
t=0 0
m=audio 3456 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; mode-change-period=2
a=rtpmap:96 telephone-event
a=maxptime:20
m=message 0 TCP/MSRP 98
a=accept-types:text/plain
```

TS 24.237 Proposed Changes

The Oracle Communications Session Border Controller implements a proposed change in the processing of failed or cancelled SRVCC sessions. The new processing model has been presented to the 3GPP for inclusion in TS 24.237, IP Multimedia (IM) Core Network (CN) subsystem IP Multimedia Subsystem (IMS) service continuity; Stage 3.

Sections 12.2.4.13, 12.3.3.1, and 12.3.3.1A of 3GPP TS 24.237, IP Multimedia (IM) Core Network (CN) subsystem IP Multimedia Subsystem (IMS) service continuity; Stage 3, describe procedures in response to SRVCC handover cancellation. These procedures allow the UE to generate a e-INVITE request with the Reason header field containing a value of “SIP”, a “cause” parameter set to a value of “487” (Request Terminated), and with reason-text parameter set to a value of either “handover cancelled” or “failure to transition to CS domain”.

SCC AS processing, which does nor release the original access leg, is defined in Section 12.3.3.1 and subclause 12.3.3.1A. Section 13.3.1 describes the handling of subsequent UPDATE requests. There are, however, no defined procedures at the ATCF response to this scenario with the result that the ATCF fails to reconfigure the ATGW to reconnect the bearer in LTE.

Oracle Corporation, in conjunction with other interested parties has proposed the addition of a new Section 12.7.2.3.3 to TS 24.237. This section requires the following ACTF behavior.

Requirement 1: When the ATCF receives either a

- SIP BYE request on the Source Access Leg containing a Reason header field containing a SIP 503 (Service Unavailable) response code, that is terminating an established dialog or an early dialog on the Source Access Leg
- SIP CANCEL request on the Source Access Leg with the Reason header field containing a SIP 503 (Service Unavailable) response code then, that is terminating an early dialog on the Source Access Leg originated by the SC UE, or
- SIP 503 (Service Unavailable) response on the Source Access Leg, that is terminating an early dialog on the Source Access Leg terminating at the SC UE

Then -- The ATCF shall retain session state information and ATGW resources associated with the session until either it receives a SIP INVITE request due to STN-SR, or a specified time period elapses (default value is 8 seconds).

The session remains recognizable for SRVCC access transfer as described in Section 12.7.2.1.

The SIP BYE request is forwarded to the SCC AS, which also delays release of the session, as described in Section 12.3.3.2.

Requirement 2: If the transferable session set determined in Section 12.7.2.1 does not contain any sessions and the identity in the P-Asserted-Identity header field is a C-MSISDN that is not bound to a registration path in the ATCF, the ATCF shall respond with a SIP 404 (Not Found) response.

Requirement 3: When the ATCF receives a SIP re-INVITE request containing Reason header field containing protocol “SIP” and reason parameter “cause” with value “487” on the original source access leg,

- after having initiated an access transfer that was triggered by a SIP INVITE request due to STN-SR, and
- the SIP INVITE request due to ATU-STI transaction is not yet completed

Then -- The ATCF shall wait until this transaction has completed and then continue with the steps described in Requirement 4.

Requirement 4: When the ATCF receives a SIP re-INVITE request(s) containing protocol “SIP” and reason parameter “cause” with value “487” after having performed an access transfer that was triggered by a SIP INVITE request due to STN-SR,

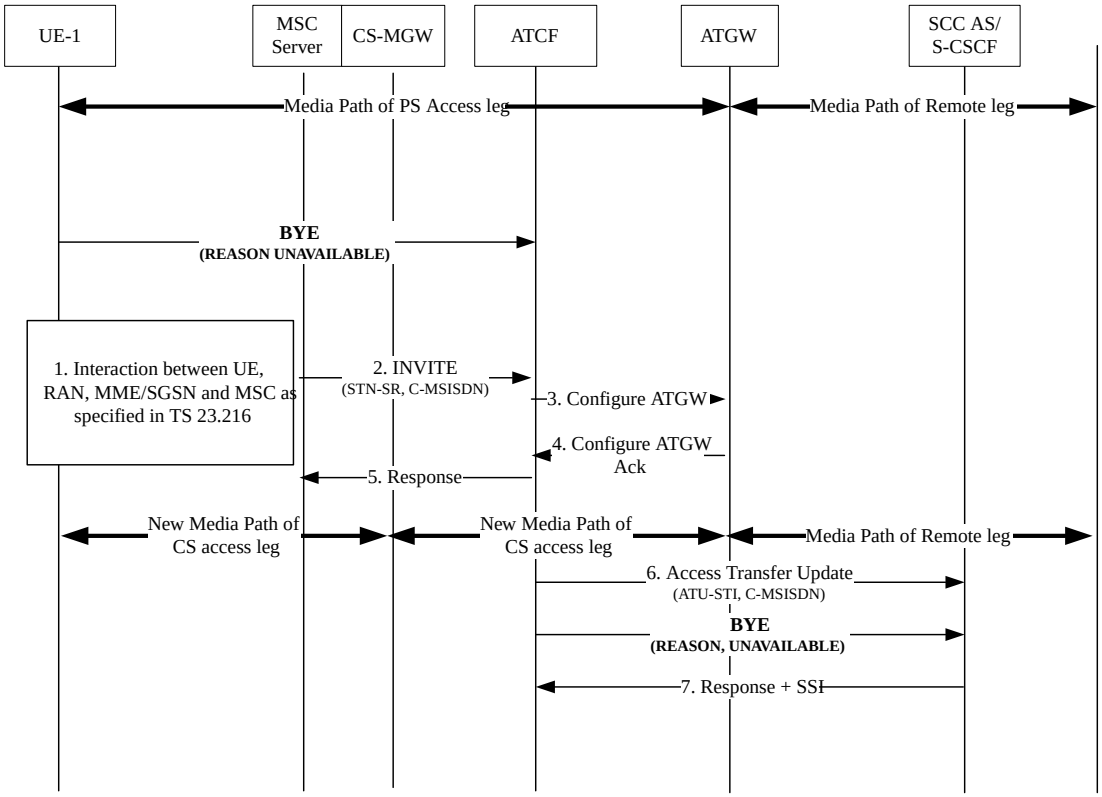
Then -- The ATCF shall act as B2BUA as described in Section 5.6 and shall.

1. Interact with ATGW to provide information needed in the procedures below and to request ATGW to start forwarding the media from the remote UE to the local UE. The details of interaction between ATCF and ATGW are out of scope of this document.
2. Send a SIP 200 (OK) response to the received SIP re-INVITE request. The SIP 200 (OK) response contains the SDP answer that includes the ATGW ports and the IP addresses as provided by the ATGW and the media used on the original source access leg as before the access transfer; and
3. Forward the received reINVITE with the Reason header intact to the SCC AS on the existing source dialog with the SDP offer containing the ATGW IP addresses and ports towards the remote UE as provided by the ATGW.

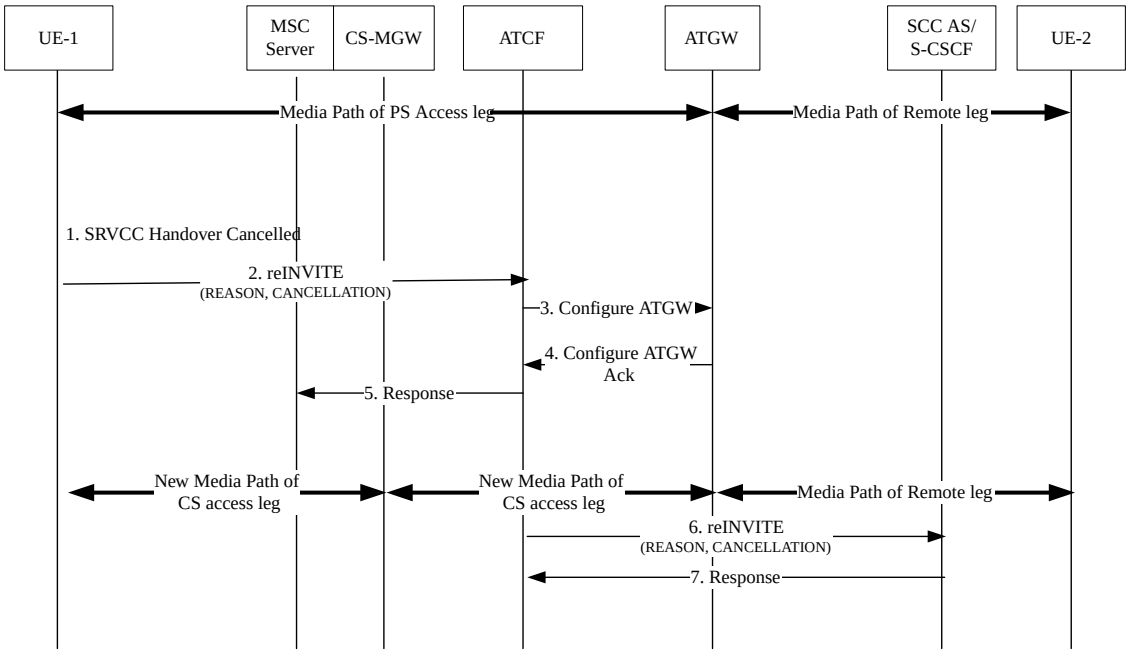
This proposed behavior, which requires no user configuration) is implemented in Version S-CX7.2.0 and later releases.

Related call flows are shown below.

BYE before handover INVITE



Cancellation after handover complete.



Accounting

eSRVCC calls involve two SIP sessions, the accounting records for which need clear association. To correlate the records, the Oracle Communications Session Border Controller performs these actions for eSRVCC calls:

- When it detects a session handover, the system generates an Interim accounting record for the initial SIP session. This way, the records provide the exact time of the handover (i.e., the timestamp of the Interim record).
- The Oracle specific AVP Generic-ID will appear in Start, Interim, and Stop records for the handover SIP session. The Generic-ID contains the Call ID of the initial SIP session, which helps to correlate the two SIP sessions.
  - For RADIUS accounting records, refer to Oracle specific VSA 40.
  - For DIAMETER accounting records, refer to Oracle specific VSA 30.
- The Stop record for the initial SIP session will not have media flow information because the media session is considered part of the handover SIP session.
- If you are using QoS, the Stop records for the handover SIP session reflects the cumulative QoS statistics for both the initial SIP session and the handover SIP session.

External Bandwidth Management

Is you are using the Oracle Communications Session Border Controller's external bandwidth management for eSRVCC calls, note these considerations:

- At the time of handover and if the new handover realm has the same external policy server configured as the initial SIP session, the Rx will continue seamlessly with the same DIAMETER session identifier directed toward the policy server. From the perspective of the policy server, the same session is simply continuing.
- At the time of handover and if the new handover realm does not have an external policy server configured, policy server services will stop.

ATCF Configuration

ATCF functionality requires configuration in the **sip-config** and **sip-interface** configuration elements.

1. Access the **sip-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. **atcf-stn-sr**—Enter the value of the Session Transfer Interface, Single Radio (STN-SR). Your entry will resemble this example: tel:+1-237-555-9999. This value will be included in the g.3gpp.atcf media feature tag that the ATCF allocates in the REGISTER message.
3. **atcf-psi-dn**—Enter the value to use for the Public Service Identity Domain Name (PSI-DN). Your entry will resemble this example: sip:atcf.visited2.com. If configured, this value will be included in the g.3gpp.atcf media feature tag that the ATCF allocates in the REGISTER message. If you leave this parameter blank, the Oracle Communications Session Border Controller will set this value to the SIP interface accress.

4. **atcf-route-to-sccas**—If you leave this parameter set to disabled (default), the handover update, an INVITE, is routed to the IMS Core. If you set this parameter to enabled, the Oracle Communications Session Border Controller will route the handover update directly to the Service Centralization and Continuity Application Server (SCC-AS).
5. Type **done** to save your configuration.

Continue to and select the existing sip-interface configuration element targeted for this ATCF configuration:

```
ORACLE(sip-config)# exit
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# select
<RealmID>:
1: public

selection: 1
ORACLE(sip-interface)#
```

6. **sip-atcf-feature**—Change this parameter from **disabled** (default) to **enabled** to turn on ATCF functionality for the ingress SIP interface.
7. Type **done** to save your configuration.

ATCF INVITE ICSI Matching

The Oracle Communications Session Border Controller can check, on reception of an INVITE on an ingress sip-interface that has a configured ATCF and before applying any of the already implemented logic, whether the incoming INVITE includes the ICSI (Instantaneous Channel-State Information) of the requested service. The ATCF will be involved in the call flow when the configured ICSI value matches the ICSI value in the original INVITE; otherwise the handoff call will be rejected with code 480 (Temporarily Unavailable) or 404 (Not Found).

The system looks for the ICSI string in the following headers:

- P-Preferred-Service
- P-Asserted-Service
- Feature-Caps (within the "g.3gpp.icsi-ref" feature-capability indicator)
- Accept-Contact (within the tag-value within the g.3gpp.icsi-ref media feature tag)

An example of the ICSI string in the P-Preferred-Service or P-Asserted-Service header is "urn:urn-7:3gpp-service.ims.icsi.mmtel". Examples of the ICSI string in the Feature-Caps or Accept-Contact headers are "+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel" and "g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel". The ATCF will be involved in the call flow only when the ICSI matches. Configure the **atcf-icsi-match** parameter with the ICSI string you want to match. If **atcf-icsi-match** is blank, the check is not done and the behavior remains the same as before.

ATCF INVITE ICSI Matching Configuration

You can configure the Oracle Communications Session Border Controller to check, on reception of an INVITE on an ingress sip-interface that has a configured ATCF and before applying any of the already implemented logic, whether the incoming INVITE includes the ICSI (Instantaneous Channel-State Information) of the requested service and, if so, to involve the ATCF in the call flow.

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **atcf-icsi-match** — enter the ICSI string you want to match.
4. Type **done** to save your configuration.

SRVCC PS-CS Access Transfer

This section describes various scenarios for transferring sessions between Packet-Switched (PS) and Circuit-Switched (CS) networks, providing details about the following cases:

- Mobile Switching Center (MSC) server-assisted mid-call feature supported by the SCC AS
- Call flows for the MSC server-assisted mid-call feature supported by the SCC AS
- Failure cases
- Confirmed dialog
- Early dialog

For PS-CS transfers in SRVCC, the ATCF performs several actions based on the STN-SR when it first receives the SIP INVITE. First, the ATCF determines the set of transferrable sessions. It defines this set as the SRVCC-transferrable sessions that are associated with a C-MSISDN matching the URI in the P-Asserted-Identity header field in the INVITE. The ATCF compares the INVITE to its set of transferrable sessions to see if the INVITE is part of the set; if so, its responds with 2xx response for the initial INVITE.

Active session transfers require the ATCF to act as a back-to-back user agent (B2BUA) if the session undergoing transfer has media anchored at the ATGW. In this case, the ATCF responds with a 200 OK to the INVITE; the OK carries an SDP answer with ATGW IP address and port information. Then the ATCF updates the following information and sends the INVITE to the remote UE:

- Original SDP offer is replaced with an SDP offer containing media information currently in use with the ATGW IP address and port information.
- The Request-URI with the ATU-SI associated with the session being transferred is inserted.
- The Target-Dialog header field with the dialog identifier of the session being transferred is inserted.

When the ATCF receives an SDP answer and if media is anchored at the ATGW for the session being transferred, the ATCF directs the ATGW to start sending and receiving media to/from the remote UE according to the newly received answer.

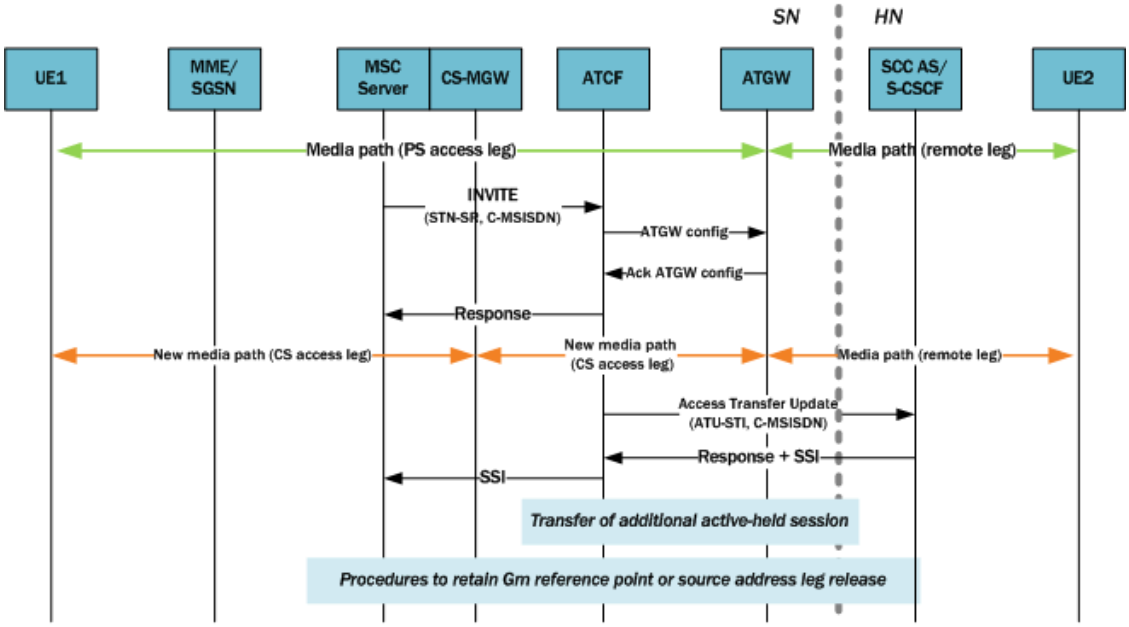
If it receives a BYE with a 503 Service Unavailable header on the source access leg, the ATCF retains session state information and ATGW resources associated with the session until either:

- The ATCF receives an INVITE request because of STN-SR, or
- An operator-defined time period elapses. The default value for the operator-defined time period is eight (8) seconds.

Thus, the session remains recognizable for SRVCC transfer for a time. Release of the session is also delayed by the ATCF's forwarding the BYE to the SCC AS. If the transferrable session cannot progress, the ATCF responds with a 404 Not Found. For transfer cases when only hold or alerting sessions exist and the session in question is an early dialog or a confirmed dialog with inactive media, the ATCF replaces the Request-URI received in the original INVITE with the ATU-STI associated with a session in the transferrable set before forwarding the request; this proxy-role behavior complies with the 3GPP standard.

MSC Server-Assisted Mid-Call Feature Supported by SCC AS

The following scenario assumes an active session between UA1 and UA2, and that UA1 attaches to the CS domain. It is also assumed that the CS-MGW supports the codecs used for LTE voice calls, minimizing the likelihood that the ATCF will need to instruct the ATGW to insert codecs.



Upon receiving the Access Transfer message, the ATCF identifies the correct anchored session and proceeds with the transfer of the most recently active session. Using a Configure ATGW message, the ATCF replaces the existing PS access leg media path information with new CS access leg media path information for the ATGW. However, the ATCF might instruct the ATGW to continue using the local port of the PS access leg media path. Once the ATGW acknowledges the ATCF's Configure message, the ATCF sends the MSC server an Access Transfer response and the media path is moved to the CS when receiving SDP information.

Voice break interruption starts either when media moves to the CS MGW (controlled by the MSC server enhanced for SRVCC) or when the UE relocates to the target—whichever comes first. When the UE tunes to the target or media switches to the CS MGW (whichever is last),

the voice break interruption ends. The assumption is that media is switched to the CS MGW during the time to UE tunes to the target.

After receiving the Access Transfer message, the ATCF re-establishes communication with the SCC AS and updates the SCC AS. When the MSC server supports the mid-call feature, the ATCF also communicates this fact to the SCC AS. The ATU message initiates a new dialog between the ATCF and SCC AS, a dialog the SCC AS associates with the old dialog using the C-MSISDN. This new dialog is necessary because it replaces the old dialog set up over PS access, thereby assuring that if the PS user registration expires the new home leg will not be released or even affected.

The following shows a sample INVITE request the ATCF sends to the S-CSCF. Note the following:

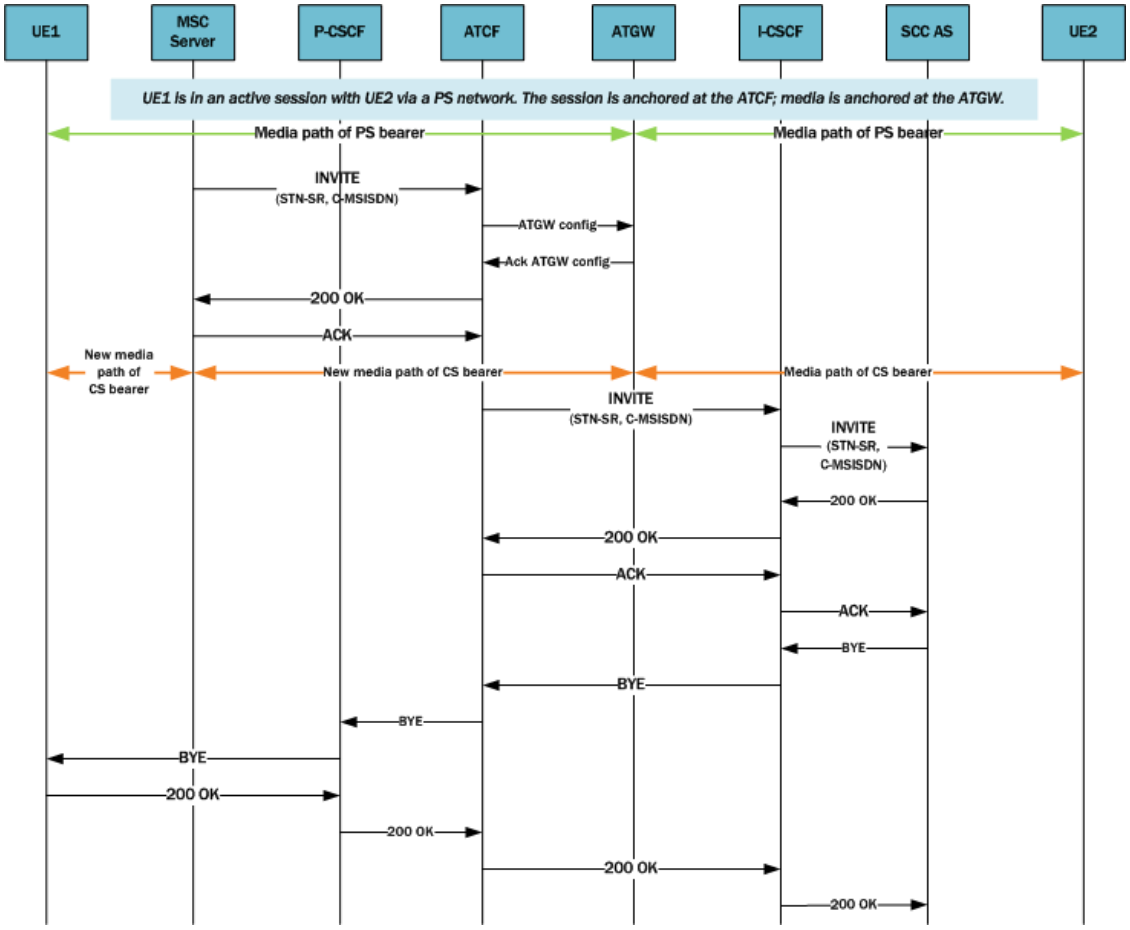
- The Request-URI header contains the ATU-STI, as routed to the SCC AS.
- The Target dialog header specifies the existing dialog is associated with this request.
- The P-Asserted-Identity header reflects the C-MSISDN of the UE being served.
- The From header reflects the C-MSISDN of the serving UE.
- The SDP reflects the media information at the ATGW.

```
INVITE sip:AUT-STI1@sccas.home1.net SIP/2.0
Via: SIP/2.0/UDP msc1.visit1.net;branch=z9hG4bk731b87
Max-Forwards: 70
P-Asserted-Identity: <tel:+1-237-555-2222>
P-Charging-Vector: icid-value="AyretyU0dm+602IrT5tAFrbHLso=023551024";orig-
ioi=visit1.net
Privacy: none
From: <tel:+1-237-555-9999>;tag=171828
To: <tel: +1-237-555-4444>
Call-ID: cb03a0s09a2sdfg1kj490334
Cseq: 127 INVITE
Supported: 100re1, precondition, gruu
Target-Dialog: me03a0s09a2sdfgjk1491777; to-tag=774321; from-tag=64727891
Accept-Contact: *;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.mmtel
Contact: <sip:msc1.home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-
a765-00a0c91e6bf6>;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER
Content-Type: application/sdp
Content-Length: (...)

v=0
o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ggg
s=
c=IN IP6 5555::aaa:bbb:ccc:ggg
t=0 0
m=audio 3456 RTP/AVP 97 96
a=tcap:1 RTP/AVPF
a=pcfg:1 t=1
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
```

```
a=fmtp:97 mode-set=0,2,5,7; mode-change-period=2
a=rtpmap:96 telephone-event
a=maxptime:20
```

The SCC AS sends the ATCF a confirmation, including the session state information (SSI) if the SCC AS and the MSC server support the mid-call feature. The MSC server receives the SSI from the ATCF; the access leg for the control has moved to the CS access. When MSC server receives SSI for more multiple active or inactive speech sessions, it initiates transfer directed at the SSC AS for the additional sessions.

The following diagram shows a call session in the active phase for MSC server assisted midcall feature when supported by the SCC AS.



Failure and Cancellation

For cases of failure and/or cancellation, the ATCF behaves in accordance with TS 23.216 [3], Section 8:

- In the case of session failure before the MSC server initiates session transfer, the standard failover procedures (as defined in TS 23.401 [2])apply and no further action is required by the UE.
- In the case of session failure after the UE receives the handover command, the UE attempts to return to the Universal Mobile Telecommunications System (UTRAN) or evolved Universal Mobile Telecommunications System (eUTRAN). The UE initiates

signaling to transfer the session back, which the ATCF handles if the ATCF recognizes the session transfer.

- In the case when handover is cancelled, the UE—having received the handover cancellation message—begins the re-establishment procedure as though it needed to transfer the session to the eUTRAN/UTRAN. If he ATCF identifies this as a transfer session, it handles the session transfer back to the eUTRAN/UTRAN.

Confirmed Dialogs

When an SC UE is engaged in one or more ongoing IMS sessions, it will send a re-INVITE containing:

- An SDP offer, including the media characteristics used in the existing dialog, and
- A Reason header field containing the protocol SIP and reason parameters cause with the value 487 (as specified in RFC 3326 [57]), with reason text reading either Handover cancelled or Failure to transition to CS domain.

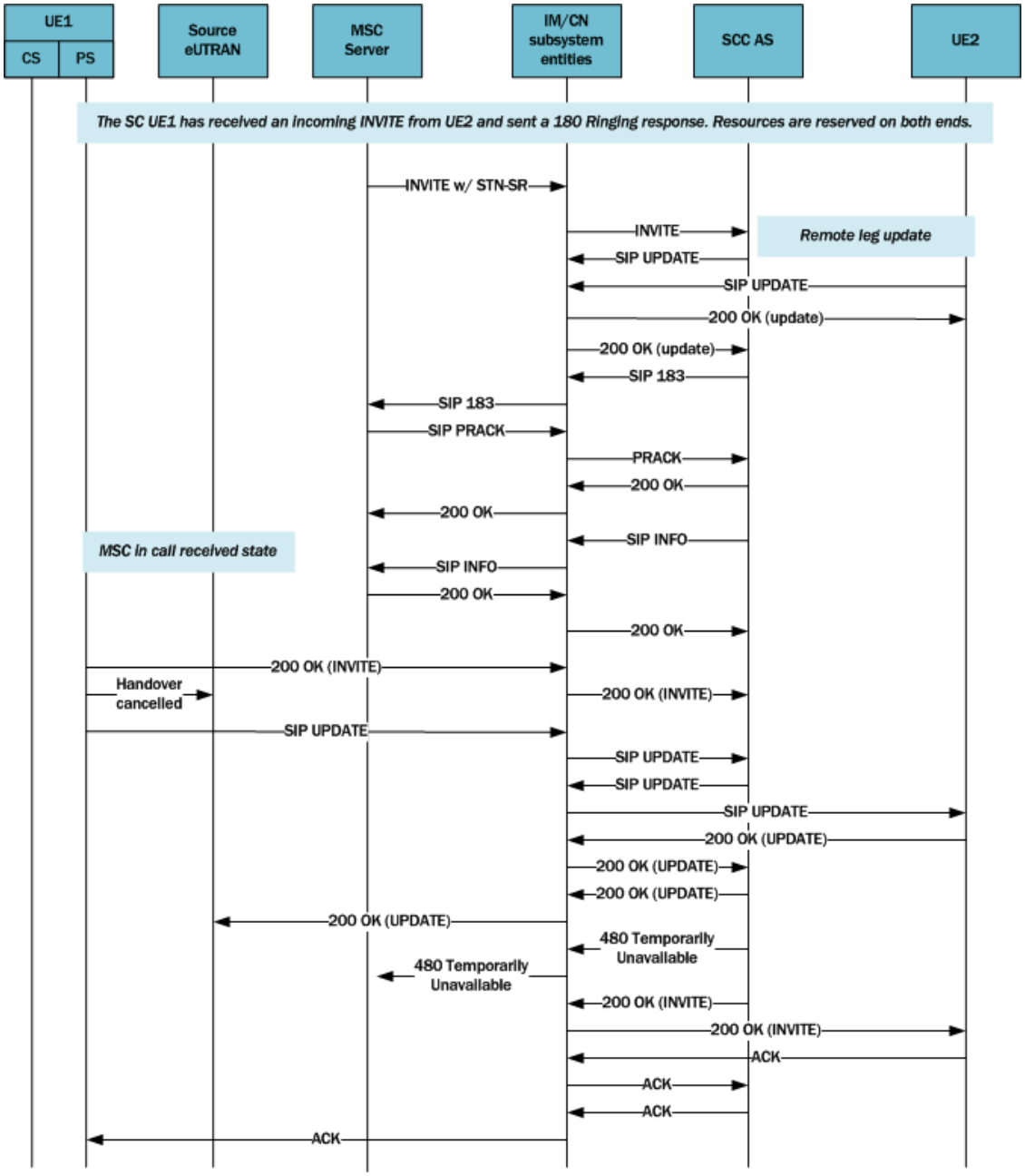
In all other ways, confirmed dialogs behave in accordance with TS 24.229 [2].

Early Dialogs

If the SC UE engages in a session which is in early dialog state, it will send a SIP UPDATE containing:

- An SDP offer, including the media characteristics used in the existing dialog, and
- A Reason header field containing the protocol SIP and reason parameters cause with the value 487 (as specified in RFC 3326 [57]), with reason text reading either Handover cancelled or Failure to transition to CS domain.

In all other ways, early dialogs behave in accordance with TS 24.229 [2].



SRVCC Handover Support in the Pre-Alerting Phase

In addition to other SRVCC support, the Oracle Communications Session Border Controller (SBC) supports procedures to manage the handover from 4G to 3G/2G of sessions in pre-alerting phase. The conditions by which a session is defined as in the pre-alerting phase include the calling party has not yet received a 180 RINGING message.

Refer to TS 24.237 to review the anchoring endpoints' behavior.

To ensure that calls in a pre-alerting phase are transferred between PS and CS networks, set the **sip-feature-caps** value as follows:

- Set **state** to **enabled**
- Set **atcf-management-uri** to **management** or **psi**

- Set **atcf-pre-alerting** to **enabled**

For information on the results of these settings, refer to the *SIP Feature Capabilities* section.

The SBC, as Proxy Call Session Control Function (P-CSCF) or Active Transfer Control Function (ATCF), selects any of the early dialogs as the session being transferred when:

- There are no confirmed dialogs supporting a session with an inactive speech media component ("sendonly" or "inactive" directionality) in the transferable session set, and
- There are one or more dialogs in the transferable session set supporting one session where the SC UE has completed a reliable offer / answer procedure and has an active speech media component ("recvonly" or "sendrecv" directionality).

Applicable dialogs include those for which:

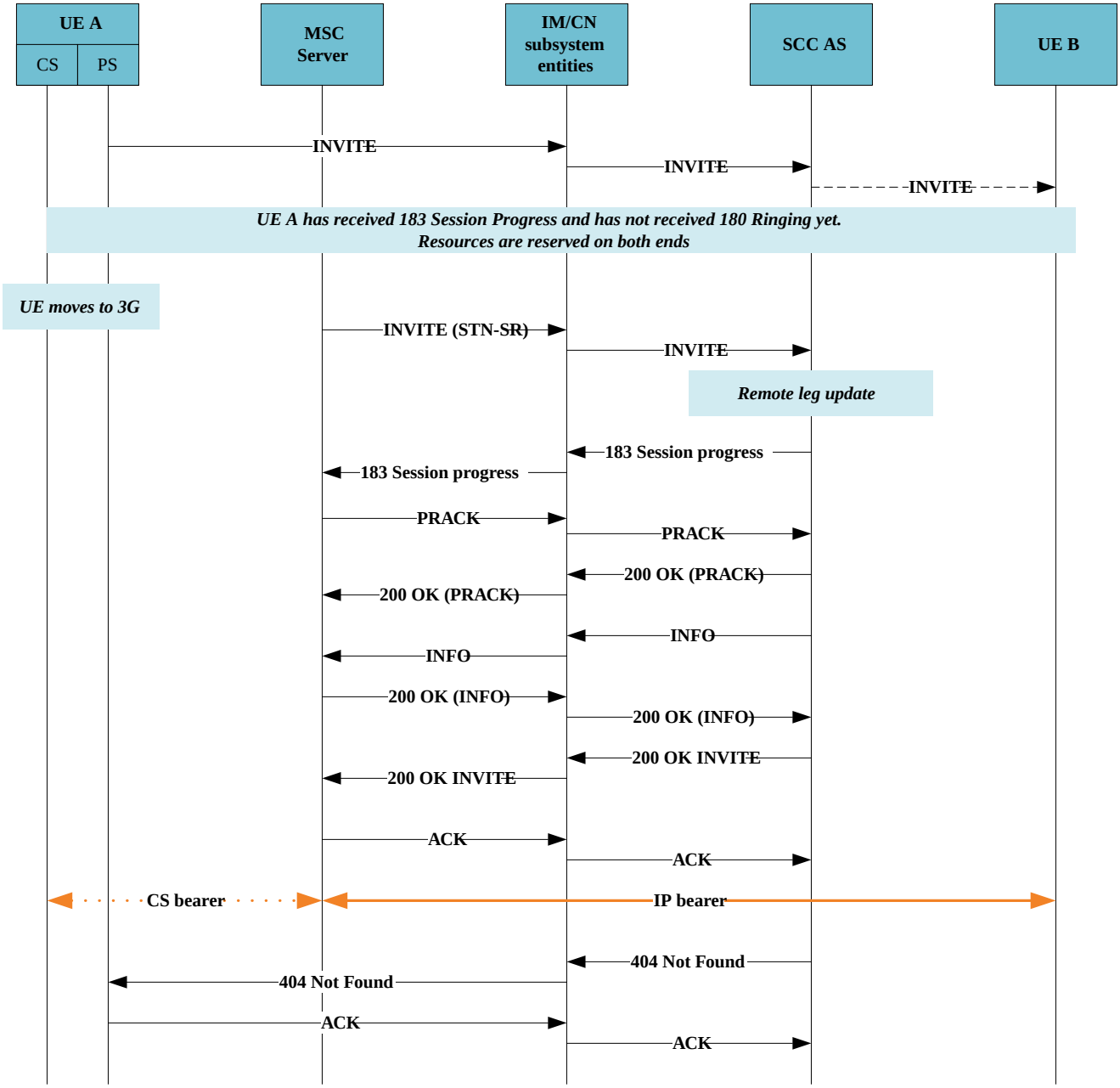
- A SIP 180 (Ringing) response to the SIP INVITE request has not been received yet in any of the existing dialogs.
- All dialogs are early dialogs created by the same SIP INVITE request.
- The Contact header field provided by the SC UE includes the **g.3gpp.ps2cs-srvcc-orig-pre-alerting** media feature tag; and
- The Feature-Caps header field provided by the SCC AS towards the SC UE includes the **g.3gpp.ps2cs-srvcc-orig-pre-alerting feature-capability** indicator.

The range of SBC SRVCC pre-alerting phase support includes:

- SRVCC support for both outgoing (originating) and incoming (terminating) calls
- Responds to media feature tag and fcaps indicator
- SRVCC support of EATF emergency calls
- Handover (HO) cancelling scenarios
- Rx interactions
- SRVCC Error scenarios, including:
  - HO cancellation failures
  - Rx failures
  - Standard SIP transaction failures
- ACLI, SNMP and HDR statistics for applicable, successful and failed SRVCC handovers

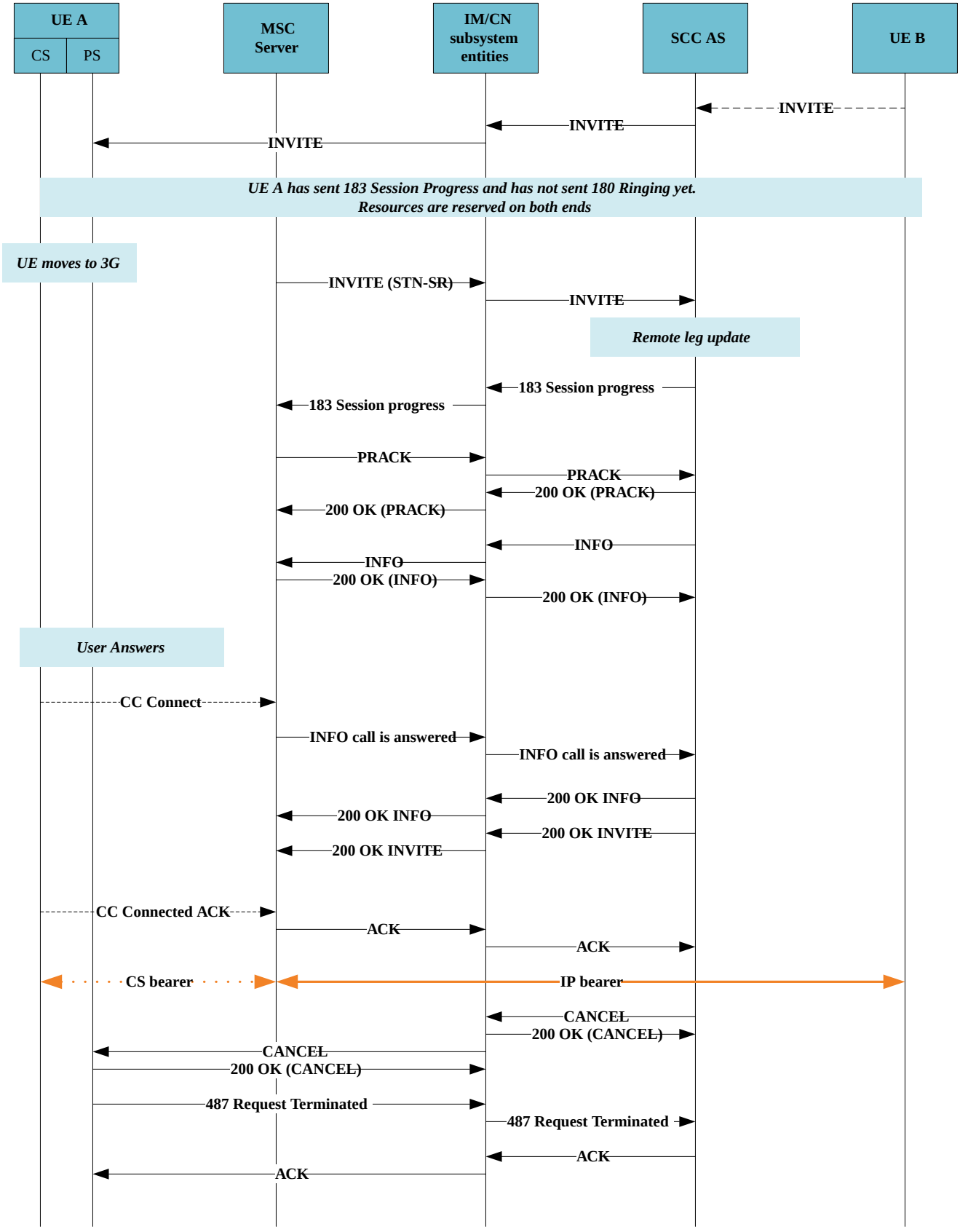
The diagram below shows a flow for an outgoing call that includes an SRVCC handover during the pre-alerting phase. Prior to the alerting phase, the SBC, acting as ATCF (IM/CN subsystem) receives and forwards the new INVITE to update the call to the circuit switched (CS) leg. Ultimately, the signaling between the MSC, the ATCF and the SOC AS succeeds. The components proceed through the alerting phase, eliminating the packet switched based (PS) leg and using the MSC server to manage the CS call.

Figure 18-15 Mobile Originating Call in Pre-Alerting Phase



The diagram below shows a flow for an incoming call that includes an SRVCC handover during the pre-alerting phase. Prior to the alerting phase, the SBC, acting as ATCF (IM/CN subsystem) receives and forwards the new INVITE to update the call to the circuit switched (CS) leg. Ultimately, the signaling between the MSC, the ATCF and the SOC AS succeeds. The components proceed through the alerting phase, eliminating the packet switched based (PS) leg and using the MSC server to manage the CS call.

Figure 18-16 Mobile Terminating Call in Pre-Alerting Phase



SRVCC Handover Support in Alerting Phase

The SBC supports handovers between Packet-Switched (PS) and Circuit-Switched(CS) networks for calls in an alerting phase; that is, a 180 ringing response for the initial INVITE has been sent or received and the SIP final response has not been sent or received.

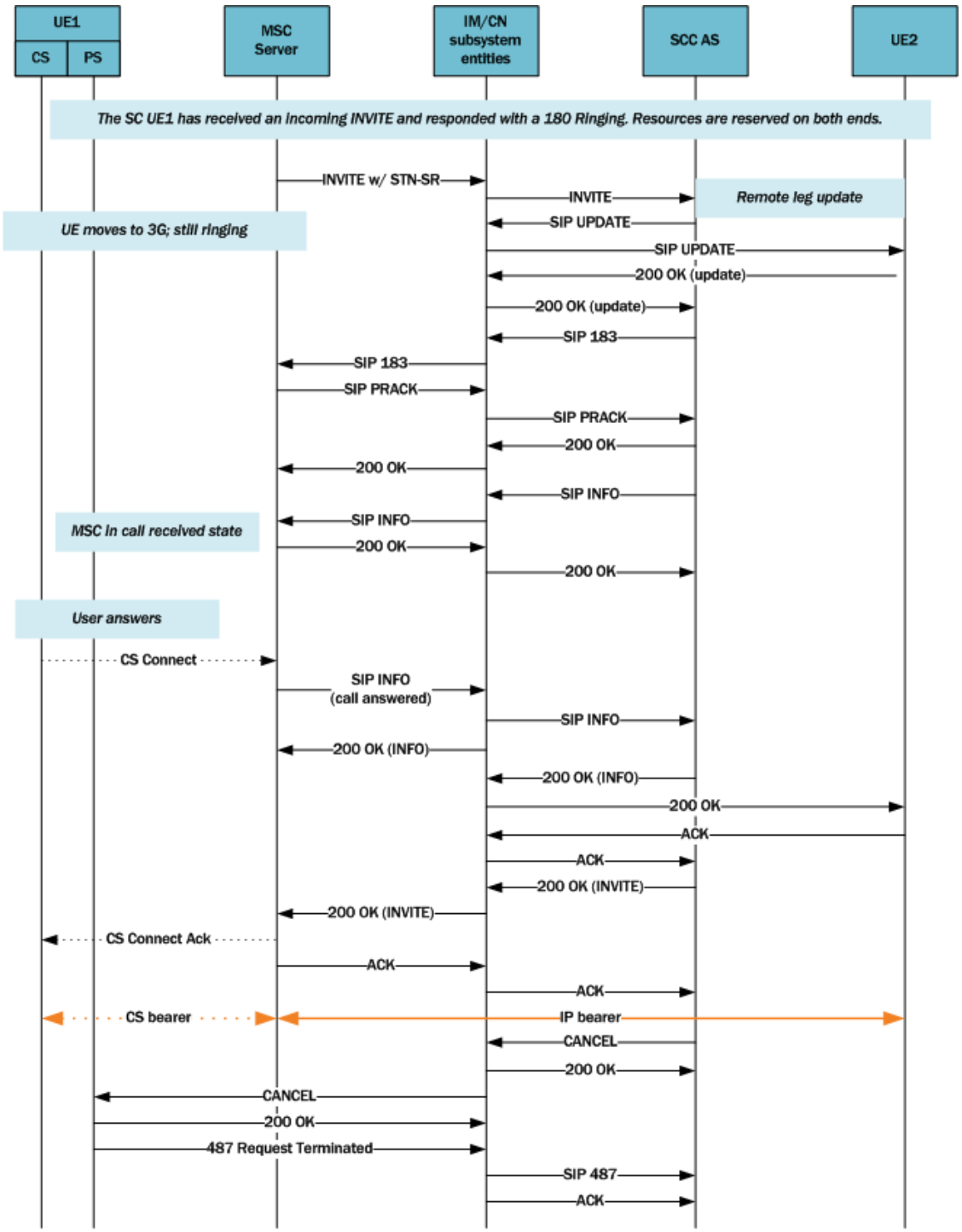
The behavior of the two anchoring points, ATCF and ATGW, is defined by 3GPP in Release 12 of Technical Specification TS 24.237. Oracle Communications developed these functional entities based on the initial version of TS 24.237 Release 10, and has added the **sip-feature-caps** configuration element to align with Release 12.

To ensure that calls in an alerting phase are transferred between PS and CS networks, set the **sip-feature-caps** value as follows:

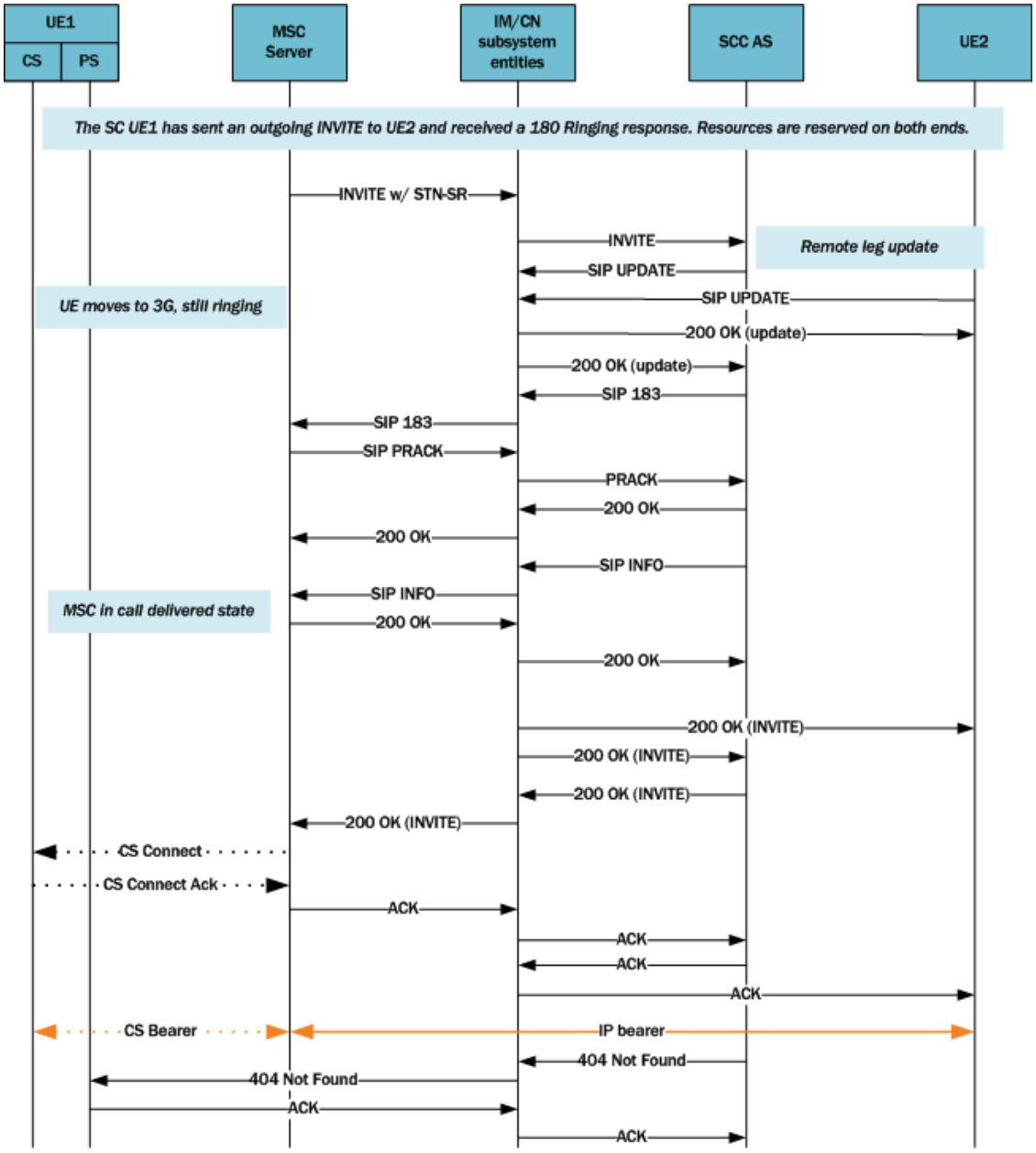
- Set **state** to “enabled”
- Set **atcf-management-uri** to "management" or "psi"
- Set **atcf-alerting** to "enabled"

For information on the results of these settings, refer to the *SIP Feature Capabilities* section.

The diagram below shows an incoming call session during the alerting phase.



This diagram shows an outgoing call session during the alerting phase.



SIP Feature Capabilities

The ATCF can announce a feature capability in a message by transporting the information in the Feature-Caps header, which is supported by the SBC.

The behavior of the two anchoring points, ATCF and ATGW, is defined by 3GPP in Release 12 of Technical Specification TS 24.237. Oracle Communications developed these functional entities based on the initial version of TS 24.237 Release 10, and has added the **sip-feature-caps** configuration element to align with Release 12.

When the **state** is enabled, the SBC includes the Feature-Caps header in applicable SIP messages. The information in the header is dependent on **sip-feature-caps** configuration.

Specifically, When you enable **sip-feature-caps**, regardless of other settings, the SBC inserts the Feature-Caps header with:

- The **g.3gpp.mid-call** capability indicator.
- The **g.3gpp.atcf-path** parameter with a value set to the ATCF URI for terminating requests

Assume the **state** is **enabled** and both **atcf-alerting pre-atcf-alerting** are **disabled**. The SBC adds the Feature-Caps header with the above and:

- The **g.3gpp.atcf** parameter with a value of the configured **atcf-stn-sr** in **sip-config**
- The **g.3gpp.atcf-mgmt-uri** parameter with a value of the configured **atcf-psi-dn** in **sip-config**

Finally, when **state** is **enabled**, **atcf-management-uri** is not disabled, and **atcf-alerting** is **enabled**, the SBC adds the Feature-Caps header with the above and also adds the **g.3gpp.srvcc-alerting** capability indicator. Similarly, the SBC adds the **g.3gpp.srvcc-pre-alerting** capability indicator when **atcf-pre-alerting** is **enabled**.

SIP Feature Capabilities Configuration

You can configure Oracle Communications Session Border Controllers to have the ATCF announce a feature capability in a message by transporting the information in the Feature-Caps header.

1. Access the **session-router** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-feature-caps
ORACLE(sip-feature-caps)#
```

2. **state** — identifies whether to enable the feature and add the Feature-Caps header to messages. Possible values are "enabled" and "disabled". The default value is "disabled".
3. **atcf-alerting** — identifies whether to turn on the alerting feature and add the alerting Feature-Caps header to messages. Possible values are "enabled" and "disabled". The default value is "disabled".
4. **atcf-pre-alerting** — identifies whether to turn on the alerting feature and add the pre-alerting Feature-Caps header to messages. Possible values are "enabled" and "disabled". The default value is "disabled".
5. **atcf-management-uri** — identifies the feature capability indicator that will be used to transport the ATCF management URI. Possible values are "management" and "psi". The default value is "management". When the value is "management" and the value of **state** is "enabled", the Feature-Caps header “g.3gpp.atcf-mgmt-uri” is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element. When the value is "psi" and the value of **state** is "enabled", the Feature-Caps header “g.3gpp.atcf-psi” is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element.
6. Type **done** to save your configuration.

Reporting SRVCC Statistics

You access SRVCC Hand-Over (HO) call counters and statistics from the SBC's via ACLI Show command, SNMP and HDR tools.

Applicable statistic access details include:

- ACLI—Run the command **show sipd srvcc** to display total, successful and failed calls for each handover phase. Find additional detail on **show sipd** from the *Maintenance and Troubleshooting Guide*.
- SNMP—Access the same detail provided by the ACLI command above via SNMP. Get this detail from **ap-sip.mib** mmm. Find additional detail on **ap-sip.mib** from the *MIB Guide*. Example OID and object identifiers include:
  - 1.3.6.1.4.1.9148.3.15.1.1.3.13 — (apSipSRVCCStatsTotalCallsDuringPreAlerting)
  - 1.3.6.1.4.1.9148.3.15.1.1.3.14 — (apSipSRVCCStatsDuringPreAlertingSuccess)
  - 1.3.6.1.4.1.9148.3.15.1.1.3.15 — (apSipSRVCCStatsDuringPreAlertingFailed)
- HDR—Access the same detail provided by the ACLI command above via HDR. Get this detail from the **sip-srvcc** collect group. Find additional detail on the **sip-srvcc** group from the *HDR Reference Guide*. Example HDR objects include:
  - Calls During Pre-Alerting Counter — (0-2^32-1) — Total calls subjected to SRVCC during pre-alerting.
  - During Pre-Alerting Success Counter — (0-2^32-1) — Total successful SRVCC HO during pre-alerting.
  - During Pre-Alerting Failed Counter — (0-2^32-1) — Total failed SRVCC HO during pre-alerting.

S8HR Roaming Compatibility

The Oracle Communications Session Border Controller (SBC) allows you to configure support for S8 Home Routing (S8HR) routing architecture. S8 is the 3GPP-defined Packet Data Network (PDN) Gateway to Serving Gateway (S-GW) interface used when a UE is roaming. S8HR is described in full by 3GPP TR 23.749. The SBC feature provides support for UE connectivity. sec-agree and emergency call services.

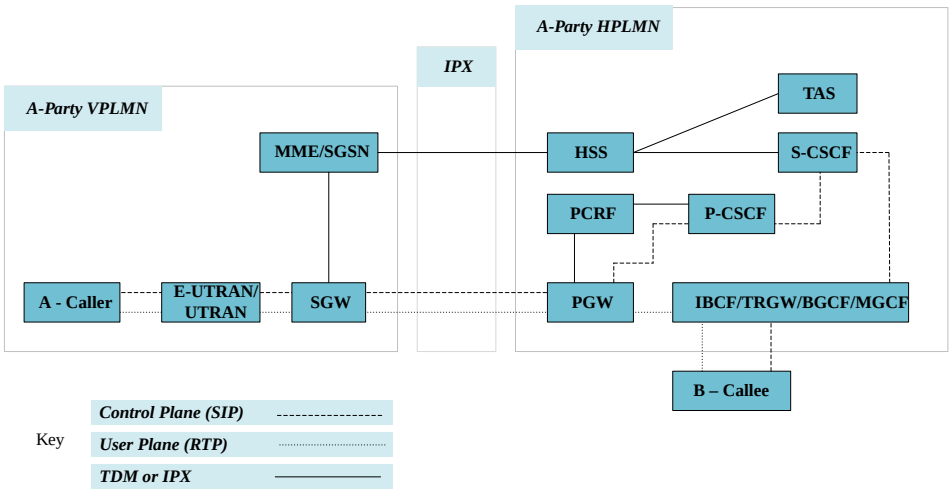
S8HR Overview

S8HR supports roaming in VoLTE networks whereby the IMS infrastructure elements are located in the Home Public LAN Mobile Network (HPLMN), and the UE is roaming in a Visited Public Land Mobile Network (VPLMN).

Applicable IMS infrastructure elements include:

- Packet Data Network Gateway (PGW) —Supports the UNI interface from the roaming UE to the HPLMN
- Policy and Charging Rules Function (PCRF)—Supports the policy decision function by acting as the source for validating roaming and the International Mobile Equipment Identity (IMEI)
- Proxy-Call Session Control Function (P-CSCF)—The SBC performs this function at the HPLMN edge.

Interconnections between S8HR components are displayed below.



Key S8HR features include:

- IMS services are home-routed using a "well known" IMS Access Point Name (APN) via the S8 interface, similar to other data roaming traffic. Signaling and media use the same "well known" IMS APN established with the HPLMN, each with a specific QoS Class Identifier (QCI).
- The HPLMN controls call routing and per service logic.
- The IMS UNI is provided directly between UE and the HPLMN over the S8 interface. (S8HR does not require an IMS network-to-network interface (NNI) between the VPLMN and HPLMN.)
- The VPLMN is not service aware but it is QoS/APN aware.
- The architectures uses the Policy and Charging Control (PCC) framework of the HPLMN. QoS rules are generated in the HPLMN and enforced by the VPLMN per the carriers' roaming agreement.

Within the context of S8HR roaming support, the SBC is responsible for:

- Providing VPLMN identity to IMS entities in HPLMN
  - The HPLMN IMS entities need to be aware of the ID of the visited PLMN at session set up to populate the charging records.
  - During the registration, the S-CSCF and TAS need to be made aware about the ID of the visited PLMN to enable the HPLMN to subsequently perform roaming registration restriction or communication barring supplementary services.
  - To handle a non-UE detectable IMS emergency session establishment, the P-CSCF needs to be aware of the MCC of the VPLMN from where the UE initiates the IMS session. This information is needed at the P-CSCF at every IMS session establishment.
- Enforcing encryption procedures and requirements
  - The HPLMN must ensure that the IMS layer signaling and media confidentiality protection can be de-activated to enable the VPLMN to meet local regulatory requirement.
  - The SBC shall force NULL encryption for roaming users as part of the sec-agree negotiating mechanism based on an Mobile Network Code (MNC) and Mobile Country Code (MCC) match configured on the SBC.

  - When Sec-Agree (IMS-AKA) and S8HR are both enabled, the SBC triggers an AAR upon 1st unsecured SIP REGISTER to determine if the roaming network does not need encryption.

- Supporting Emergency calls
  Since there is no NNI over which typical authentication can be performed, the SBC enables the network to skip the IMS level authentication with sufficient confidence based on the fact that the UE has been successfully authenticated at EPS level. The identities number retrieved via the PCRF serves as a "soft" proof with which IMS proceeds without authenticating the UE at the IMS level.

S8HR Configuration and Configuration Considerations

To enable S8HR roaming, you create an S8HR profile and apply it to the applicable interface. Refer to the detail on profile configuration within this document.

S8HR roaming, however, is dependent on additional SBC configuration for its functionality:

- You must have enabled the VoLTE feature flag.
  sip-interface, **sip-ims-feature** enabled
- You must have enabled the Sec-Agree feature.
  sip-interface, **sec-agree-feature** enabled and

  sec-agree-pref is configured as **3gppIpsec**
- You must have disabled the Provision Signaling Flow feature.
  media-manager, **ext-policy-server**, **provision-signalingflow** disabled.
- You must configure a subscription to PLMN changes.
  media-manager, **ext-policy-server**, **specific-action-subscription** set to **plmn-change**.
- You must configure a signaling flow subscription when you need to support an Abort-Session-Request (ASR) via the PCRF to close a session when a user returns to their home-network from a roaming network. This allows the system to de-register and clear the registration cache of the user. Upon receiving a new register request from user, the system performs a new Rx dip.
  media-manager, **ext-policy-server**, **specific-action-sig-flow-subscription** set to **release-of-bearer**
- You must enable network management controls on the realm.
  media-manager, **realm-config**, **net-management-control** enabled.
- You must have configured an external policy server and applied it to a SIP interface.
  media-manager, **ext-policy-server** configured and applied to the **sip-interface**:
  - **reserve-incomplete** set to **enabled** or **origin-realm-only**.
  - **optimize-aar** disabled
  - **asynchronous-mode** disabled

If you intend to accept unregistered emergency calls:

- Reject Unregistered priority calls feature is disabled
  sip-interface, **options**, **reject-unreg-priority-calls** disabled and

  sip-interface, **send-380-response** set to null
- You must have disabled the disallow priority calls feature.
  sip-interface, **options**, **disallow-priority-calls** disabled

Reporting on S8HR Operation

The **show sipd register** command includes cached information, including the MNC/MCC/IMSI/IMEI/MSISDN for each user, and specifies if the user is an emergency roaming contact.

```
ORACLE# show sipd register
```

In addition, the **show sipd status** includes the following S8HR-related statistics:

- Forward User PVNI
- Forward Default PVNI
- Encrypt Disabled
- S8HR Emgy Reg 200
- S8HR Emgy Reg 403
- S8HR Emgy Inv
- S8HR Emgy Inv 403

```
ORACLE# show sipd status
```

VPLMN-ID Management Support

The Oracle Communications Session Border Controller (SBC) performs tasks to ensure the HPLMN IMS entities are aware of the VPLMN-ID at session set up for multiple purposes, including populating the charging records. 3GPP TR 23.749 covers this process

To support connectivity, the home network determines the serving PLMN of the UE using a procedure where the P-CSCF (SBC) gets the current location of the UE during registration and maintains it by establishing a PLMN-ID via AAR/RAR exchanges with the PCRF. The P-CSCF forwards this PLMN-ID information in the SIP REGISTER request in the **P-Visited-Network-ID** header, in addition to other SIP transactions.

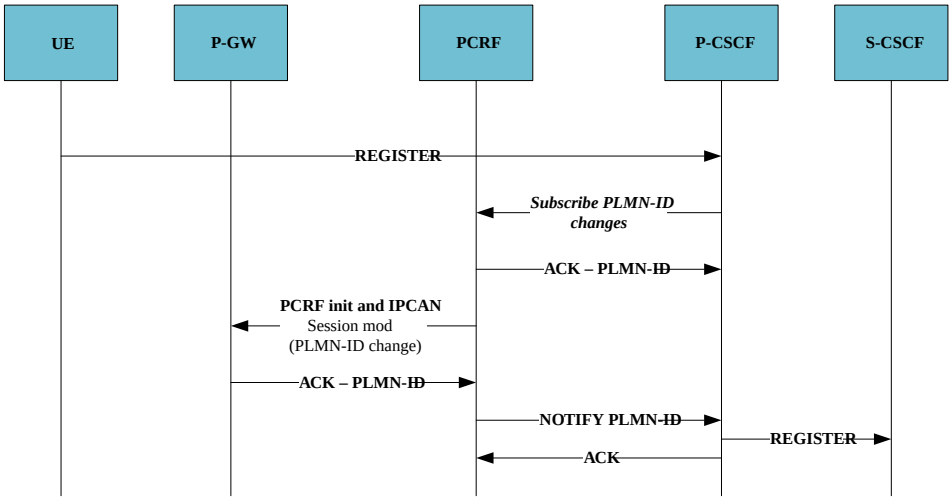
Examples of other SIP message that use this roaming location information from the P-CSCF include:

- The P-CSCF creates and inserts the PLMN-ID into any INVITE coming from the UE.
- The P-CSCF can include the **P-Access-Network-Info** and **P-Visited-Network-ID** header in the SIP Register before forwarding.
- To handle non UE detectable IMS emergency session establishment the P-CSCF needs to be aware of the MCC of the VPLMN from where the UE initiates IMS session. This information is needed at the P-CSCF for every IMS session.

PLMN-ID Info Subscription Case

UE registration also triggers the P-CSCF subscription to the PLMN-ID changes via the PCRF. During the subscription, the PCRF retrieves the PLMN-ID from the PGW and provides the information to the P-CSCF using a NOTIFY message.

During this exchange, the subscription from the P-CSCF can trigger a subscription by the PCRF to the P-GW, if one is not already in place. The subscription sequence results in the P-CSCF receiving MCC and MNC information to be used for the PLMN-ID, which it then stores for use with the subsequent REGISTER and other sessions.



As long as the registration is in effect, the PCRF sends the P-CSCF NOTIFY messages when the PLMN-ID changes. Termination of the registration, by any means, triggers the systems to terminate the subscription.

PLMN-ID Info Registration Scenarios

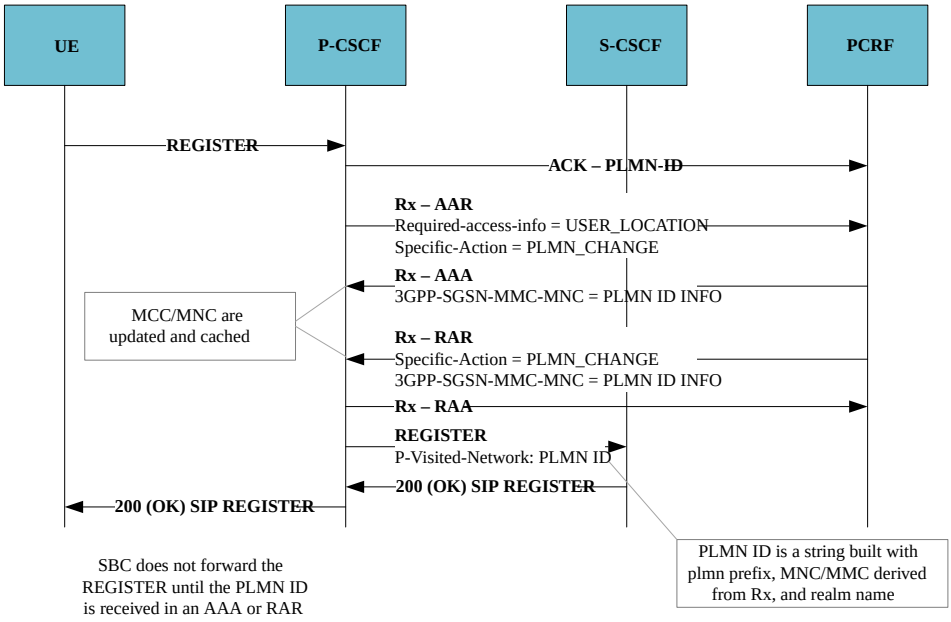
During the registration, the S-CSCF and Telecommunication Application Server (TAS) need to know the ID of the visited PLMN for charging purposes and to enable the HPLMN to subsequently perform roaming registration restriction or communication barring supplementary services. The P-CSCF gets this information by subscribing to the PLMN Change via the PCRF.

Initial SIP REGISTRATION with PLMN-ID information is provided in the AAR/AAA exchange. This is similar to and happens in conjunction with the subscription logic above. Registration scenarios, however, also include applicable updates via RAR/RAA exchanges. The scenario assumes that both the AAA and RAR arrive within the holding period. The PLMN ID information is passed on via **P-Visited- Network-ID** header.

Ibn this case, key AVPs include:

- **Required-Access-Info**—The USER_LOCATION value holds the PLMN-ID for use by the S-CSCF.
- **Specific-Action**—The PLMN_CHANGE value informs the components of the action.
- **3GPP-SGSN-MCC-MNC**— Provides information used for **P-Visited-Network-ID** and queues the system to reset the **register-hold-for-plmn-info timer**.

The typical initial registration scenario includes both AAR and RAR exchanges occurring within the S8HR profile's hold time.



Additional scenarios, with deviations from the basic registration scenario include:

- PLMN-ID Info received by AAA and hold time expired—If the profile's hold time expires before, the SBC proceeds without the RAR/RAA exchange, creating the PLMN-ID with information determined during the AAA/AAR exchange and forwarding the REGISTER.
- Hold time expired, default PLMN-ID Info is used—This scenario differs from a typical registration in that the SBC does not receive or have any PLMN-ID information prior to the hold time expiry. The SBC proceeds, however, using default PLMN information you configure on the access side of sip interface (**session-router**, **sip-interface**, **network-id**. The SBC adds this default PLMN information to the SIP REGISTER message in the **P-Visited-Network-Id** header and forwards it to the core.
- PLMN-ID Info re-registration—The P-CSCF supports re-registration, triggered by the UE and avoiding the any Rx exchange, using cached location information and relying on the subscription to update that information if it changes. If the NPLI feature is enabled and the re-registration occurs after more than half the registration time has expired, however, the SBC proceeds with the Rx exchange. In addition, the SBC renews its subscription to PLMN information changes. Furthermore, the SBC sends out the REGISTER with both PVNI (for S8HR) and PNNI (for NPLI) headers.
- PLMN-ID Info de-registration Case—The SBC performs standard de-registration procedures. This happens the UE presents a re-registration with an expires time of zero or the registration reaches its previously set expires time. The SBC also un-subscribes from the PLMN changes using STR exchanges with the PCRF.

Sec-Agree for S8HR Roaming Support

The Oracle Communications Session Border Controller (SBC) supports security agreement negotiation as described in RFC3329. For S8HR, the SBC extends upon sec-agree to disable encryption for trusted roaming networks. You specify which networks are trusted with the **s8hr-profile** that you apply to an interface. The SBC enforces your configuration by offering **NULL** as the only encryption algorithm for trusted networks.

The SBC determines whether a network is trusted based on its VPLMN-ID. To determine this, the SBC queries the PCRF. The subsequent Rx exchange provides the VPLMN-ID. The SBC triggers this Rx exchange when it receives the unsecured 1st REGISTER. (PLMN info is required from the PCRF to determine whether or not the UE is roaming.)

> **Note:**
>
> The SBC also performs functions, including NPLI subscription and PANI header population upon this 1st REGISTER to minimize the number of Rx queries.

System process:

1. Determine if UE is roaming with S8HR by comparing the local configured MNC/MCC with the roaming MNC/MCC received via Rx and cached for this contact. If they are different, the UE is roaming with S8HR.
2. Check if the **sip-interface** is configured to disable encryption for all roaming partners. The **encrypt-disabled-mnc-mcc** parameter takes a list of roaming network MNC/MCC the S8HR profile defines to disable the encryption.
3. Check if roaming network (core) is one of the configured partners to disable the encryption. SBC compares the roaming MNC/MCC to the encryption disable network list configured in the S8HR profile.
4. If applicable, disable encryption for this roaming network (core). The SBC caches the “encryptDisabled” flag for the contact and sets all subsequent SIP request message security headers with the **ealg** parameter set to **NULL**.

MNC/MCC received from the PCRF are presented as unsigned integer data type. They are converted to a string in MNC/MCC format where 0-3 characters are MNC and 4-6 characters are MCC. The SBC uses this same format for the **encrypt-disabled-mnc-mcc** parameter.

> **Note:**
>
> The ealg header presents only when sec-agree chooses “ipsec3gpp” as the security mechanism.

The uses the Rx exchange to determine the roaming network and whether or not it is on the encryption disable list. If so, encryption is disabled in SIP message header.

Additional considerations include:

- The HPLMN must be able to ensure that the IMS layer signaling and media confidentiality protection is not activated in order to enable the VPLMN to meet the local regulatory requirements.
- If the HPMN uses IMS layer signaling and media confidentiality protection on its network (e.g. for the HPMN’s own subscribers, for inbound roaming LBO IMS subscribers), then based on the customer location and IMS Roaming agreement type, this protection may have to be deactivated in the HPMN.

Emergency Service Support

The Oracle Communications Session Border Controller (SBC) supports emergency calls from roaming UEs over S8HR.

When S8HR roaming is used for VoLTE there is no IMS roaming NNI between the VPLMN and the HPLMN. It is, therefore, not possible to use standard methods to authenticate the UE in the IMS Domain. To support Emergency calls and meet regulatory requirements, the SBC supports multiple variations on S8HR emergency call support, including:

- Emergency Registration
- Emergency Re-Registration
- Emergency De-Registration
- Emergency Registration Based on Roaming Status
- Emergency INVITE from Un-Registered User
- Emergency INVITE after emergency registration

Supporting Emergency Registration

Basic Emergency Registration

When the SBC receives an initial un-secured REGISTER SIP message for an IMS emergency registration, it requests the the Evolved Packet Core (EPC)-level identities available for that IP-Connectivity Access Network (IP-CAN) session from the PCRF, including:

- Mobile Station International Subscriber Directory Number (MSISDN)
- International Mobile Subscriber Identity (IMSI)
- International Mobile Equipment Identity IMEI and Software Version (SV)

To do so, the SBC initiates an Rx Diameter session with the PCRF using an AA-Request (AAR) command. The AAR includes three AVPs:

- **Framed-IP-Address** (or Framed-IPV6-Prefix) AVP with value set to the IP in the REGISTER Contact header field.
- **AF-Requested-Data** AVP with value set to “EPC-level identities required(0)”, indicating that the AF requests the PCRF to provide the EPC-level identities.
- **Service-URN** AVP with value set to “sos”, indicating this is an emergency registration.

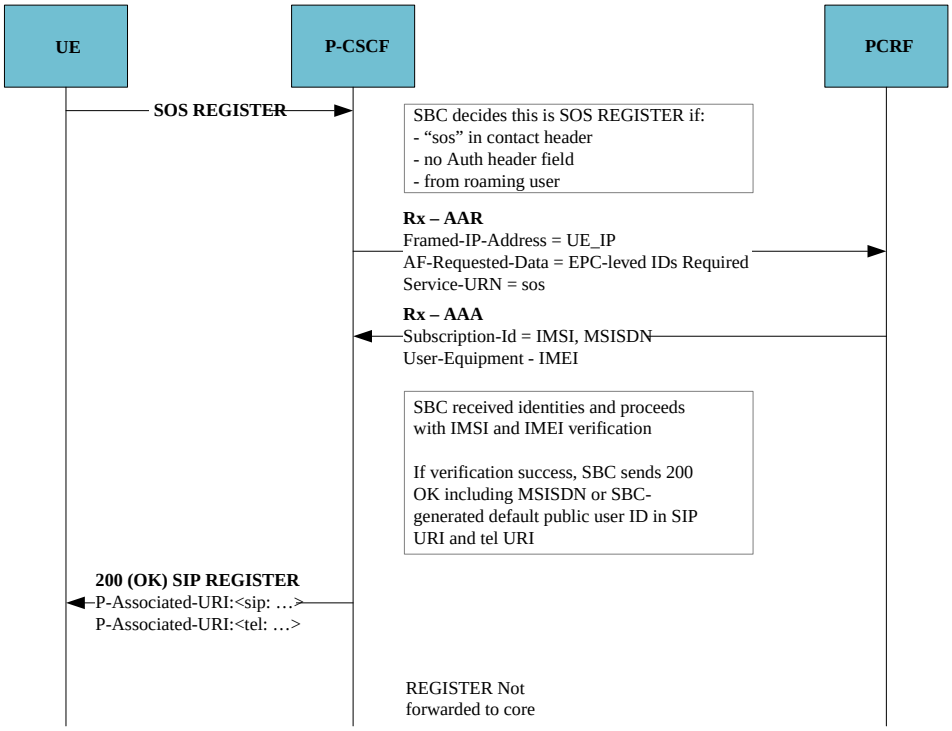
When the **Service-URN** AVP indicates that the AF session relates to emergency traffic and the **AF-Requested-Data** AVP indicates "EPC-level Identities required", the PCRF provides the requested identity information for the IPCAN session within the **Subscription-Id** AVP(s) and/or the IMEI(SV) within the **User-Equipment-Info** AVP in the AAA.

The PCRF provides the UE public identities in the AAA message. The SBC looks for the IMSI and/or the MSISDN in the **Subscription-Id** group AVPs, and IMEI in the **User-Equipment-Info** group AVPs:

- **Subscription-Id** AVP
  - MSISDN: value of Subscription-Id-Data with Subscription-Id-Type = END_USER_E164 (0)
  - IMSI: value of Subscription-Id-Data with Subscription-Id-Type = END_USER_IMSI (1)
- **User-Equipment-Info** AVP
  - IMEI: value of **User-Equipment-Info-Value** with **User-Equipment-Info-Type** = IMEISV (0)

The SBC does not forward this REGISTER request. Instead, it stores and verifies the received IMSI or IMEI. If verification succeeds, it sends a 200 (OK) to the UE and the UE is successfully registered. The SBC keeps the entry in its registration cache and allows the associated contact address to use emergency service only.

If identity verification fails, the SBC sends a 403 (Forbidden) with configurable reason back to the UE. Successful registration is depicted below.



> **Note:**
>
> For the emergency roaming user, the SBC does not support the 420 response with sec-agree. This means the 1st un-secured REGISTER is handled with 200 or 403 response from SBC. There is no sec-agree or secured REGISTER involved. Refer to 3GPP TS 24.299 5.2.10.5 for more details.

Emergency re-Registration

When the SBC a re-registration from a S8HR roaming user, it refreshes its registration cache for this contact. It does not forward the REGISTER to the core. Instead, it sends a 200 (OK) response including the P-Associated-URI header filed containing the list of implicitly registered public user identities saved in its cache for this contact. There is no Rx exchange with PCRF triggered in this case.

However, when the UE re-registers with a different contact for the emergency registrations, the SBC deletes (expires) the original contact for the emergency registration and add the newly registered contact. The SBC the initiates an Rx exchange with the PCRF to retrieve EPC-level identities, like the initial registration.

Emergency de-Registration

Like a normal emergency contact, once the UE registers a public user identity and an associated contact address via emergency registration, it cannot de-register the associated user identity and contact address. The SBC removes the registration when it expires.

If the SBC receives a de-register from the S8HR roaming user for the emergency registration, SBC will NOT forward the deregister to the registrar. SBC will send response 200 (OK) to UE.

If it receives a de-register from the roaming user for the emergency registration, the SBC does not forward the de-register to the registrar. Instead, it searches for the registration in its cache.

If it finds the registration, the SBC sends a 200 (OK) response to the UE and removes the registration entry from its cache.

Emergency Registrations Based on Roaming Status

The SBC IMEI/IMSI validation for Emergency registration from S8HR roaming user utilizes the **epc-id-required** parameter to determine its authentication behavior based on input from the REGISTER and the PCRF.

Using **epc-id-required** within the context of emergency registration feature for S8HR Roaming station, the SBC:

- Provides the VPLMN identity to IMS entities in the HPLMN:
  - During the registration, the S-CSCF and TAS need to be made aware about the ID of the visited PLMN e.g. for charging purposes and to enable the HPLMN to subsequently perform roaming registration restriction or communication barring supplementary services.
  - To handle non UE detectable IMS emergency session establishment the P-CSCF needs to be aware of the MCC of the VPLMN from where the UE initiates IMS session. This information is needed at the P-CSCF at every IMS session establishment.
  - The HPLMN IMS entities need to be aware of the ID of the visited PLMN at session set up in order to populate the charging records.
- Supports the HPLMN, which must ensure that IMS layer signaling and media confidentiality protection is not activated in order to enable the VPLMN to meet the local regulatory requirements.
  Supports the HPLMN, if it uses the IMS layer signaling and media confidentiality protection on its network (e.g. for the HPMN's own subscribers, for inbound roaming LBO IMS subscribers). Based on the customer location and IMS Roaming agreement type, this protection may be deactivated in the HPMN.
- Supports Emergency calls regulatory requirement when you are using S8HR roaming for VoLTE and there is no IMS roaming NNI between the VPLMN and the HPLMN. This would mean it is not possible to use existing methods to authenticate the UE in the IMS Domain.

An emergency REGISTER from an S8HR roaming user has following characteristics:

- Contains an "sos" SIP URI parameter in the Contact header field (3GPP TS 24.299 describes the syntax)
- Does not contain an Authorization header field
- The Domain name retrieved from the Request-URI is different from the SBC local network (**sip-interface**, **uri-fqdn-domain**).

The SBC retrieves and caches the IMSI, and/or IMEI, and/or MSISDN from AAA message, and performs the following identity verification tasks:

- If the IMSI derived from the public user identity in the TO header of the REGISTER is the same as the IMSI received from the PCRF, the SBC marks the IMSI verification as successful.
- If the IMEI obtained from instance ID conveyed in the Contact header field of the REGISTER is the same as the IMEI received from the PCRF, the SBC marks the IMSI verification as successful.

The SBC performs this authentication dependent on your setting for the **epc-id-required** parameter. If you set the **epc-id-required** parameter to **IMSI_STRICT**, the SBC behaves as follows:

- If the SBC receives both the IMSI and IMEI from the PCRF, it verifies both of them. If either of those verifications fail, the SBC rejects the REGISTER request by returning a 403 (Forbidden). Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC receives the IMSI only from the PCRF, it verifies the IMSI. If the IMSI verification fails, the SBC rejects the REGISTER request by returning a 403 (Forbidden). Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC receives the IMEI only from the PCRF, it rejects the REGISTER request by returning a 403 (Forbidden).
- If the SBC does not receive both the IMEI and IMSI from the PCRF, it rejects the REGISTER request and sends a 403 (Forbidden) response.

> **Note:**
>
> Oracle recommends the **IMSI_STRICT** setting for environments that require an IMSI in the end user's device as a prerequisite to setting up an emergency call.

If you set the **epc-id-required** parameter to **IMSI_LOOSELY**, the SBC behaves as follows:

- If the SBC receives both the IMSI and IMEI from the PCRF, it verifies both the IMSI and the IMEI. If either verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC receives only the IMSI from the PCRF, it verifies the IMSI. If the verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC receives only the IMEI from the PCRF, it verifies the IMEI. If the verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC does not receive both the IMSI and the IMEI, it rejects the REGISTER request and sends a 403 (Forbidden) response.

If you set the **epc-id-required** parameter to **IMSI_AND_IMEI_LOOSELY**, the SBC behaves as follows:

- If the SBC receives both the IMSI and IMEI from the PCRF, it verifies both IMSI and IMEI. If either verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC receives only the IMSI from the PCRF, it verifies the IMSI. If the verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC receives only the IMEI from the PCRF, it verifies the IMEI. If the verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC does not receive both the IMSI and IMEI, it accepts the REGISTER request and sends a 200 (OK)

If you set the **epc-id-required** parameter to **IMSI_OR_IMEI**, which is the default, the SBC behaves as follows:

- If the SBC receives both the IMSI and IMEI from the PCRF, it verifies both the IMSI and the IMEI. If either of the verifications succeed, it accepts the REGISTER request and sends a 200 (OK). Otherwise, it rejects the REGISTER request and sends a 403 (Forbidden).
- If the SBC receives only the IMSI from the PCRF, it verifies the IMSI. If the verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC receives only the IMEI from the PCRF, it verifies the IMEI. If the verification fails, the SBC rejects the REGISTER request and sends a 403 (Forbidden) response. Otherwise, it accepts the REGISTER request and sends a 200 (OK).
- If the SBC does not receive both the IMSI and IMEI, it rejects the REGISTER request and sends a 403 (Forbidden) response.

> **Note:**
>
> If there is no IMSI and/or IMEI in the TO or Contact header field of the REGISTER, the SBC marks the corresponding (IMSI and/or IMEI) validation as failed. It does this even if the PCRF returns corresponding (IMSI and/or IMEI) values. The SBC behavior is based on the **epc-id-required** setting in the applicable **s8hr-profile**.

When identity verification for a roaming user succeeds, the SBC begins to prepare the 200 (OK) response to UE. It uses the MSISDN to generate the following URIs:

- A SIP URI with user=phone for the MSISDN—This is the default public user identity and becomes the first URI in the list of public user identities.
- A tel URI from the MSISDN

The SBC uses these URIs in the P-Associated-URI header fields, so the response contains the list of implicitly registered public user identities bound to the contact.

> **Note:**
>
> If MSISDN is not received from PCRF, SBC will generate a temporary public user identity from the received IMSI. This temporary public user id is used as MSISDN to create URIs described above.

When identity verification for a roaming user fails, the SBC begins to prepare the 403 (Forbidden) response to UE. The 403 (Forbidden) includes:

- A Content-Disposition header field with a disposition type “render” value and a “handling” header filed parameter with an “optional” value.
- Content-Type header field with the value set to associated MIME type of the 3GPP IM CN subsystem XML body. This is the same XML as when the SBC responds with a 380.
- A P-Asserted-Identity header field set to the value of the SBC SIP URI.
- 3GPP IM CN subsystem XML body containing an <ims-3gpp> element with the "version" attribute set to "1" and with an <alternative-service> child element, set to the parameters of the alternative service:
  - A <type> child element, set to "emergency" to indicate that it was an emergency call
  - A <reason> child element, set to a configurable reason you set with the **emergency-reject-reason** in the applicable **s8hr-profile**

– An <action> child element, set to "anonymous-emergencycall"

> **Note:**
>
> The existing **registration-interval** configuration on the **sip-interface** specifies the maximum expires allowed for the registration.

Configure the Registration Behavior

The syntax for configuring the **epc-id-required** parameter to the IMSI_OR_IMEI value is:

```
ORACLE(s8hr-profile)# epc-id-required IMSI_OR_IMEI
```

Emergency INVITEs

Emergency INVITE from an Un-Registered User

If the SBC receives an emergency INVITE from an un-registered roaming user, it initiates an Rx exchange with the PCRF requesting EPC-level identities (MSISDN, IMEI, IMSI). This Rx message sequence is the same as for the Emergency REGISTER scenario.

The SBC compares the IMEI received from the PCRF with the one in the INVITE. If there is a match, it modifies the SIP INVITE to include the UE's public identities and then forwards it to the core. It sends the corresponding responses from core back to UE when it receives them.

Specifically, if the IMEI obtained from “+sip.instance” parameter conveyed in the Contact header field of INVITE is the same as the IMEI received from PCRF, the IMEI validation is successful. The SBC modifies the INVITE as following and forwards it to the core:

- Remove any P-Preferred-Identity header field from the INVITE.
- If the SBC retrieves an MSISDN, it inserts a P-Asserted-Identity header field set to “tel:<MSISDN>” in the INVITE.
- If the SBC retrieves an IMSI, it inserts a P-Asserted-Identity header field set to a temporary public user identity derived in the INVITE.

If the IMEI verification fails, and you have configured the SBC to reject in this case, it replies to the UE with a **403 (forbidden)**.

> **Note:**
>
> If identity verification failed but the SBC is not configured to reject, it modifies the INVITE forwards it to the core similarly to the IMEI verification success case.

Emergency INVITE from a Registered User

The SBC determines the registration associated with the session initiator as well as the next possible route normally for a registered user that happens to be roaming. The emergency registration is valid for emergency sessions only.

Creating the PAI from Rx IMSI information for Emergency Calls

You can configure the SBC to build the host part of a P-Asserted-Identity header based solely on the EPC level IMSI information provided by the PCRF within Rx information, if required. This configuration applies to S8HR deployments when the system handles a local or roaming emergency call. The system performs this function regardless of whether the emergency call comes from an unregistered, emergency registered or a non-emergency registered endpoint. This feature establishes compliance with the applicable requirement within 3GPP 24.229.

When generating public user identity information by default, the SBC first tries to verify any public user identity information received from the PCRF by checking whether the IMSI is consistent with the currently configured **operator-config-local-mccmnc**. If so, the SBC recognizes the caller as being a local subscriber and builds the host part of the PAI based solely on the Rx information. If this is not true, however, the SBC proceeds with creating a PAI with public user identity using the PPI, or the PAI, or the FROM header host part from the incoming request. The system includes this PAI host in the applicable outgoing request.

If the information in the incoming INVITE includes EPC level IMSI that is not consistent with your configuration, and you need to comply with the applicable requirements within 3GPP 24.229, you can set the **emergency-pai-host-from-rx** option in the **ext-policy-server** configuration that points to the PCRF. When you set this option, the SBC builds the outgoing PAI header based only on information received from PCRF, ignoring locally configured and incoming INVITE data.

The syntax below shows how to set this option.

```
ORACLE(ext-policy-server)#options +emergency-pai-host-from-rx
```

> **Note:**
>
> If you type the option without the plus sign, you overwrite any previously configured options. To append the new option to the options list, prepend the new option with a plus sign as shown in the previous example.

When building the IMSI identity, the egress INVITE has a P-Asserted-Id containing a temporary public user identity derived from the IMSI, as defined in 3GPP TS 23.003. The format is as follows.

```
sip:<IMSI>@ims.mnc<MNC>.mcc<MCC>.example.com
```

When configured with this option, the SBC follows the steps below to build the MNC:

1. If the IMSI from the PCRF is consistent with the currently configured **operator-config-local-mccmnc**, then it's a local subscriber and the SBC builds the host part of the PAI based solely on the Rx information.
2. If the call is an emergency call and the SBC receives EPC level IMSI from the PCRF, then SBC builds the host part of the PAI based solely on the Rx information and regardless of user status (local or roaming).
3. If the IMSI from the INVITE was not consistent and the SBC did not receive EPC level IMSI from the PCRF, the user is a roaming subscriber. In this case, the SBC uses the PPI, or the PAI, or the from header host part, if it is in the format ims.mncXXX.mccXXX.example.com. The selection precedence is PPI, then PAI, then FROM header.

> **Note:**
>
> When you set the **emergency-pai-host-from-rx** option, the system does not perform steps 3 and 4.

4. If none of the above, the SBC builds the PAI using the locally configured mnc/mcc values.

Although the SBC replicates this configuration across an HA pair, it does not replicate the PAI headers. The feature works on current data received from the PCRF, which the SBC uses to create the PAI header for the INVITE. But the system does not synchronize this data with the standby.

MNC Digit Constraints

In addition to the configuring the SBC to use the PAI host from the Rx information, you need to consider whether you need the system to parse the IMSI for a 3 digit MNC instead of 2. This allows the system to build the PAI properly.

An IMSI presented by the PCRF may be 14 or 15 digits, depending on whether the MNC within is 2 digits (European standard) or 3 digits (American standard). The IMSI begins with the mobile country code (MCC), which is three digits, followed by the mobile network code. If your environment uses 3 digit MNC values, set this option so the SBC can parse the MNC properly.

To configure this behavior, set the **emergency-pai-host-three-digit-mnc** options in the same external policy server configuration for which you set the PAI from Rx option. The syntax below shows how to set this option.

```
ORACLE(ext-policy-server)#options +emergency-pai-host-three-digit-mnc
```

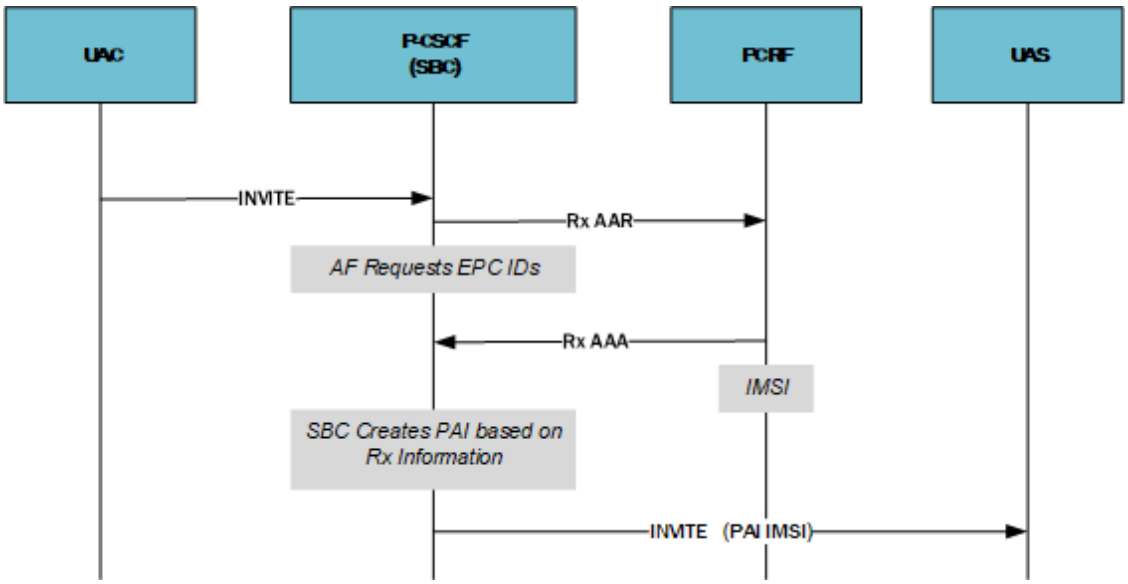
If you do not set this option, the SBC parses and extracts only the applicable 2 digits as the MNC.

Applicable Call Flows

The call flows below depict the signaling when an emergency call comes from unregistered, emergency registered and non-emergency registered users. In all of these cases, the SBC, configured with this option, constructs the outgoing PAI header based on information received from PCRF and not from the local configured or incoming INVITE data.

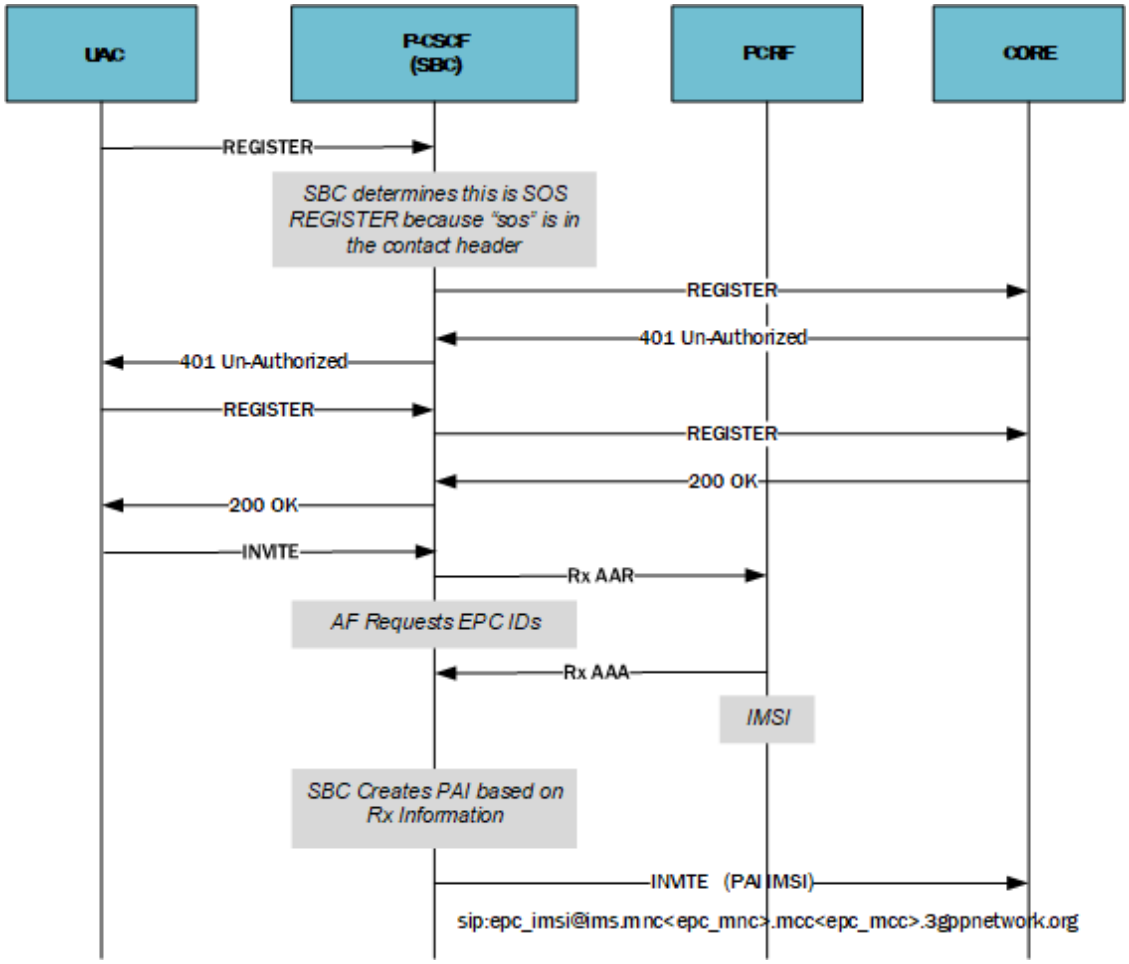
The first call flow depicts an emergency INVITE received from an unregistered user:

1. The SBC receives an emergency INVITE from an unregistered user.
2. SBC sends the AAR towards the PCRF, including the AF-Requested-Data AVP.
3. The PCRF sends the AAA with the IMSI in the Subscription-Id grouped AVP.
4. SBC constructs the PAI header appropriately and sends the Emergency INVITE to the UAS.



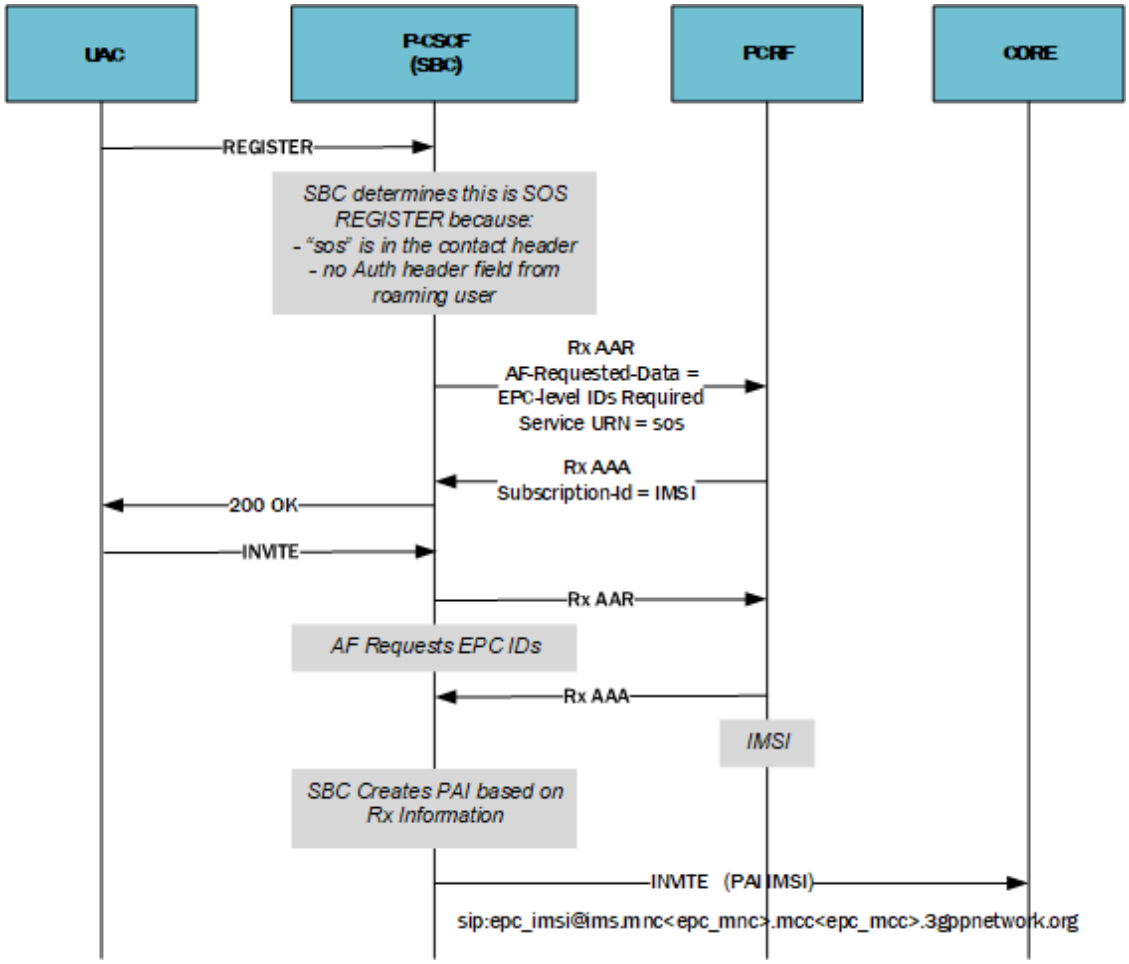
The second call flow depicts an emergency INVITE received from an registered user:

1. The end user performs an emergency registration.
2. The SBC receives an emergency INVITE from this newly registered user.
3. SBC sends the AAR towards the PCRF, including the AF-Requested-Data AVP.
4. The PCRF sends the AAA with the IMSI in the Subscription-Id grouped AVP.
5. SBC constructs the PAI header appropriately and sends the Emergency INVITE to the UAS.



The third call flow depicts an emergency INVITE received from an registered user that is currently roaming:

1. The end user performs an emergency registration.
2. The SBC receives an emergency INVITE from this registered, roaming user.
3. SBC sends the AAR towards the PCRF, including the AF-Requested-Data AVP.
4. The PCRF sends the AAA with the IMSI in the Subscription-Id grouped AVP.
5. SBC constructs the PAI header appropriately and sends the Emergency INVITE to the UAS.



Use of the AF-Requested-Data AVP to Obtain EPC Identity for Emergency Calls

You can configure the Oracle Communications Session Border Controller (SBC) to get EPC-level identities from the PCRF through the Rx interface. When triggered, the feature issues an AAR that includes the AF-Requested-Data AVP set to the value, EPC-level identities required, as described in TS 29.214, during emergency calls. Within this context, the SBC is the Application Function (AF). Target deployment scenarios include supporting emergency call for native, multi-SIM subscribers. These subscribers may use multiple terminals with the same IMS-level MSISDN, such as a smart phone and a wearable device, that are in different locations. You can use this feature to populate an emergency INVITE with the terminal identity retrieved by the EPC.

The SBC performs this procedure on the initial INVITE of an emergency call. Applicable caller status includes unregistered, emergency registered and non-emergency registered. All forms of emergency calls that are supported by the SBC, including SOS URN, MPs, NMC configuration, and RPH calls trigger this functionality.

This feature is disabled by default. Applicable configuration includes the following parameters on the **ext-policy-server** of the applicable access realm:

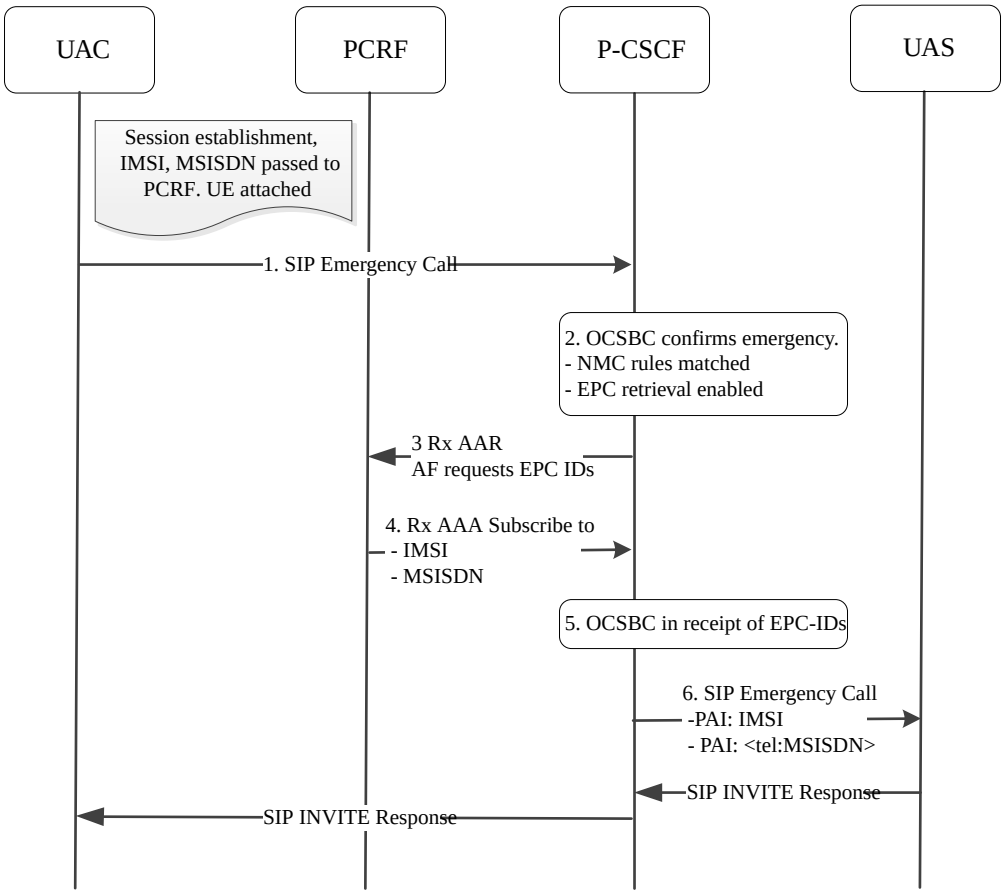
- Enable the **emergency-epc-level-identities** parameter to enable the feature.

- Enable the **use-epc-level-msisdn** parameter to specify that the SBC should insert an EPC-level MSISDN into the PAI of the egress INVITE. This parameter is only relevant if you have also enabled **emergency-epc-level-identities**.
- Set the **operator-config-local-mcc-mnc** parameter to specify the MNC digit.

The PCRF can reply to the SBC with an AAA that includes either no identities, the IMSI identity, the MSISDN identity or both. It may also include the IMEI identity, but this is always ignored. The identities arrive within Subscription-Id AVPs objects, as specified in 3GPP TS 29.214. Having received the AAA, the SBC constructs the components of the INVITE as follows:

- When building the IMSI identity, the egress INVITE has a P-Asserted-Id containing a temporary public user identity derived from the IMSI, as defined in 3GPP TS 23.003. The format is sip:<IMSI>@ims.mnc<MNC>.mcc<MCC>.3gppnetwork.org. The SBC builds the MNC digit based on:
  - If the IMSI from the PCRF is consistent with the currently configured **operator-config-local-mcc-mnc**, then it's a local subscriber and the SBC builds the host part of the PAI accordingly based solely on the Rx information.
  - The SBC includes an additional behavior, based on configuration. If the call is an emergency call and the SBC receives epc level IMSI from the PCRF, the SBC builds the host part of the PAI based solely on the received Rx information, regardless of user status (local or roaming).
    See "Creating the PAI from Rx IMSI information for Emergency Calls" for configuration and detailed behavior.
  - If the IMSI from the INVITE was not consistent and the SBC did not receive EPC level IMSI from the PCRF, the user is a roaming subscriber. In this case, the SBC uses the PPI, or the PAI, or the from header host part, if it is in the format ims.mncXXX.mccXXX.3gppnetwork.org. The selection precedence is PPI, then PAI, then FROM header.
  - If none of the above, the SBC builds the PAI using the locally configured mnc/mcc values.
- When building the MSISDN identity, if you have enabled **use-epc-level-msisdn**, the SBC uses it as the P-Asserted-Id in the tel-uri format.
- When building both MSISDN and the IMSI identities, both are processed independently as per the guidelines above.
- The IMEI identity is always ignored.

The basic sequence resulting in the retrieval includes the 6 steps shown in the call flow below.



The SBC modifies the INVITE as following and forwards it to the core.

- If an MSISDN is retrieved, insert or replace a P-Asserted-Identity header field in the INVITE set to "tel:<MSISDN>". See RFC 3325 for header details and format.
- If an IMSI is retrieved, insert/replace a P-Asserted-Identity header field in the INVITE set to a temporary public user identity derived from the IMSI. See 3GPP TS 23.003 for derivation and format details.

The SBC uses additional conditions for processing these EPC-level identities, depending on whether or not the user is registered. These conditions are presented in sections below.

Emergency Call Hand off

Emergency call handoffs may occur after the SBC has processed the initial INVITE. At this point, the SBC has already sent the initial AAR with the AF-Requested-Data with EPC-Level identities required. The SBC only requests the EPC identities once. As such, the SBC does not request these identities again during a handoff, making the assumption the identities do not change.

If you configure the applicable **ext-policy-server** on the interfaces for CS call leg, then the SBC sends the Rx AAR with the Rx-Request-Type AVP set to UPDATE_REQUEST. This INVITE is not the same as the initial INVITE, so the SBC never includes an AF-Requested-Data AVP in these AARs.

Related Feature Dependencies

Other SBC configuration can impact the operation of this feature including the following:

- If you have enabled **asynchronous-mode** on the applicable **ext-policy-server**, the SBC does not wait for an AAA in response to an AAR. In this case, the AF-Requested-Data is

not applicable because the SBC does not send the AF-Requested-Data AVP in the Rx AAR to the PCRF.
Refer to Asynchronous SIP-Diameter Communication for additional detail.

- Keep in mind the inclusion of both SIP and TEL PAI types when you enable the **add-second-pai-for-emergency-calls** option in the **sip-config**.
- All **net-management-control** configurations that trigger this feature require:
  - The **type** parameter set to **priority**.
  - The **destination-identifier** parameter set to **urn:service:sos**.
  - The **rph-feature** parameter set to **enabled**.

EPC-Level ID Processing for Registered Users

The tables below show the results of the processing performed by the SBC to produce the applicable output. The first column shows what the SBC would send out if EPC identities were not retrieved. The top row shows the EPC identities the SBC receives in the AAR.

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured
- **use-epc-level-msisdn** not configured
- 2nd PAI Option (**add-second-pai-for-emergency-calls**) disabled

Outgoing INVITE	Only IMSI Received	Only MSISDN Received	IMSI and MSISDN Received
Only SIP PAI	Replace SIP PAI with IMSI PAI	SIP PAI as is	Replace SIP PAI with IMSI PAI
Only Tel PAI	Tel PAI as is	Tel PAI as is	Tel PAI as is
None	Add IMSI PAI	No Change	Add IMSI PAI

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured.
- **use-epc-level-msisdn** not configured.
- Second PAI Option (**add-second-pai-for-emergency-calls**) enabled.

Outgoing INVITE	Only IMSI Received	Only MSISDN Received	IMSI and MSISDN Received
Only SIP PAI	Replace SIP PAI with IMSI PAI	SIP PAI as is	Replace SIP PAI with IMSI PAI
Only Tel PAI	Add IMSI PAI Tel PAI as is	Tel PAI as is	Add IMSI PAI Tel PAI as is
SIP PAI and Tel PAI	Replace SIP PAI with IMSI PAI Tel PAI as is	SIP and TEL PAIs as are	Replace SIP PAI with IMSI PAI Tel PAI as is
None	Add IMSI PAI	No Change	Add IMSI PAI

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured.

- **use-epc-level-msisdn** configured.
- 2nd PAI Option (**add-second-pai-for-emergency-calls**) disabled

Outgoing INVITE	Only IMSI Received	Only MSISDN Received	IMSI and MSISDN Received
Only SIP PAI	Replace SIP PAI with IMSI PAI	SIP PAI as is	Replace SIP PAI with IMSI PAI
Only Tel PAI	Tel PAI as is	Replace Tel PAI with MSISDN PAI	Replace Tel PAI with MSISDN PAI
None	Add IMSI PAI	Add MSISDN PAI	Add IMSI PAI

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured.
- **use-epc-level-msisdn** configured.
- 2nd PAI Option (**add-second-pai-for-emergency-calls**) enabled.

Outgoing INVITE	Only IMSI Received	Only MSISDN Received	IMSI and MSISDN Received
Only SIP PAI	Replace SIP PAI with IMSI PAI	SIP PAI as is. Add MSISDN PAI	Replace SIP PAI with IMSI PAI Add MSISDN PAI
Only Tel PAI	Add IMSI PAI Tel PAI as is	Replace Tel PAI with MSISDN PAI	Replace SIP PAI with IMSI PAI Add IMSI PAI
SIP PAI and Tel PAI	Replace SIP PAI with IMSI PAI Tel PAI as is	SIP PAI as is. Replace Tel PAI with MSISDN PAI	Replace SIP PAI with IMSI PAI Replace Tel PAI with MSISDN PAI
None	Add IMSI PAI	Add MSISDN PAI	Add IMSI PAI Add MSISDN PAI

EPC-Level ID Processing for Unregistered Users

Some networks support IMS services for roaming users in deployments without IMS-level roaming interfaces. This section details EPC-level processing for these unregistered users.

The tables below specifies the results of the processing performed by the SBC to produce the applicable output. The first column shows what is received in the original INVITE. The top row shows the EPC identities the SBC receives in the AAR.

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured.
- **use-epc-level-msisdn** NOT configured.
- Second PAI Option (**add-second-pai-for-emergency-calls**) enabled.

Only IMSI	Only MSISDN	IMSI and MSISDN	None
Add IMSI as SIP PAI	None	Add IMSI as SIP PAI	Remove PPIs from incoming INVITE and forward to core

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured.
- **use-epc-level-msisdn** NOT configured.
- 2nd PAI Option (**add-second-pai-for-emergency-calls**) disabled.

Only IMSI	Only MSISDN	IMSI and MSISDN	None
Add IMSI as SIP PAI	None	Add IMSI as SIP PAI	Remove PPIs from incoming INVITE and forward to core

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured.
- **use-epc-level-msisdn** configured.
- 2nd PAI Option (**add-second-pai-for-emergency-calls**) disabled

Only IMSI	Only MSISDN	IMSI and MSISDN	None
Add IMSI as SIP PAI	Add MSISDN as TEL PAI	Add IMSI as SIP PAI	Remove PPIs from incoming INVITE and forward to core

Assume the SBC configuration list is activated:

- **emergency-epc-level-identities** configured.
- **use-epc-level-msisdn** configured.
- 2nd PAI Option (**add-second-pai-for-emergency-calls**) enabled.

Only IMSI	Only MSISDN	IMSI and MSISDN	None
Add IMSI as SIP PAI	Add MSISDN as TEL PAI	Add IMSI as SIP PAI ADD MSISDN as TEL PAI	Remove PPIs from incoming INVITE and forward to core

Per TS 24.229 section 5.2.10.2 point number 5, the SBC attempts to get EPC-level identities. The SBC must also strip any PPI. If it receives an MSISDN, the SBC inserts P-Asserted-Identity header field in the request set to a tel-URI carrying the MSISDN. If it receives an IMSI, the SBC inserts a P-Asserted-Identity header field in the request set to a temporary public user identity derived from the IMSI, as defined in 3GPP TS 23.003.

If the SBC receives no subscription-Id AVP, then it forwards the call by removing the P-Preferred-Identity header field, and keeps any PAIs as is. If it receives no AAA, the SBC behaves as if the **emergency-epc-level-identities** and **use-epc-level-msisdn** are not enabled.

Configuring AF-Requested EPC Identities

You enable the following parameters to enable specific components of Configuring AF-Requested EPC Identities. Refer to related configuration information to ensure you have considered and are implementing complementary or conflicting configuration properly for your deployment.

Use the following procedure to configure AF-Requested EPC Identities support.

1. Navigate to the **ext-policy-server** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# ext-policy-server
```

2. Enable the **emergency-epc-level-identities** parameter.

```
ACMEPACKET(ext-policy-server)#emergency-epc-level-identities enable
```

3. Enable the **use-epc-level-msisdn** parameter.

```
ACMEPACKET(ext-policy-server)#use-epc-level-msisdn enable
```

4. If the default is not acceptable, specify the MNC value and number of digits to use when creating the outgoing PAI using the **operator-config-local-mcc-mnc** parameter.

```
ACMEPACKET(ext-policy-server)#operator-config-local-mcc-mnc 23232
```

5. If applicable to your deployment, enable the **emergency-pai-host-from-rx** option.

```
ACMEPACKET(sip-config)#option +emergency-pai-host-from-rx
```

6. If applicable to your deployment, enable the **emergency-pai-host-three-digit-mnc** option.

```
ACMEPACKET(sip-config)#option +emergency-pai-host-three-digit-mnc
```

7. Use **done**, **exit**, **exit** to save the configuration.
8. Navigate to the **sip-config** configuration element.

```
ACMEPACKET# session-router
ACMEPACKET(configure)# sip-config
```

9. If applicable to your deployment, enable the **add-second-pai-for-emergency-calls** option.

```
ACMEPACKET(sip-config)#option +add-second-pai-for-emergency-calls
```

10. Use **done**, **exit**, and **verify-config** to complete configuration.

    This configuration supports real-time configuration, and therefore does not require a reboot.

Configuring an S8HR Profile

You configure an S8HR profile for serving roaming UEs with values that apply to your deployment.

To configure an S8HR profile:

1. In Superuser mode, use the following command sequence to access s8hr profile configuration:

   ```
   ORACLE# configuration terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# s8hr-profile
   ```

   If you are configuring a pre-existing profile, use **select** to choose the profile you want to edit.

2. Use the name parameter to create a reference name for this profile and press Enter.

   ```
   ORACLE(s8hr-profile)# name s8hr1
   ```

3. Type register-hold-for-plmn-info, and enter the length of time to hold REGISTERs while waiting for PLMN information. The valid value range from 0-30 seconds. The default of zero disables the parameter.

   ```
   ORACLE(s8hr-profile)# register-hold-for-plmn-info 10
   ```

4. Type plmn-id-prefix, and enter the prefix string the system uses for P-Visited-Network-ID headers associated with this profile.

   ```
   ORACLE(s8hr-profile)# plmn-id-prefix s8hr
   ```

5. Type emergency-reject-on-ident-error, then type enabled or disabled. If this flag is enabled, the SBC rejects any emergency session for which user identity validation fails.

   ```
   ORACLE(s8hr-profile)# emergency-reject-on-ident-error enabled
   ```

6. Type emergency-reject-reason, and enter a reason to explain why an emergency session is rejected.

   ```
   emergency-reject-reason "IMEI cannot be verified"
   ```

7. Type local-mccmnc, and enter the local MNC where the SBC resides. This value should be a 2 or 3-digit integer.

   ```
   ORACLE(s8hr-profile)# local-mccmnc 22
   ```

8. Type epc-id-required, and enter the epc operational mode for emergency roaming service. Values include:

   - IMEI_OR_IMSI (Default)
   - IMSI_AND_IMEI_LOOSELY
   - IMSI_LOOSELY
   - IMSI_STRICT

   ```
   ORACLE(s8hr-profile)# epc-id-required IMSI_OR_IMEI
   ```

9. Type encrypt-disabled-mnc-mcc, and enter a list of networks for which the system must disable encryption when using this profile.

```
ORACLE(s8hr-profile)# encrypt-disabled-mnc-mcc 033444 456789 *
```

   To disable encryption for all roaming networks, enter an asterisk (*).

Apply this S8HR profile to the appropriate **sip-interface**.

Applying an S8HR Profile to a SIP Interface

You configure the applicable sip interface to use the applicable S8HR profile to support this roaming on that interface.

To apply an S8HR profile:

1. In Superuser mode, use the following command sequence to access existing sip interface configuration element:

```
ORACLE# configuration terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
```

2. Select the access side of the sip interface, use select:

```
ORACLE(sip-interface)# select
```

3. Set the **s8hr-profile** parameter to the applicable profile's **name** parameter.

```
ORACLE(sip-interface)# s8hr-profile s8hr1
```

4. Type **done** to save the work.

Configuring a PLMN INFO Change Subscription

For S8HR operations, you configure the Oracle Communications Session Border Controller (SBC) with a subscription to PLMN location changes with the PCRF. The external policy configuration provides for this subscription configuration.

To configure a subscription to PLMN location changes:

1. Access the **ext-policy-server** configuration element.

```
ORACLE #
configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. To select the applicable external policy server, use select:

```
ORACLE(ext-policy-server)# select
```

3. Configure the **specific-action-subscription** parameter with the **plmn-change** value.

```
ORACLE(ext-policy-server)# specific-action-sig-flow-subscription plmn-
change
```

4. Type **done** to save the work.

Apply this policy server to the appropriate **sip-interface**.

LIR LIA Lookup to Home Subscriber Server from I-BCF

The Oracle Communications Session Border Controller, acting as an I-BCF, can perform a Location Information Lookup (LIR) to a Home Subscriber Server (HSS) to find the S-CSCF where to forward a request message. LIR/LIA messages are part of the Diameter Cx interface, which is used to communicate with HSS.

In an IMS environment, a user registers with S-CSCF and that S-CSCF assigns itself with an HSS as the serving registrar for that user. When a call needs to be routed to the user, an I-BCF performs a location information lookup to the HSS to determine the S-CSCF for that user and route the request toward that S-CSCF.

Home Subscriber Server

The Oracle Communications Session Border Controlleris configured with an HSS configuration element called home subscriber server. When the Oracle Communications Session Border Controlleri receives an INVITE from a user, a local policy routing lookup is performed.

The home subscriber server configuration element is configured with a name, which is referenced in a local policy next-hop parameter. It also includes, a state, IP address, port, and realm.

P-Acme-Serving Parameter Creation

The Oracle Communications Session Border Controlleri can insert a P-Acme-Serving parameter into the Route header of a message. This parameter is inserted when the **add-lookup-parameter** parameter is enabled in the home subscriber server configuration element and the Oracle Communications Session Border Controlleri queried that HSS to find the message's destination.

The P-Acme-Serving parameter can be helpful for informing other Oracle Communications Session Border Controllers that this system is acting as the UA's I-BCF and that the HSS lookup has been done.

HSS Watchdog Keepalives

The home subscriber server configuration element contains a **watchdog-ka-timer** parameter that sets a time to send DWR messages to the HSS. This is similar to external policy server Diameter Heartbeat functionality. When 3/4 of this timer elapses, the Oracle Communications Session Border Controlleri begins disconnection processes. DWR message creation is suspended when any other activity with the HSS occurs within watchdog-ka-timer seconds. DWR creation resumes watchdog-ka-timer seconds if all other traffic stops.

Local Policy

The Oracle Communications Session Border Controlleri performs all expected processing on an incoming message and begins a local policy lookup. When it encounters a policy attribute with a next hop in the form of cx:element-name, it will perform an LIR/LIA transaction over the Cx Diameter interface to the home subscriber server referenced as element-name.

The Oracle Communications Session Border Controlleri forwards the initial INVITE to the address in the Server-Name AVP returned by the HSS in the LIA message.

Compliance for the Vendor-Specific-Application-Id

You can configure the SBC to perform CER and LIR transactions over the Cx interface in compliance with RFC 6733 with respect to the contents of the Vendor-Specific-Application-Id AVP (260). You do this by setting the **rfc6733compliant** option under the applicable **home-subscriber-server**. RFC 6733 compliance consists of several behaviors, including limiting the number of Vendor-Ids present in the CER and LIR diameter messages to one. By default, the system aligns with RFC 3588 and sends out both Vendor-IDs in the diameter messages.

In addition to establishing this compliance you can also specify the Vendor-Id for the CERs Vendor-Specific-Application-Id AVP by configuring the **vendor-id** option under the applicable **home-subscriber-server**. This option has two valid values, including **vendor-id=etsi** and **vendor-id=3gpp**.

The SBC uses both the **vendor-id** and **rfc6733compliant** options to determine the contents of the Vendor-Specific-Application-Id AVP it sends in these CER and LIR messages:

If you set the **rfc6733compliant** option, the SBC includes the following as Vendor-ID, based on your **vendor-id** setting:

- If you set **vendor-id=3gpp**, then SBC sends the value 10415 as Vendor-Id.
- If you set **vendor-id=etsi**, then SBC sends the value 13019 as Vendor-Id.

The SBC ignores any **vendor-id** setting if you have not set the **rfc6733compliant** option. If you have set the **rfc6733compliant** option, but not set the **vendor-id** option, the SBC adds both Vendor-Ids in the Diameter messages.

The syntax for configuring both options, with vendor-id set to etsi, follows:

```
ORACLE(home-subscriber-server)# options +vendor-id=etsi
ORACLE(home-subscriber-server)# options +rfc6733compliant
```

If you type the option without the plus sign, you overwrite any previously configured options. To append new options to an element's options list, prepend the new option with a plus sign as shown in the example.

You can monitor the processing of this feature in debug logs. Applicable log entries would appear as shown below.

```
For HSS <hss_name> the configured Vendor-Id option is xxxx
```

Reporting

You can monitor this feature using the **show home-subscriber-server** ACLI command to check connection establishment and LIR/LIA transaction detail.

```
SBC_1# show home-subscriber-server
02:50:03-186
HSS Status                  -- Period -- -------- Lifetime        --------
                 Active       High   Total      Total  PerMax    High
Client Trans          0          0       0          2       1       1
Server Trans          0          1       6       2380       4       1
Sockets               1          1       0          1       1       1
Connections           1          1       0          1       1
1
---- Lifetime        ----
                     Recent      Total       PerMax
LIR                       0          1       1
Sent Requests             0          1       1
Sent Req Accepted         0          0       0
Sent Req Rejected         0          0       0
Sent Req Expired          0          0       0
Sent Req Error            0          0       0
HSS  Errors               0          1       1
```

Vendor-Specific-Application-Id AVP

The Vendor-Specific-Application-Id AVP (AVP Code 260) is of type Grouped. This AVP indicates that the SBC supports an application. This AVP includes the following AVPs:

- Auth-Application-Id—This unsigned32 AVP uses code 258. It indicates support of the Authentication and Authorization portion of an application, and is required in all accounting messages.
- Acct-Application-Id—This unsigned32 AVP uses code 259. It indicates support of the Accounting portion of an application, and is required in all accounting messages.
- Vendor-Id AVP—This unsigned32 AVP uses code 266. It contains the code assigned to the vendor responsible for this Diameter application.

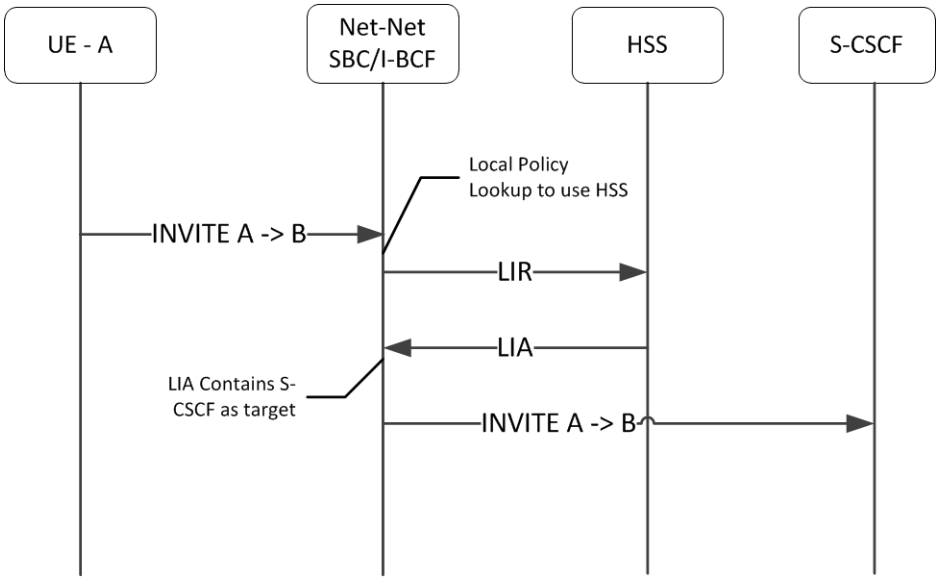
The Vendor-Specific-Application-Id AVP must have only one instance of both or either Auth-Application-Id or Acct-Application-Id AVP to avoid being rejected. The Vendor-Id may have a value of zero, which indicates that it should be ignored.

LIR LIA Transaction

The LIR includes the Public-User-Identity AVP containing the request URI from the SIP request. The HSS responds with the assigned S-CSCF server (a Oracle Communications Session Border Controlleri) for this PUID. The answer is the form of a Location Info Answer (LIA) and is in the Server Name AVP. If the S-CSCF specified in this AVP is not the current Oracle Communications Session Border Controlleri, then the INVITE is forwarded to the address specified in the LIR.

If the S-CSCF returned in the LIR is this Oracle Communications Session Border Controlleri, then the AoR from the request URI is found in the registration cache and the message is forwarded to that endpoint. When the registration cache entry does not exist or is invalid, local

policy processing continues with the next policy-attribute following "stop-recurse" rules. If there are no other routes, then a 404 is sent to the UA who sent the INVITE.



LIR Format

The LIR format is dependent on the RFC compliance for which you have configured the SBC. When configured for RFC 3588 (3GPP), the LIR contains the following information:

```
Vendor-Id: 9148
Supported-Vendor-Id: 10415
Supported-Vendor-Id: 13019
Vendor-Specific-Application-Id (grouped):
    Vendor-ID: 10415
    Vendor-ID: 13019
    Auth-Application-ID: 16777216
Origin-Host: IP Address:port of the system's source interface.
Origin-Realm: Realm-name where the system talks to this HSS.
Destination-Host: IP address of the HSS.
Public-Identity: Request URI of SIP Request
```

When configured for RFC 6733 (ETSI), the LIR contains the following information:

```
Vendor-Id: 9148
Supported-Vendor-Id: 13019
Vendor-Specific-Application-Id (grouped):
    1*  Vendor-ID: 13019
    0*1 Auth-Application-ID: 16777216
Origin-Host: IP Address:port of the system's source interface.
Origin-Realm: Realm-name where the system talks to this HSS.
Destination-Host: IP address of the HSS.
Public-Identity: Request URI of SIP Request
```

EXAMPLE - CER Format

The CER format is dependent on the RFC compliance for which you have configured the SBC. When configured for RFC 3588 (3GPP), the CER contains the following information:

```
CER::= < Diameter Header: 257, REQ >
          { Origin-Host }
          { Origin-Realm }
          { Host-IP-Address }
          { Vendor-Id }
          { Product-Name }
          [ Origin-State-Id ]
          [ Supported-Vendor-Id ]
          [ Auth-Application-Id ]
          [ Inband-Security-Id ]
          [ Acct-Application-Id ]
          [ Vendor-Specific-Application-Id ]
          [ Firmware-Revision ]
          [ AVP ]
```

When configured for RFC 6733 (ETSI), the CER contains the following information:

```
CER::= < Diameter Header: 257, REQ >
          { Origin-Host }
          { Origin-Realm }
         1* { Host-IP-Address }
          { Vendor-Id }
          { Product-Name }
          [ Origin-State-Id ]
        * [ Supported-Vendor-Id ]
        * [ Auth-Application-Id ]
        * [ Inband-Security-Id ]
        * [ Acct-Application-Id ]
        * [ Vendor-Specific-Application-Id ]
          [ Firmware-Revision ]
          [ AVP ]
```

LIA Format

The LIA is expected to contain:

```
Server-Name: S-CSCF address to forward this message
```

Home Subscriber Server Configuration

To configure a home subscriber server (HSS):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

```
ORACLE(configure)# session-router
```

3. Type **home-subscriber-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# home-subscriber-server
ORACLE(home-subscriber-server)#
```

4. **name**—Enter the name for this home subscriber server configuration element to reference from other configuration elements.
5. **state**—Set this to **enabled** to use this configuration element.
6. **address**—Enter the IP address of this HSS.
7. **port**—Enter the port number on the HSS where to connect. The default value is 80.
8. **realm**—Enter the realm name where this HSS exists.
9. **watchdog-ka-timer**—Enter the period of time to send DWRs to the HSS.
10. **add-lookup-parameter**—Set this to enabled for the Oracle Communications Session Border Controlleri to add a P-Acme-Serving parameter to the Route header.
11. Type **done** when finished.

Local Policy Configuration

To direct a policy attribute to retrieve a next hop from an HSS via an LIR lookup:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the session router path.

```
ORACLE(configure)# session-router
```

3. Type **local-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# local-policy
```

4. Choose an existing local-local object with the select command.

```
ORACLE(local-policy)# select
<source-realm>:
1: realms 'realm01'; from *; to *
selection: 1
```

5. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

```
ORACLE(local-policy))# policy-attributes
ORACLE(policy-attributes)#
```

6. **next-hop**—Set this parameter to **cx:name-of-HSS-configuration** for this policy attribute to lookup routes by querrying the named HSS via an LIR.
7. Type **done** and save and activate your configuration.

Statistics

The show home-subscriber command displays detailed information about HSS transactions. For example:

```
ORACLE## show home-subscriber-server
17:54:58-186
HSS Status                  -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0         12       4       1
Server Trans           0       0       0          1       1       1
Sockets                1       1       0          1       1       1
Connections            1       1       0          1       1       1
                           ---- Lifetime ----
                    Recent      Total  PerMax
LIR                      0          0       0
Sent Req Accepted        0         11       3
Sent Req Rejected        0          0       0
Sent Req Expired         0          0       0
Sent Req Error           0          0       0
Internal Errors          0          0       0
```

Note the following statistics provided for Recent and Lifetime periods:

- LIR—Number of LIR requests sent
- Sent Req Accepted—Number of requests for which we got success response (2xxx)
- Sent Req Rejected—Number of permanent failures (5xxx)
- Sent Req Expired—Number of requests for which there was no response
- Sent Req Error—Number of protocol errors/bad requests (1xxx, 3xxx, 4xxx)

Emergency Access Transfer Function

The Emergency Access Transfer Function (EATF) is a logical, functional service defined in 3GPP TS 23.167, *IP Multimedia Subsystem (IMS) Emergency Sessions*, and TS 23.237, *IP Multimedia Subsystem (IMS) Service Continuity*; Stage 2. The EATF, essentially a special-purpose B2BUA, anchors emergency calls to enable access transfer between packet-switched and circuit-switched networks during eSR-VCC procedures when the LTE equipment is moving outside LTE coverage to either a 2G or 3G carrier network. Similar to the Access Transfer Control Function (ATCF) and ATGW (Access Transfer Gateway), the EATF is always located in the visited network when the user equipment is roaming.

Lacking this capability, the LTE equipment would be forced to re-establish the emergency session in the circuit-switched network through the legacy accesses (2G or 3G).

When the LTE equipment initiates a packet-switched emergency session, the INVITE is sent to the EATF thru the P-CSCF/E-CSCF (collocated on the SBC). This original INVITE is identified as an emergency session by the A-SBC/P-CSCF because it contains either an emergency short number (112, 991, and so forth) or an emergency service URN such as urn:service:sos.fire.

In the event that handoff to a circuit-switched network is required, the Mobile Switching Center (MSC) server initiates the transfer with a SIP INVITE containing an E-STN-SR (Emergency Session Transfer Number for Single Radio VCC) in the Request URI of the INVITE. Each network has a single E-STN-SR, essentially the telephone number of the EATF service, that is used exclusively for emergency session transfer access. The MSC directs the INVITE to the I-CSCF, which, in turn, which forwards the request directly to the EATF.

The EATF checks the E-STN-SR to determine that handoff to the circuit-switched network is requested and proceeds with the access transfer of the active session. The EATF associates the received SIP INVITE with an existing SIP session already anchored at the EATF using the instance-id feature tag. The EATF then sends a re-INVITE to the E-CSCF, which terminates the emergency session.

Once the session modification procedures are complete, as indicated by the reception of the SIP ACK request from the target access leg, the source access leg, previously established via IMS, is released.

Enabling EATF Capability

Use the following procedure to enable EATF operations.

1. Use the following procedure to move to sip-config configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sip-config
ACMEPACKET(sip-config)#
```

2. Use the **eatf-stn-sr** parameter to enable EATF operation and to provide E.164 telephone number of the EATF service.

   Provide the E.164 telephone number as either a tel: or a sip: URI. When confirming the E-STN-SR value, the match succeeds based solely on the E.164 portion regardless of the URI type.

```
ACMEPACKET(sip-config)# eatf-stn-sr tel:+1-237-555-9999
ACMEPACKET(sip-config)#
```

3. Use **done** and **exit** to complete configuration.

Monitoring SRVCC Sessions

The **show sipd srvcc** command displays SRVCC handover counts including ATCF and EATF sessions.

```
ORACLE# show sipd srvcc
08:22:13-188
SRVCC Handover Stats                          ---- Lifetime ----
                                    Recent      Total  PerMax
Total Calls                              0          0       0
  Success                                0          0       0
  Failed                                 0          0       0
Calls After Answer                       0          0       0
  Success                                0          0       0
  Failed                                 0          0       0
Calls During Alerting                    0          0       0
```

```
  Success                          0          0          0
  Failed                           0          0          0
ATCF Cancellation                  0          0          0
Emergency Calls                    0          0          0
  Success                          0          0          0
  Failed                           0          0          0
EATF Cancellation                  0          0          0
```

These counters are defined as follows:

- Total Calls - Total calls subjected to SRVCC
- Total Success - Total successful SRVCC hand-off
- Total Failed - Total failed SRVCC hand-off
- Calls After Answer - Total calls subjected to SRVCC in established phase
- After Answer Success - Total successful SRVCC hand-off in established phase
- After Answer Failed - Total failed SRVCC hand-off in established phase
- Calls During Alerting - Total calls subjected to SRVCC in alerting phase
- During Alerting Success - Total successful SRVCC hand-off in alerting phase
- During Alerting Failed - Total failed SRVCC hand-off in alerting phase
- ATCF Cancellation - Total ATCF cancellations
- Total Emergency Calls - Total SRVCC hand-off for Emergency calls
- Emergency Success - Total successful SRVCC hand-off for Emergency calls
- Emergency Failed - Total failed SRVCC hand-off for Emergency calls
- EATF Cancellation - Total EATF Cancellations

Multimedia Priority Service for VoLTE Access

The Oracle Communications Session Border Controller (SBC) supports Multimedia-priority services (MPS) to prioritize communications by emergency personnel within VoLTE networks. Specifically, the SBC supports emergency service authorization and resource allocation for an MPS call. You enable support for MPS by enabling **mps-volte** in the **sip-config**. You configure this support in conjunction with your **net-management-control** and **rph-profile** configurations.

MPS is designed by 3GPP as a component of National Security/Emergency Preparedness (NS/EP) in IMS networks. During an emergency, Multimedia-priority services (MPS) enables authorized emergency personnel to coordinate their efforts by applying priority to their communications.

MPS calls can differ from, for example, a 911 call in that the destination does not necessarily specify that it requires high priority. Instead, the information used to authorize priority call treatment is distributed via reg-event subscriptions to the P-CSCF. The SBC, acting as P-CSCF, authorizes the priority for the call and informs the PCRF about the call over the Rx interface. Having authorized priority and informed the PCRF, the SBC then handles the call via its Network Management Controls (NMC) function. NMC filters traffic to identify priority (and other) flows. For MPS, NMC offers a means directing the filtered traffic through to egress appropriately for its priority, as well as a fallback mechanism for applying priority to emergency calls.

Authorization and Resource Management

An SBC can support MPS priority calls when authorized via the name-space and priority values. These are specified by the resource value (r-value) parameter in a SIP method's Resource Priority Header (RPH). Applicable SIP methods, including INVITEs, are presented in RFC 4412. The SBC supports the following name-spaces:

- Emergency Telecommunications Services (ETS)
- Wireless Priority Service (WPS)

Supported priority values range from 0 to 4.

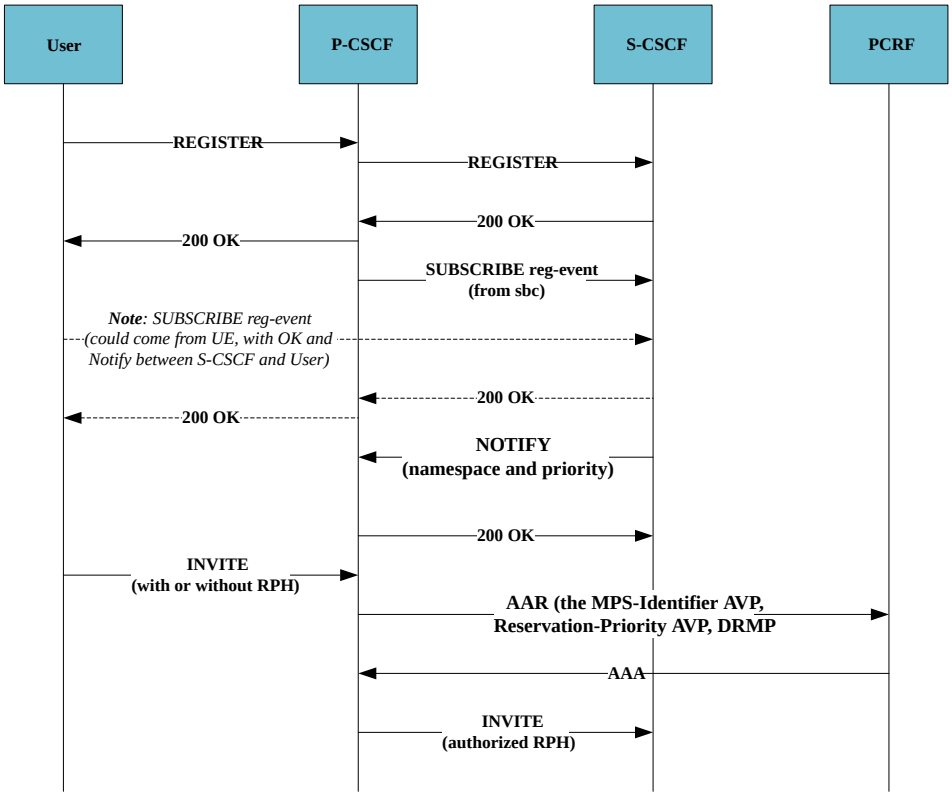
MPS operation requires that an administrator has assigned applicable user profiles for priority service at the HSS. Those users register their UEs via the P-CSCF. After successful registration, either the P-CSCF, and in some cases the UE, subscribes to registration events with the S-CSCF. The S-CSCF immediately sends a NOTIFY that specifies the priority available to the UE. The P-CSCF stores this information so it can perform priority authorization for these users and support RPH header processing for all applicable methods. The P-CSCF learns of priority changes, if any, via the subscription.

When this UE sends an INVITE that includes RPH, the SBC authorizes the call based on RPH match. This match is based on a single or multiple priority levels allowed by the HSS. Subsequently, the SBC informs the PCRF of the priority call over the Rx interface and the call proceeds.

General rules on RPH insertion include:

- For INVITEs from registered UEs by public identities for whom the HSS allows priority:
  - If there is no RPH, the SBC inserts the RPH it learns from the subscription.
  - If there is no match and the call is an emergency, the SBC inserts the RPH it learns from the subscription.
  - If there is no match and the call is not an emergency, the SBC inserts the RPH it learns from the subscription.
- For emergency or other unregistered call, the HSS does not have information needed for authorization. In these cases, the SBC falls back to its configured NS/EP configuration and uses its own RPH configuration to proceed with the call:
  - If there was an RPH header, the SBC overwrites it with the first RPH in the **rph-profile**.
  - If there was no RPH header, the SBC inserts it with the first RPH in the **rph-profile**.
  - If there was no RPH header or RPH in the reg-event, refer to the conditions described in the section on Matching by NMC and by RPH.

This MPS call flow depicts registration, AAR and INVITE, resulting in an authorized RPH.



When the SBC identifies an authorized priority call, it gets the priority treatment defined by the NMC control rule's treatment method. This typically exempts the priority call from the following local network management controls:

- Session agent constraints
- Bandwidth constraints (e.g. per realm bandwidth)
- External Policy Servers
- Per UE call admission control
- CPU constraints, assuming the call clears the ETS congestion control threshold

Diameter Processes

When the SBC handles an authorized RPH: containing an appropriate namespace and priority value in SIP Invite, it includes the following AVPs in an AAR towards the PCRF:

- **MPS-Identifier** AVP (AVP code 528, type OctetString, Rx Interface)
- **Reservation-Priority** AVP (AVP code 458, type Enumerated, Rx Interface)
- **DRMP** AVP (AVP code 301, type Enumerated, S6A Interface)

Important behaviors with respect to the AAR include:

- Generation of this AAR is based on **reserve-incomplete** and other configurations within the applicable **ext-policy-server** configuration and the AAR optimization configuration.
- The DRMP AVP, which indicates relative priority of the diameter message, contains the same value as the Reservation-Priority AVP, as defined by TS 29.214.
- The SBC includes these AVPs in the first AAR message, which initiates the Rx session for that subscriber and does not include them in subsequent AARs for the same session.

RPH Management

Important RPH header insertion conditions include:

- If an EP is authorized to use multiple namespaces, the SBC bases priority treatment on the first matched r-value in the RPH header of the ingress SIP INVITE.
- If there are multiple r-values for the same namespace, but different priorities are sent by the HSS and the RPH of the ingress SIP INVITE, the SBC sends the highest priority for that namespace that matches the first r-value received in the RPH header of the ingress SIP INVITE.
- The SBC provides MPS priority service for PS-CS handoff calls like any other. If subscribed for priority service at the HSS, the call gets priority service.
- If the SBC sends a call to an untrusted realm, it removes the RPH.
- If the SBC receives an INVITE from an untrusted realm, it removes the RPH.

The SBC performs RPH validity checks on SIP INVITEs during MPS processing. These are the same as those used for the existing GETS/NSEP feature. The SBC rejects Invalid SIP INVITE with a **400 - bad response** message along with a specific reason header. Reasons include:

- The RPH header contains an r-value that is not supported. (reason header-"Invalid RPH - Invalid r-value")
- A WPS call without an ETS r-value (reason header-"Invalid RPH - No ETS R-value")
- The SBC found two r-values of the same namespace (reason header- "Invalid RPH - Namespace repeated")

The table below presents scenarios and whether they result in authorizing priority.

Scenario (MPS feature enabled)	Send AAR?	RPH R-Value	Reg-event R-value	MPS-Identifier	Reservation-Priority	DRMP	Authorize Priority?
R-value in SIP Invite matches reg-event.	YES	WPS.3	WPS.3	WPS	3	3	Yes
R-value in SIP Invite partially matches reg-event.	YES	WPS or ETS.1	WPS.2	WPS	2	2	Yes
R-value in SIP Invite does not match reg-event.	YES	ETS.0	WPS.2	WPS	2	2	yes
R-Value in SIP invite, no r-value via reg-event. ETS call detected using NSEP.	YES (r-value in CFG)	ETS.2	none	Use CFG name-space	Use CFG priority value	Use CFG priority value	No

Scenario (MPS feature enabled)	Send AAR?	RPH R-Value	Reg-event R-value	MPS-Identifier	Reservation-Priority	DRMP	Authorize Priority?
No RPH header in SIP invite. R-value received from HSS.	YES	none	WPS.2	WPS	2	2	Yes
No RPH header in SIP invite. No R-value received from HSS. ETS call detection using NSEP	YES (r-value in CFG)	None	none	Use CFG name-space	Use CFG priority value	Use CFG priority value	No
No RPH header in SIP Invite or via reg-event and no ETS call detection using NSEP	NO	None	none	NA	NA	NA	No

Dependence on NSEP Configuration

Full MPS support utilizes the NSEP configurations listed below:

- National Security/Emergency Preparedness (NS/EP) (**rph-feature** enabled under **sip-config**)
- Network Management Control (NMC) configuration on the ingress realm (**net-management-control**)
- The **insert-arp-header** under **sip-config**, when enabled. This ensures the SBC includes the Accept-Resource-Priority header (ARPH) in responses.

You must ensure that the following configuration options are disabled to allow unregistered priority calls:

- **sip-interface**, **options -reject-unreg-priority-calls**
- **sip-interface**, **options -disallow-priority-calls**

Interaction with NSEP Configuration

Enabling **mps-volte** extends upon NSEP/NMC behavior by including MPS call authorization procedures and preventing the execution of the default NS/EP behavior. When you enable the MPS for VoLTE and the NSEP feature, they interact as follows:

- The SBC defaults to using the r-values received from HSS to determine priority. It uses the NSEP defined **rph-profile** only when priority information is not provided by HSS.

- If the SBC receives an unregistered emergency call and is configured to accept it, priority treatment based on existing NSEP feature using related configuration.
- If there are no r-values available from any source for a call, the SBC does not authorize priority and uses NSEP procedures to detect an ETS call.
- In case of congestion, SBC NSEP call treatment mechanisms ensure adequate resources for priority calls. Both the **nsep-user-sessions-rate** and the **nsep-load-limit** option in the **sip-config** establish the priority for RPH and other emergency calls, by reserving bandwidth and CPU cycles. The recommendation is to ensure that this traffic has more than 50% of bandwidth and CPU available to it.
- If the SBC receives an unregistered emergency call and the SBC configuration allows it to process further, the SBC decides whether to allow the call based on the NSEP feature's configuration.
- The SBC first attempts to derive values for the **MPS-Identifier** and **Reservation-Priority** AVP's from priority information received from the HSS. If an endpoint's subscription information does not reflect priority information and you have enabled the **rph-feature**, then these AVP's have values derived based on Network Management Controls, if the NSEP feature is enabled.

> **Note:**
>
> As you configure, you must be aware when your MPS configuration is overriding your GETS/NSEP configuration. The system does notify you of such conditions.

MPS Reporting

Execute the **show mps-stats** command to display the priority call statistics. Running the command without argument displays all MPS calls. Specifying an r-value displays statistics on sessions that use that r-value.

```
ORACLE# show mps-stats
             ------ Lifetime -----
                      Current      Total     PerMax
Incoming Calls              0          0          0


ORACLE# show mps-stats ets.2
                             -- Period -- -------- Lifetime --------
                   Active    High   Total      Total  PerMax    High
Inbound Sessions        0       0       0          0       0       0
Outbound Sessions       0       0       0          0       0       0
InbSessions Rej         0       0       0          0       0       0
OutbSessions Rej        0       0       0          0       0       0
```

Display the cached r-value for a specific user with the **show registration sipd by-user sip:[user] detailed** command.

```
ORACLE# show registration sipd by-user sip:user0@example.com detailed

Registration Cache (Detailed View)    Wed Mar 21 2018  18:35:41

User: sip:user0@example.com
  Registered at:  2018-03-21-18:34:34    Surrogate User: false
```

```
  Emergency Registration? No
  R-value: ETS.2 WPS.1
...
```

MPSIdentifier AVP (AVP 528)

The MPS-Identifier AVP is of type OctectString and indicates that an Application Function (AF, eg P-CSCF) session relates to an MPS session.

When the SBC receives an authorized Resource-Priority header containing an appropriate namespace and priority value in SIP Invite, and recognizes the need for priority treatment, the SBC will include the **MPSIdentifier** AVP in the AAR command towards the PCRF.

Reservation-Priority AVP (468)

The Reservation-Type AVP is of type Enumerates and specifies the priority associated with the reservation. The ETSI defines the protocol values in TS 183 026-V3.1.1, with a default of 0 being the lowest and the highest being 15. The SBC maps a subset of these AVP values for use in MPS.

For MPS, IMS defines priority values using the default of 0 as the lowest priority and a range from 0 to 7. RFC 4412 further redefines these values with the level of priorities in reverse order. The SBC operates by mapping ETSI to IMS values with a default of 4 and the highest priority as 0, as shown below.

- The SBC implementation defines the diameter priorities as shown below, with the applicable strings recorded in logs:
  - DIAM_PRIORITY_DEFAULT (0)—This is the lowest level, and indicates no priority.
  - DIAM_PRIORITY_ONE
  - DIAM_PRIORITY_TWO
  - DIAM_PRIORITY_THREE
  - DIAM_PRIORITY_FOUR
  - DIAM_PRIORITY_FIVE
  - DIAM_PRIORITY_SIX
  - DIAM_PRIORITY_SEVEN—This is the highest priority, indicating an emergency.
- The SBC maps priorities in compliance with RFC 4412, for use with the HSS as follows:
  - HSS priority of 0 — value of 7 in AVP—Highest priority
  - HSS priority of 1 — value of 6 in AVP
  - HSS priority of 2 — value of 5 in AVP
  - HSS priority of 3 — value of 4 in AVP
  - HSS priority of 4 — value of 3 in AVP—Lowest priority

When the SBC recognizes the need for priority treatment by receiving an authorized Resource-Priority header containing an appropriate namespace and priority value in SIP Invite, and , In these cases, the SBC includes the **Reservation-Priority** AVP in the AAR command towards the PCRF.

DRMP AVP (301)

The Diameter Routing Message Priority (DRMP) AVP in Diameter messages is of type Enumerated and specifies the relative priority of the Diameter message on a scale of 0 to 15 where priority value zero is highest and 15 is the lowest priority.

When the SBC handles an authorized Resource-Priority header containing an appropriate namespace and priority value in SIP Invite, it includes the DRMP AVPs in an AAR towards the PCRF to specify its own priority.

Enabling MPS Services

You enable the **mps-volte** parameter within the **sip-config** .

To enable MPS services:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLEORACLE(session-router)# sip-config
   ```

   Use the ACLI **select** command to select the **sip-config**.

4. **mps-volte**—Set this parameter to **enabled** to enable MPS services. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Use Done, Save and activate to complete the procedure.

Applying Network Management Control Rules to a Realm

Once you have configured network management control rules, you enable their use on a per-realm basis.

To apply a network management control rule to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

If you are enabling network management controls for a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **net-management-control**—Set this parameter to **enabled** to apply network control rules in this realm. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

Restoration Procedures

When functioning as a P-CSCF within an IMS network, an SBC is required to conform to Restoration Procedures as defined in 3GPP TS 23.380, *IMS Restoration Procedures*, and 3GPP TS 24.229, *IP Multimedia Call Control Protocol Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)*.

TS 23.380 defines a set of standardized procedures for automatic restoration of connectivity after node or link failure. Section 4.4.3 of TS 23.380 imposes a specific injunction requiring that if the UE initiates an originating service request different from SIP REGISTER and the P-CSCF is unable to contact the S-CSCF in the Route, the P-CSCF shall return a specific error response to the UE to trigger a new registration. Section 5.2.6.3.2A of the related 3GPP TS 24.229 defines the structure and content of the required error response.

In the event of a P-CSCF failure to forward a request to the next hop device (whatever the reason), TS 24.229 mandates that the P-CSCF shall reject the call by returning a 504 (Server Time-out) response to the user endpoint. The 504 response must contain:

- a Content-Type header field with the value "application/3gpp-ims+xml"
- a P-Asserted-Identity header field set to the value of the SIP URI of the P-CSCF included in the Path header field during registration of the user whose UE sent the request causing this response
- an attached XML node-set (specified in Section 4.4.1 of 3GPP TS 24.615, *Communication Waiting (CW) using IP Multimedia (IM) Core Network (CN) subsystem,* containing a child element that conveys the reason for requiring a new registration

After issuing the 504 Server Time-out, the P-CSCF must reject any messages from the user endpoint except for registration requests.

3GPP IM CN Subsystem XML Body

The attached XML node-set is constructed as shown below.

```
<3gpp-ims version="1">
   <alternative-service>
      <type>
         <restoration/>
      </type>
      <reason>User configurable value</reason>
      <action>
```

```
        <initial-registration/>
      </action>
   </alternative-service>
<3gpp-ims>
```

Enabling Restoration Procedures

Use the following procedure to enable compliance with IMS Restoration Procedures mandated by 3GPP TS 23.380 and TS 24.229. You can enable compliant behavior on individual SIP interfaces. By default, TS 23.30/TS 24.229 compliance is disabled on all SIP interfaces.

1. From superuser mode, enter the following ACLI command sequence to access sip-interface configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# sip-interface
   ACMEPACKET(sip-interface)#
   ```

2. Use the select ACLI command to select the target SIP interface, that is the SIP interface to be brought into compliance with TS 23.30/TS 24.229 requirements.
3. Use the pcscf-restoration ACLI parameter to enable TS 23.30/TS 24.229 compliance.

   Enable compliance by assigning a string value to this parameter. The string provides the contents of the <reason> element.

   ```
   ACMEPACKET(sip-interface)# pcscf-restoration "Restoration procedures in
   force"
   ACMEPACKET(sip-interface)#
   ```

4. Use **done** and **exit** to complete enabling restoration procedures on the current SIP interface.
5. As required, repeat steps 1 through 4 to enable Restoration Procedures on other SIP interfaces.

19

SBC Processing Language (SPL)

SPL provides a means for Oracle to craft solutions and features to unique problems and deploy them in a portable plugin-type software package. SPL only works for SIP messaging. You may only run signed SPL files on your SBC available directly from Oracle.

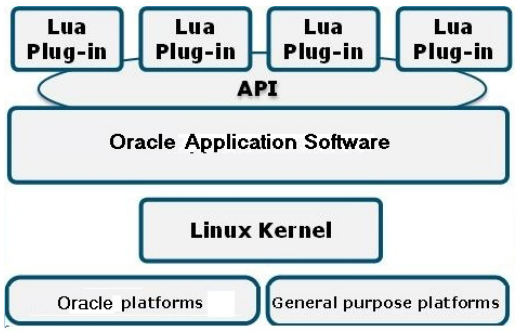
Upon boot, the SBC compiles all scripts in the /code/spl directory that are configured in the spl-config configuration element. If there is an error parsing SPL files, it is written to the log.sipd and the script is not loaded. Scripts are loaded in the order in which they are configured in the spl-plugins configuration element.

SPL Packages act identically to SPL plugins but contain multiple plugins in one file. When a package file is configured in the **name** parameter of the spl plugins configuration element, the SBC will load all SPL plugins contained in that package. You may also configure the SBC to execute a single plugin contained within the package with the syntax package-name:plugin-name. You may omit the .pkg extension when configuring the SBC to load one plugin from a package.

Oracle SPL Plug-ins

An SPL is an Oracle signed plug-in that integrates with the Oracle Communications Session Border Controller (SBC) application software to quickly add feature extensions without requiring a software upgrade or causing operational impacts. Each SPL plug-in is an executable, customized script that is based on the Lua open scripting language. Oracle SPL plug-ins allow you to add enhancements when you need them, rather than waiting for the next software release.

The following illustration shows how an SPL plug-in integrates with the SBC platform.



Supported SPL Engines

Each release supports a number of versions of the SBC Programming Language (SPL) engine, which is required to run SPL plug-ins on the Oracle Communications Session Border Controller (SBC).

View the Release Notes to see which SPL engines are supported for your software release.

Use the `show spl` command to see the version of the SPL engine running on the SBC.

SPL Parameter Configuration

SPL Plugins may create the **spl-options** parameter in either the session-agent, sip-interface, realm-config, or spl-config configuration elements. The **spl-options** parameter appears in the ACLI after an SPL plugin that creates the parameter is executed. The **spl-options** parameter will not necessarily appear in all four (or any) configuration elements. Where and when to configure the **spl-options** parameter is discussed in each plugin's specific documentation.

Upload an SPL Plug-in

The SBC comes preloaded with many SPLs. Use `show spl` to see the preloaded SPLs.

To upload a non-preloaded SPL to the SBC, SFTP the plugin file to the SBC's `/code/spl` directory.

Add an SPL Plug-in to the Configuration

Before the SBC executes an SPL, you must add the SPL plug-in file to the global **spl-config** element. SPLs that are not preloaded must also be configured in the **plugins** element.

The system executes SPL plug-ins in the order they are configured.

1. Access the **spl-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# spl-config
ORACLE(spl-config)#
```

2. Add the SPL to the global SPL options parameter.

   Use the table below to identify the right argument when configuring one of the preloaded SPLs.

SPL File Name	Option
SipHeaderExtensionMetadata.lua	sip-header-extension-metadata
HeaderNat.lua	header-nat
ComfortNoiseGeneration.lua	comfort-noise-generation
NttMsgConverter.lua	ntt-msg-converter
SurrogateRegister.lua	surrogate-register
UniversalCallId.lua	Universal-CallId

```
ORACLE(spl-config)# spl-options +header-nat
```

> **Note:**
>
> If you are enabling a preloaded SPL, type **done** and skip the rest of these steps.

3. To enable a previously uploaded SPL, access the **plugins** element.

```
ORACLE(spl-config)# plugins
ORACLE(spl-plugins)#
```

4. **name**—The name of the SPL file.

```
ORACLE(spl-plugins)# name HelloWorld.spl
```

5. Type **done** to save your work.
6. Save and activate the configuration.

Executing SPL Files

SPL files are executed in one of the following ways.

1. Run a **save-config** and **activate-config**.
2. Run a **save-config** and reboot the system.
3. Run the **reset spl** command.

All configured SPL files are refreshed by with the **reset spl** command. You can also refresh a specific file by typing **reset spl <spl-file>**.

```
ORACLE# reset spl HelloWorld.spl
```

Operators must remember that HelloWorld.spl will not be loaded on the next reboot.

> **Note:**
>
> Oracle suggests that scripts are only refreshed during system downtime.

If an non-preloaded SPL file exists in the /code/spl directory, but is not configured in the **spl-files** parameter, it will be ignored when the SBC loads all SPL plugins. You may still manually load an SPL file directly with the **reset** command.

Synchronize SPL Plug-in Files Across an HA Pair

In a High Availability (HA) configuration, both the active and the standby systems require the same version of the SBC Programming Language (SPL) plug-in script.

SPL files do not automatically synchronize when saving and activating your configuration. To configure the standby system, use the **synchronize spl** command from the active node in the HA pair.

```
ORACLE# synchronize spl
```

If you have any SPL files in the `/code/spl` directory on the active system, these files will sync and overwrite any existing SPL files in the `/code/spl` directory on the standby system.

To copy individual files, add the specific filename as an argument to the **synchronize spl** command. For example:

```
ORACLE#synchronize spl /code/spl/MediaPlayback.1.0.spl
ORACLE#synchronize spl /code/spl/LyncEmergencyCall.1.0.spl
ORACLE#synchronize spl /code/spl/SipHeaderExtensionMetadata.1.2.spl
ORACLE#synchronize spl /code/spl/UniversalCallId.1.spl
ORACLE#synchronize spl /code/spl/ComfortNoiseGeneration.1.1.spl
```

Maintenance and Troubleshooting

show spl

Typing **show spl** displays the following items:

- The version of the SPL engine
- The filenames and version of the SPL plugins currently loaded on the Oracle Communications Session Border Controller
- The signature state of each plugin
- The system tasks that each loaded plugin interacts with, enclosed in brackets.

For example:

```
ORACLE# show spl
SPL Version: C1.0.0
[acliConsole] File: signed_valid_lower_version.spl version: 1 signature:
signed and valid
[acliConsole] File: signed_valid.spl version: 1 signature: signed and valid
[sipd] File: signed_valid_lower_version.spl version: 1 signature: signed and
valid
[sipd] File: signed_valid.spl version: 1 signature: signed and valid
```

Adding the task to the end of the show spl command displays only the plugin information for the specified task. For example:

```
ORACLE# show spl sipd
SPL Version: C1.0.0
[sipd] File: signed_valid_lower_version.spl version: 1 signature: signed and
valid
[sipd] File: signed_valid.spl version: 1 signature: signed and valid
```

show running-config spl-config

The ACLI **show running-config spl-config** displays SPL specific configuration information on the system.

```
ACMEPACKET# show running-config spl-config
spl-config
        spl-options
        plugins
                name                            LyncEmergencyCall.1.0.spl
```

```
last-modified-by                admin@216.41.24.2
last-modified-date              2012-10-12 15:31:05
```

show spl-options

The ACLI **show spl-options** command displays SPL-specific options registered by an SPL.

```
ACMEPACKET# show spl-options
   1. return_183_initial_invite: Returns a 183 provisional response when a
emergency call is placed through Lync [LyncEmergencyCall.1.0.spl,config]
```

show directory code spl

The ACLI **show /code/spl** command displays the SPLs stored in the /code/spl directory.

```
ACMEPACKET# show directory /code/spl
Listing Directory /code/spl:
drwxrwxrwx  1 0       0                4096 Aug 13 10:07 ./
drwxrwxrwx  1 0       0                4096 Aug 19 22:25 ../
-rwxrwxrwx  1 0       0                3163 Aug 13 10:07
LyncEmergencyCall.1.0.spl
```

SPL Signature State

Upon executing **show spl <task>** , the ACLI displays SPL file information including the signature which will be in one of three states:

1. not signed
2. signed and valid
3. signed but invalid

SPL Log Types

SPL log messages can often be found in the log file for the system task to which the SPL applies when that task is set to DEBUG level. You can find the output specific to SPL by the identifying prefix **[SPL]**.

```
Aug 30 15:06:07.454 [SPC] Executing SPL callback from file:
SipHeaderExtensionMetadata.1.2.spl
Aug 30 15:06:07.454 [SPL] Checking for LRE-Identifier to match triggered
session-recording-server
Aug 30 15:06:07.454 [SPC] Creating table of name
'AcmeSipServerTransDataTable' with key [0x34522878]
Aug 30 15:06:07.454 [SPC] Creating new temporary session table of key
[_SESSION_0x34522878]
Aug 30 15:06:07.454 [SPL] SIP Interface ingressSIP has option
Aug 30 15:06:07.454 [SPL] Storing data from message to insert into metadata
```

Delete SPL Plugin Files

Deleting files from /code/spl must be performed via SFTP.

Service Provider SPLs

The following SPLs are available for service providers.

Universal Call Identifier SPL

The Universal Call Identifier SPL generates or preserves a UCID based on configuration. Once a UCID is generated or preserved, the system adds the value to subsequent egress SIP INVITE and ACK messages within the session. The SBC only inserts a UCID into methods it does not generate itself. Any methods that the SBC generates to send to the egress side do not have the UCID. You can also set the SPL to remove unwanted UCID headers to avoid duplicity in egress SIP requests.

Using the Universal Call Identifier SPL, you can identify requests within a particular session by manipulating the following vendor specific UCID headers:

- User-to-User
- Cisco-GUID
- Cisco-GUCID
- Genesys-UUID

The UCID is added as extension data to the session element of the recording's metadata when using SIPREC.

You must configure one of the following SPL options for it to be enabled:

- UCID-App-ID
- GUID-Node-ID
- GUCID-Node-ID
- GenUUID-App-ID

Each SPL option allows you to set an identifying value, as defined by the vendors. The SPL does not validate any input for the SPL options. It is the responsibility of the Administrator to set the correct value.

You may further modify the action of the SPL by adding **replace-ucid** or **convert-to** or both to your SPL options.

> **Note:**
>
> The replace-ucid and convert-to options have no effect unless you also configure UCID-App-ID, GUID-Node-ID, GUCID-Node-ID, or GenUUID-App-ID.

UCID-App-ID

The UCID-App-ID SPL option allows the Oracle Communications Session Border Controller (SBC) to examine ingress SIP requests for the "User-to-User" header. When present, the header is transparently passed through the egress SIP message. If set to replace-ucid or the header is not present, the system generates a new value for "User-to-User".

You must set the value to a 2-byte hex-ascii value that represents the app ID. All input is truncated to 4 characters. Any characters outside the range of 0-9 and A-F will result in an invalid User-to-User header.

GUCID-Node-ID

The **GUCID-Node-ID** SPL option allows the Oracle Communications Session Border Controller (SBC) to examine ingress SIP requests for the Cisco-GUCID header. When present, the header is transparently passed through the egress SIP message. If set to **replace-ucid** or the header is not present, the system generates a new value for Cisco-GUCID.

You must set the value to a 48-bit node ID in the version 1 UUID defined by RFC 4122. You can enter the value in decimal or hexadecimal notation. The value must be prefixed with 0x when hexadecimal.

GUID-Node-ID

The **GUID-Node-ID** SPL option allows the Oracle Communications Session Border Controller (SBC) to examine ingress SIP requests for the Cisco-GUID header. If present, the header is transparently passed through the egress SIP message. The system generates a new value for Cisco-GUID if not present or the SPL option is set to **replace-ucid**.

You must set the value to a 48-bit node ID in the version 1 GUID defined by RFC 4122. You can enter the value in decimal or hexadecimal notation. The value must be prefixed with 0x when hexadecimal.

GenUUID-App-ID

The GenUUID-App-ID SPL option allows the SBC to examine ingress SIP requests for the “X-Genesys-CallUUID” header. When present, the header is transparently passed through the egress SIP message.

If you configure the option with the **replace-ucid** parameter or the header is not present, the system generates a new value for “X-Genesys-CallUUID”. The SBC replaces the Genesys UUID value only if the incoming message has a X-Genesys- CallUUID header. If the header is not present, the SBC adds the SIP header with the value equal to the generated/preserved UUID.

The uuid specified is a 33 byte unique string, including the \0 terminating symbol.

convert-to

The **convert-to** SPL option allows the Oracle Communications Session Border Controller (SBC) to examine ingress SIP requests for multiple UCID headers. This option has no effect unless appended to another SPL option.

You must set the convert-to SPL option to one of the following values:

- **Avaya**—Removes all Cisco-GUCID and Cisco-GUID headers from egress SIP requests.
- **GUID**—Removes all User-to-User and Cisco-GUCID headers from egress SIP requests.
- **GUCID**—Removes all User-to-User and Cisco-GUID headers from egress SIP requests.
- **Genuuid**—Remove any Avaya or Cisco UUID header and will add the XGenesys-CallUUID header.

Example SPL Options

The following are examples of the Universal Call Identifier SPL.

Example 1

```
UCID-App-ID=0023,replace-ucid,convert-to=Avaya
```

This creates a User-to-User header based on a node ID of 23. Any value on the ingress side is replaced with the newly generated value. Removes all Cisco-GUID and Cisco-GUCID headers from egress messages.

Example 2

```
GUID-Node-ID=0x124578,convert-to=Guid
```

This creates a Cisco-GUID header if one does not exist in the ingress request. Removes all User-to-User and Cisco-GUCID headers from egress messages.

Sample Metadata

The following sample shows metadata with new extension data added by the Universal Call Identifier SPL:

```
<extensiondata xmlns:apkt=http:/acmepacket.com/siprec/extensiondata>
        <apkt:ucid>00FA08001900014E3E7D5C;encoding=hex</apkt:ucid>
</extensiondata>
<extensiondata xmlns:apkt=http:/acmepacket.com/siprec/extensiondata>
        <apkt:ucid>C0934BE72BF711D6800285D16359919A</apkt:ucid>
</extensiondata>
```

Configuring Universal Call Identifier Options

The SPL options must be configured on the ingress session-agent, realm-config, or sip-interface. If you update the values of the SPL options for the Universal Call Identifier SPL, you must perform a **save** and **activate** in order for the new option to take effect.

To configure the SPL options:

1. Access the **session-agent** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

2. Select the previously configured **session-agent** element you want to configure.

   ```
   ORACLE(session-agent)# select
   <hostname>:
   1: host1.example.com
   2: host2.example.com
   ```

```
selection: 1
ORACLE(session-agent)#
```

3. Type `spl-options +UCID-App-ID="<value>"` where <value> is the additional header information to store and press Enter. The default behavior stores only the Request-URI and realm-id.

```
ORACLE(spl-config)# spl-options +UCID-App-ID=0023
```

4. Type **done** to save your work.

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 Communications Session Border Controller (SBC). 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 Interactive Session Recorder (ISR). Metadata is a XML MIME attachment that describes recording details to the 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:

```
<?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@example.com</value>
        <value>sip:bob@example.org</value>
      </apkt:header>
      <apkt:header label="Diversion">
        <value>&lt;sip:jojo@example1.com&gt;;happy=days;green=envy</value>
        <value>&lt;sip:bebe@example2.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.

```
ORACLE(spl-config)# spl-options +Extension-Headers="P-Asserted-
Identity,Diversion"
```

3. Type **done** to save the configuration.

Configure LRE for SIPREC Metadata

The **LRE-Identifier** option may be configured on each **session-agent**, **realm-config**, or **sip-interface** that interacts with the session recording server. This option is not recognized in the global spl-config. This option is required for SPL functionality.

To configure the LRE-Identifier option:

1. Access the **spl-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# spl-config
ORACLE(spl-config)#
```

2. Type `spl-options +LRE-Identifier="<value>"`where <value> is the additional header information to store and press Enter. The default behavior stores the identifier of the logical remote identity.

```
ORACLE(spl-config)# spl-options +LRE-Identifier
```

3. Type **done** to save your work.

SBC Deployment Behind a NAT Device

Use the **Support for SBC Behind NAT** SPL plug-in for deploying the Oracle Communications Session Border Controller (SBC) on the private network side of a Network Address Translation (NAT) device. The **Support for SBC Behind NAT** SPL plug-in changes information in SIP messages to hide the end point located inside the private network. The specific information that the **Support for SBC Behind NAT** SPL plug-in changes depends on the direction of the call, for example, from the NAT device to the SBC or from the SBC to the NAT device.

Configure the **Support for SBC Behind NAT** SPL plug-in for each SIP interface that is connected to a NAT device. One public-private address pair is required for each SIP interface that uses the SPL plug-in, as follows.

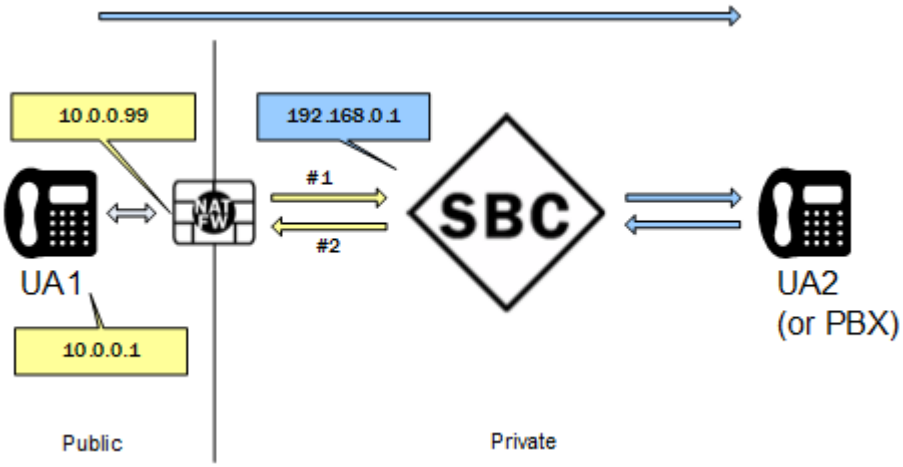
- The private IP address must be the same as the SIP Interface IP address.
- The public IP address must be the public IP address of the NAT device.

> **Note:**
>
> The HeaderNat SPL is not supported on the Session Router.

The following illustrations show the SBC deployed in the private network behind a NAT device, using the **Support for SBC Behind NAT** SPL plug-in. Examples follow each illustration to show where the **Support for SBC Behind NAT** SPL plug-in changes the SIP message information.

Call Initiated on the Access Side

In the following illustration, UA1 invites UA2 to a session and UA2 responds.



1. UA1 sends an INVITE through the NAT device to the SBC with the following message.

```
INVITE sip:service@10.0.0.99:5060 SIP/2.0
Via: SIP/2.0/UDP 10.0.0.1:5060;branch=z9hG4bK-3539-1-0
Contact: sip:sipp@10.0.0.1:5060
…
Content-Type: application/sdp

o=user1 53655765 2353687637 IN IP4 10.0.0.1
```

```
c=IN IP4 10.0.0.1
…
```

The **Support for SBC Behind NAT** SPL plug-in looks for the public SIP Interface IP address 10.0.0.99 in R-URI, Via, Contact, and SDP. The SPL plug-in finds 10.0.0.99 in R-URI and changes it to the private SIP Interface IP address 192.168.0.1.

```
INVITE sip:service@192.168.0.1:5060 SIP/2.0
Via: SIP/2.0/UDP 10.0.0.1:5060;branch=z9hG4bK-3539-1-0
Contact: sip:sipp@10.0.0.1:5060
…
Content-Type: application/sdp

o=user1 53655765 2353687637 IN IP4 10.0.0.1
c=IN IP4 10.0.0.1
…
```

2. The SBC sends a Reply to UA1.

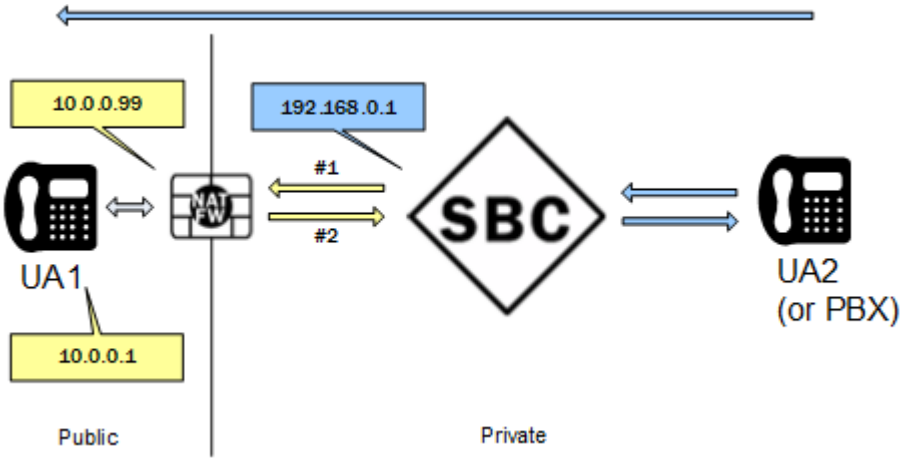
```
SIP/2.0 200 OK
Via: SIP/2.0/UDP
10.0.0.1:5060;received=192.168.0.70;branch=z9hG4bK-3539-1-0
Contact: <sip:192.168.0.1:5060;transport=udp>
Content-Type: application/sdp
…
o=user1 53655765 2353687637 IN IP4 192.168.0.1
c=IN IP4 192.168.0.1
…
```

The **Support for SBC Behind NAT** SPL plug-in looks for the private SIP interface IP address 192.168.0.1 in R-URI, Via, Contact, and SDP. The SPL plug-in finds 192.168.0.1 in Contact and SDP and changes it to the public IP 10.0.0.99.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP
10.0.0.1:5060;received=192.168.0.70;branch=z9hG4bK-3539-1-0
Contact: <sip:10.0.0.99:5060;transport=udp>
Content-Type: application/sdp
…
o=user1 53655765 2353687637 IN IP4 10.0.0.99
c=IN IP4 10.0.0.99
…
```

Call Initiated on the Core Side

In the folowing illustration, UA2 invites UA1 to a session and UA1 responds.



1. The SBC sends an Invite to UA1.

```
INVITE sip:service@10.0.0.1:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.0.1:5060;branch=z9hG4bKbgs21h30a8kh8okcv790.1
Contact: <sip:sipp@192.168.0.1:5060;transport=udp>
Content-Type: application/sdp
…
o=user1 53655765 2353687637 IN IP4 192.168.0.1
c=IN IP4 192.168.0.1
…
```

The **Support for SBC Behind NAT** SPL plug-in looks for the private IP address 192.168.0.1 in R-URI, Via, Contact, and SDP. The SPL plug-in finds 192.168.0.1 in Via, Contact, and SDP and changes it to the public IP address 10.0.0.99.

```
INVITE sip:service@10.0.0.1:5060 SIP/2.0
Via: SIP/2.0/UDP 10.0.0.99:5060;branch=z9hG4bKbgs21h30a8kh8okcv790.1
Contact: <sip:sipp@10.0.0.99:5060;transport=udp>
Content-Type: application/sdp
…
o=user1 53655765 2353687637 IN IP4 10.0.0.99
c=IN IP4 10.0.0.99
…
```

2. UA1 sends a Reply to the SBC.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.0.0.99:5060;branch=z9hG4bKbgs21h30a8kh8okcv790.1
Contact: <sip: 10.0.0.1:5060;transport=UDP>
Content-Type: application/sdp
…
o=user1 53655765 2353687637 IN IP4 10.0.0.1
c=IN IP4 10.0.0.1
…
```

The **Support for SBC Behind NAT** plug-in looks for the public IP 10.0.0.99 in R-URI, Via, Contact, and SDP. The SPL plug-in finds 10.0.0.99 in Via, changes it to the private SIP interface IP address 192.168.0.1.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.0.1:5060;branch=z9hG4bKbgs21h30a8kh8okcv790.1
Contact: <sip: 10.0.0.1:5060;transport=UDP>
Content-Type: application/sdp
…
o=user1 53655765 2353687637 IN IP4 10.0.0.1
c=IN IP4 10.0.0.1
…
```

Configure the SBC Behind a NAT Device Option

Configure one public-private address pair for each SIP interface that uses the **Support for SBC Behind NAT** SPL plug-in, as follows.

- The private IP address must be the same as the SIP interface IP address.
- The public IP address must be the public IP address of the NAT device.

Before You Begin

- Confirm that the SIP interface is configured.
- Confirm that you are in the Superuser mode.

To configure the SIP interface IP addresses:

1. Access the **sip-interface** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

2. Select the SIP interface.

   ```
   ORACLE(sip-interface)# select
   <RealmID>:
   1: DefaultENT 172.16.1.100:5060
   2: DefaultSP  192.168.0.1:5060

   selection:2
   ORACLE(sip-interface)#
   ```

3. Type `spl-options +HeaderNatPrivateSipIfIp "<value>"`, where <value> is the private SIP interface IP address.

   ```
   ORACLE(sip-interface)#spl-options +HeaderNatPrivateSipIfIp=192.168.0.1
   ```

4. Type **spl-options +HeaderNatPublicSipIfIp "<value>"**, where <value> is the public IP address of the NAT device, and press <Enter>.

   ```
   ORACLE(sip-interface)#spl-options +HeaderNatPublicSipIfIp=10.0.0.99
   ```

5. Type **done**, and press <Enter>.
6. Save and activate the configuration.

SIP Trunking SPLs for Specific Service Providers

The SBC includes SPLs to support trunking requirements for specific service providers. This support includes identifying requests within applicable sessions and manipulating values before forwarding subsequent requests.

Using these SPLs, you can configure your SBC to comply with requirements, manipulate signaling, and time traffic to inter-operate with SIP trunks:

- The surrogate registration SPL feature applies to both NTT and KDDI environments. The associated random contact exclusion child feature, however applies only to KDDI environments.
- The NTT Message coverter SPL feature applies only the NTT environments.

Depending on version, these packages are built into the SBC base code, in which case you don't have to upload them.

Surrogate Registration

When interoperating with KDDI or NTT trunks, use the following `spl-options` to support this surrogate registration function:

- dyn-contact-start—Configure on the sip-interface facing the CUCM realm.
- dyn-contact-method=randomseed—Configure on the sip-interface facing the NTT or KDDI trunk.

When configured, the SBC:

- Sends the IP address of the egress sip-interface in the host part of the Call-ID within SIP INVITE requests to NTT or KDDI.
- Removes the string “sbc” from the host part of the Call-ID while sending out de-REGISTER requests.
- Adds the egress `sip-interface` IP address to the host part of the call-id.
- Sends two de-REGISTERs messages with at least a 1 second interval for the same AOR.

SPL functions persist after a reboot.

Excluding the Random Contact Validation from the Surrogate Registration SPL Feature

You can configure an SPL option on the SBC to disable incoming INVITE validation. This feature makes use of the Control-Surr-Reg SPL, requiring the additional configuration noted below. When you configure this feature, the SBC does not attempt to match the incoming R-URI against the random user part received while performing the Surrogate Registration SPL feature processing.

Although the surrogate registration SPL feature applies to both NTT and KDDI environments, this contact validation exclusion feature applies only to KDDI environments.

As stated above, when you deploy the surrogate registration feature using SPL, the SBC compares the random contact generated during registration against the value of the user part

of incoming Invite's RURI. If this comparison fails, the SBC suppresses this call. Some deployments, however, need to use the SPL surrogate registration process without this validation.

To skip this validation, you set the **dial-inservice-validation** SPL option on the ingress **realm-config**. The syntax for setting the **dial-inservice-validation** option follows.

```
ORACLE(realm-config)# +spl-options +dial-inservice-validation
```

In addition, configure the same options described for the surrogate registration feature on the applicable realm-config elements. This results in these options set on both the applicable sip-interface and realm-config:

- dyn-contact-start—Configure on the realm-config facing the CUCM realm.
- dyn-contact-method=randomseed—Configure on the realm-config facing the KDDI trunk.

When you include this configuration with the Surrogate Registration SPL feature, the SBC continues to perform surrogate registration, sending REGISTER messages to KDDI and processing the 200 OK responses. The difference is that the SBC accepts corresponding INVITE messages, even when the user part of RURI in the INVITE does not match the stored user part of the 200 OK's contact header.

Configure Surrogate Registration

You configure this surrogate registration SPL feature on the realm-config or sip-interface facing the trunk. Configure the random contact exclusion on the ingress realm-config for KDDI environments only.

1. Access the **sip-interface** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

2. Select the **sip-interface** object to edit.

   ```
   ORACLE(sip-interface)# select
   <RealmID>:
   1: realm01 172.172.30.31:5060

   selection: 1
   ORACLE(sip-interface)#
   ```

3. If using this surrogate registration SPL configuration for a trunk, configure this option on the sip-interface facing the NTT or KDDI trunk.

   ```
   ORACLE(sip-interface)# spl-options +Control-Surr-Reg
   ```

4. If using this surrogate registration SPL configuration for a trunk, Configure this option on the sip-interface facing the CUCM realm.

   ```
   ORACLE(sip-interface)# spl-options +dyn-contact-start
   ```

5. If using this surrogate registration SPL configuration for a trunk, configure this option on the sip-interface facing the NTT or KDDI trunk.

```
ORACLE(sip-interface)# spl-options +dyn-contact-method=randomseed
```

6. Type **done** and save your work.

> **Note:**
>
> The next steps, applicable to the contact validation exclusion features, applies only for KDDI deployments.

7. If excluding the contact validation component of the SPL surrogate registration function for a KDDI trunk, move to the realm-config configuration prompt.
8. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

9. Identify the applicable ingress realm, on which you set this SPL option.
10. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

11.
12. If using this surrogate registration SPL configuration for a trunk, configure this option on the realm-config facing the CUCM realm.

```
ORACLE(realm-config)# spl-options +dyn-contact-start
```

13. If using this surrogate registration SPL configuration for a trunk, configure this option on the realm-config facing the KDDI trunk.

```
ORACLE(realm-config)# spl-options +dyn-contact-method=randomseed
```

14. If excluding the contact validation component of the SPL surrogate registration function for a KDDI trunk, type **spl-options +dial-inservice-validation** to apply this feature on the applicable ingress realm.

```
ORACLE(realm-config)# spl-options +dial-inservice-validation
```

15. Type **done** and save your work.

Configure Surrogate Registration for KDDI Environments

You configure this surrogate registration SPL feature on the **realm-config** or **sip-interface** facing the trunk. Configure the random contact exclusion on the ingress **realm-config** for KDDI environments only.

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. If using this surrogate registration SPL configuration for a trunk, Configure this option on the realm-config facing the CUCM realm.

```
ORACLE(realm-config)# spl-options +dyn-contact-start
```

4. If using this surrogate registration SPL configuration for a trunk, configure this option on the realm-config facing the KDDI trunk.

```
ORACLE(realm-config)# spl-options +dyn-contact-method=randomseed
```

5. If excluding the contact validation component of the SPL surrogate registration function for a KDDI trunk, type **spl-options +dial-inservice-validation** to apply this feature on the applicable ingress realm.

```
ORACLE(realm-config)# spl-options +dial-inservice-validation
```

6. Type **done** and save your work.

NTT Message Converter

When required by your service provider, set `spl-options` to `+NttMsgConverter`.

- ocNttMsgConverterTagging=opposite—Configure on the sip-interface facing the Unified Call Manager (CUCM) realm.
- dyn-contact-start—Configure on the sip-interface facing the CUCM realm.
- dyn-contact-method=randomseed—Configure on the sip-interface facing the NTT trunk.

To support these trunks, you load all of these SPLs on the appropriate `sip-interface` elements. When configured, these SPLs trigger the following functions by the SBC:

- As a part of the surrogate registration process, sends a unique, random user-info portion in every REGISTER request to the NTT SIP trunk as well as within outgoing INVITE messages for calls.
- Applies validity check to an incoming INVITE from the SIP trunk before sending out 100 TRYING and subsequent 1xx, 2xx messages. It is expected that the incoming INVITE Request-URI user portion contains the same randomized value that the SBC sent in the most recent REGISTER message to the trunk.
- In compliance with NTT requirements, sets the tag size of From and To headers in the SIP messages to be less than 32 bytes. This resolves issues with, for example, the tags sent by Avaya in originating SIP messages, which are approximately 51 bytes.
- In compliance with NTT requirements, sets the Rseq, Cseq, Session ID (in SDP) values less than 999900, and sets the SDP o line username character length to a maximum of 10 bytes. This resolves issues with, for example, messages from an Enterprise PBX with a large RSeq value in 18x messages. This also resolves issues with the SDP o line generated by some Enterprise PBXs, which have a username that is 19 bytes.
- Checks the RURI user portion of incoming CANCEL request for the AoR and compares it with the AoR specified in the Request-URI of the initial INVITE received. If the value is different, the SBC responds with a 481 Call/Transaction Does Not Exist.

> **Note:**
>
> These features make use of the SurrogateContact SPL. When using this SPL, triggered by the configurations described below, the system supports only one surrogate agent within the entire configuration.

Configure the CUCM Facing SIP Trunk Support Options

Configure these SPL options on the CUCM facing `sip-interface`.

To configure the SPL option:

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Type **spl-options +ocNttMsgConverterTagging=opposite** and press Enter.

```
ORACLE(sip-interface)# spl-options +ocNttMsgConverterTagging=opposite
```

3. Type **spl-options +dyn-contact-start** and press Enter.

```
ORACLE(sip-interface)# spl-options +dyn-contact-start
```

4. Type **done** and save your work.

Configure the Trunk Facing SIP Trunk Support Option

Configure these SPL options on the trunk facing `sip-interface`.

To configure the SPL option:

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Type **spl-options +dyn-contact-method=randomseed** and press Enter.

```
ORACLE(sip-interface)# spl-options +dyn-contact-method=randomseed
```

3. Type **spl-options +ocNttMsgConverterTagging=enabled** and press Enter.

```
ORACLE(sip-interface)# spl-options +ocNttMsgConverterTagging=enabled
```

4. Type **done** and save your work.

Emergency Location Identification Number (ELIN) Gateway Support

An ELIN-capable gateway supports connection to a qualified E911 service provider. The connection supports PSTN-based E911 functions, including user callback when there is a disconnect. Enterprises often deploy ELIN numbers based on physical location to locate the physical source of a 911 call. By using multiple ELINs, an enterprise can support multiple, simultaneous E911 calls.

Typically, an enterprise purchases multiple ELIN numbers. An ELIN gateway replaces VoIP extension URIs with ELIN numbers and maintains the mapping. For example, if an emergency service replied to a VoIP URI without using an ELIN gateway, the reply would be delayed or fail. An ELIN gateway can use its mapping to translate the ELIN number back to the VoIP extension from within the enterprise session network. The gateway can immediately forward the call back to the original client.

The Oracle Communications Session Border Controller supports E911 ELIN for Lync-enabled Enterprises using the ELIN_Gateway SPL option. Enable this option in the global SPL configuration. The Oracle Communications Session Border Controller supports up to 300 ELIN numbers simultaneously and it can reuse numbers allowing a greater number of emergency calls.

> **Note:**
>
> This ELIN Gateway SPL is not supported by the Oracle Communications Session Router

How the Emergency Location Identification Number (ELIN) SPL Works

When a Lync client places a 911 emergency call through a mediation server to a Oracle Communications Session Border Controller (SBC), the server indicates the emergency status in the priority field and provides a list of ELIN numbers. When the ELIN gateway module is enabled, the SBC intelligently selects a particular ELIN number and uses it as the ANI in the “From” field SIP URI in the outgoing INVITE.

The SBC preserves the mapping of used ELIN numbers in an internal table. This table includes the ELIN number, the caller (VoIP extension), the “in-use” count, and a timer field. The SBC retains these mappings for a configurable time period ranging from 30 to 60 minutes after the call is terminated. The default is 30 minutes. When the timer expires, the entry is purged from the table. The timer field shows the time of day that the timer started.

You can view the current ELIN table at any time using the ACLI command `spl show sip elins`.

After the Lync client call is disconnected, the 911 service may call back using the number provided in the “From” field of the original INVITE. This presence of this number in its ELIN number table allows the SBC to route the call back to the original caller.

Number Reuse

The SBC can use an ELIN number for multiple calls. When a call that requires an ELIN mapping arrives at the SBC, it checks to see if the numbers presented by the mediation server are in use. If a number is not in use, it simply uses that number. A number is not in use if it is not in the table or its “used count” is 0. An entry’s used count is zero when its not in use and its purge timer has not yet expired.

If all numbers are in use, the SBC employs a means of reusing a number, incrementing its used count for each additional call. The selection process proceeds in the following order:

1. If the “caller” is in the ELIN table, the SBC selects that mapping.
2. The SBCselects the number with the lowest “ELIN count”.

If an ELIN number is used by multiple calls, it maps callback attempts to that ELIN number to the client that was last associated with the number.

Error Handling

Lync mediation servers always expect 503 “Service Unavailable” as an error message to a failed ELIN call. There is a variety of error messages that the network may send back when a call fails. For the purposes of Lync support, the SBC sends 503 “Service Unavailable” to indicate call failure to a mediation server, regardless of the error it receives.

PSAP Callback Options

When you enable the ELIN gateway, the SBC does not support Public Safety Answering Point (PSAP) callback handling to numbers that are not in the PSAP callback list, which includes 911, 112 and any number you have added. You can, however, further configure the SBC to support such callbacks for scenarios wherein the PSAP service does not use a known emergency number or uses "anonymous" as the FROM. You can also configure the SBC to replace the request-URI in a PSAP callback to resolve routing issues. You enable global SPL options for these configurations.

By default, the SBC creates entries in its ELIN table using the ELINs it finds in the PIDF and the FROM in an emergency INVITE. If the call terminates unexpectedly, the emergency service may make a PSAP callback towards the calling station. When the SBC receives a PSAP callback, it performs PSAP callback handling only if the callback's FROM is in its PSAP callback list. The SBC then uses the mapping of the ELIN number and the source number to determine where to send its INVITE by setting the TO, and, if configured, the request-URI.

When enabled, the **Elin-Ignore-PSAP-Source** SPL option modifies the processing flow the SBC uses to identify a PSAP callback and forward its ensuing INVITE. When the SBC receives an INVITE, it first checks to see if you have enabled this option.

Whether or not you have enabled **Elin-Ignore-PSAP-Source** the SBC first determines whether the call is an emergency, then whether the call is a PSAP callback:

- When **Elin-Ignore-PSAP-Source** is enabled, the SBC identifies an emergency call if the INVITE includes a Priority: emergency header and a destination to 911, 112 or a number you have added using **Elin-Add-PSAP** option.
- When **Elin-Ignore-PSAP-Source** is not enabled, the SBC identifies an emergency call only if the INVITE includes a Priority: emergency header.

If the call is not an emergency call, the default behavior checks the source. If the source is 911, 112 or a number you have added using **Elin-Add-PSAP** option, the system sends the call to its PSAP callback handling logic, ultimately forwarding the call using the ELIN mapping table to replace the TO from the ELIN to the mapped target number.

In contrast, if the system finds you have enabled **Elin-Ignore-PSAP-Source**, it sends the all calls that are not emergency calls to PSAP callback handling. At this point, further PSAP callback handling is the same whether or not you have configured **Elin-Ignore-PSAP-Source**. If there is no ELIN mapping table match, the system forwards the call as a normal INVITE.

Handling of PSAP callbacks always includes updating the To-URI from the ELIN to the target number. This handling optionally includes updating the request-URI to the target number. Applicable option definitions include:

- **Elin-Ignore-PSAP-Source**—Allows the SBC to forward a PSAP callback from any number, uncoupling the PSAP callback with the source of the incoming INVITE.
- **Elin-Modify-RURI**—Allows the SBC to change the request-URI to match the TO during a PSAP callback scenario. This can allow the system to successfully forward the callback by replacing the request-URI with the original TO.

These options are global. You configure these features using the **spl-options** branch under the global **spl-config**. The system does not offer these options within realm, interface or session agent elements.

Processing the Elin-Ignore-PSAP-Source SPL Option

When the SBC receives a call, it first determines whether you have enabled the **Elin-Ignore-PSAP-Source** option.

- If not, the SBC checks for a Priority:emergency header:
  - If true, the SBC performs emergency call handling and updates the ELIN mapping table based on this emergency call.
  - If not true, the SBC checks whether the source is 911, 112 or a PSAP number you configured with the **Elin-Add-PSAP** option.
    - If true, the SBC performs a PSAP call back and a "To" header modification.
      1. The SBC does or does not modify the request-URI, based on the state of the **Elin-Modify-RURI** option. (See below.)
      2. The SBC modifies the TO header.
      3. The SBC forwards the call.
    - If not true, the SBC treats this call as a normal call and forwards it without any modifications.
- If you have enabled the **Elin-Ignore-PSAP-Source** option, the SBC checks for Priority:emergency header and whether the destination is in the PSAP callback list (911, 112 or PSAP numbers you configured using the **Elin-Add-PSAP** option):

- If true, the SBC performs emergency call handling and updates the ELIN mapping table based on this emergency call.
- If not true, the SBC passes the call to PSAP callback and performs an ELIN table lookup:
  * If there is no match in the ELIN table, the SBC hands the call to local policy and forwards it as a normal call.
  * If there is a match:
    1. The SBC does or does not modify the request-URI, based on the state of the **Elin-Modify-RURI** option. (See below.)
    2. The SBC modifies the TO header.
    3. The SBC forwards the call.

    At this point, the SBC has supported PSAP callback without needing to know the call's source.

Processing the Elin-Modify-RURI SPL Option

Regardless of your **Elin-Ignore-PSAP-Source** setting, the SBC refers to the **Elin-Modify-RURI** option only if processing comes to the point where the system has found a match in the ELIN table. If so, the system would currently be supporting a PSAP callback based on an ELIN table lookup. At this point, the SBC checks to see if you have enabled **Elin-Modify-RURI**.

- If not, the SBC:
  1. Performs a routing lookup to identify routes.
  2. Updates the To header with the FROM in the ELIN table.
  3. Forwards the PSAP callback.
- If so, the SBC:
  1. Changes the request-URI to the FROM in the ELIN table.
  2. Performs a routing lookup to identify routes.
  3. Updates the To header with the FROM in the ELIN table.
  4. Forwards the PSAP callback.

> **Note:**
>
> This option has no impact on emergency call handling.

Configuration

You configure the **Elin-Ignore-PSAP-Source** and **Elin-Modify-RURI** SPL options using the **spl-options** branch under the global **spl-config**. You must also configure the system to use the ElinGateway SPL:

```
ORACLE(spl-config)#spl-options +Elin-Gateway=60
ORACLE(spl-config)#spl-options +Elin-Ignore-PSAP-Source
ORACLE(spl-config)#spl-options +Elin-Modify-RURI
```

PSAP Callback Call Flows

Call flows for PSAP callbacks include the contents of SIP messages and detail on the processing.

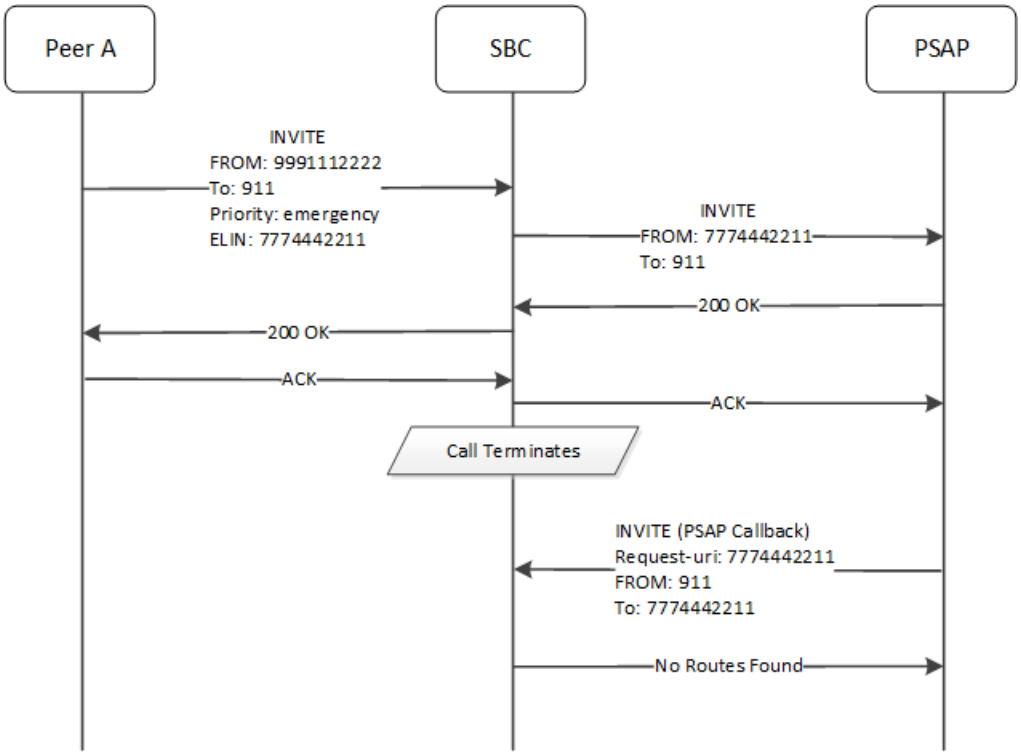
In all the cases below, the SBC modifies the “FROM” number in its INVITE to the PSAP to the ELIN number, creates an entry in the ELIN mapping table, and forwards the INVITE.

> **Note:**
>
> When you have enabled the **Elin-Ignore-PSAP-Source** option and a call arrives at the SBC with a “PRIORITY: emergency” header and a random, non-emergency number in the TO, the SBC would not handle this call as an emergency and not use the ELIN gateway function.

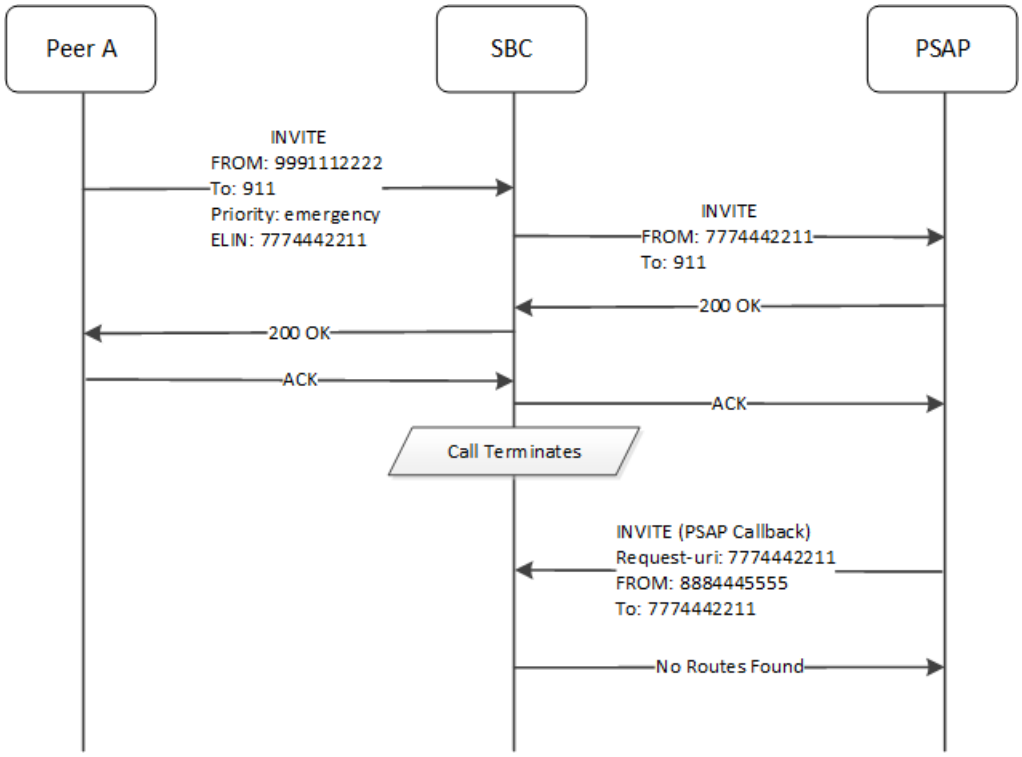
PSAP Callback with No Options Set

This call flow depicts the system not forwarding a PSAP call back because you have not configured **Elin-Modify-RURI**.

PeerA has sent an emergency call to 911. In addition, there is no local policy configured that matches a to-address with the ELIN number 7774442211. The SBC identifies the call from PeerA as an emergency call since it contains “PRIORITY: emergency” header. It modifies the “FROM” number in INVITE request to the ELIN number, and forwards the INVITE. The SBC receives the PSAP callback from 911, but replies with no route found because there is no local route that matches 7774442211.



This next call flow depicts the system not forwarding a PSAP call back despite the system having a **local-policy** with **to-address** of 9991112222. The flow is the same as above, with the exception that the PSAP has initiated a callback using a 8884445555 as the FROM.



The SBC receives the PSAP callback with 8884445555 as the FROM. The SBC does not find 8884445555 in its PSAP callback table because it has not been added using **Elin-Add-PSAP**. In addition, the **Elin-Ignore-PSAP-Source** option is not set, so the system exits PSAP callback handling and treats the call as a normal call. Ultimately, there is no route to the ELIN, 7774442211.

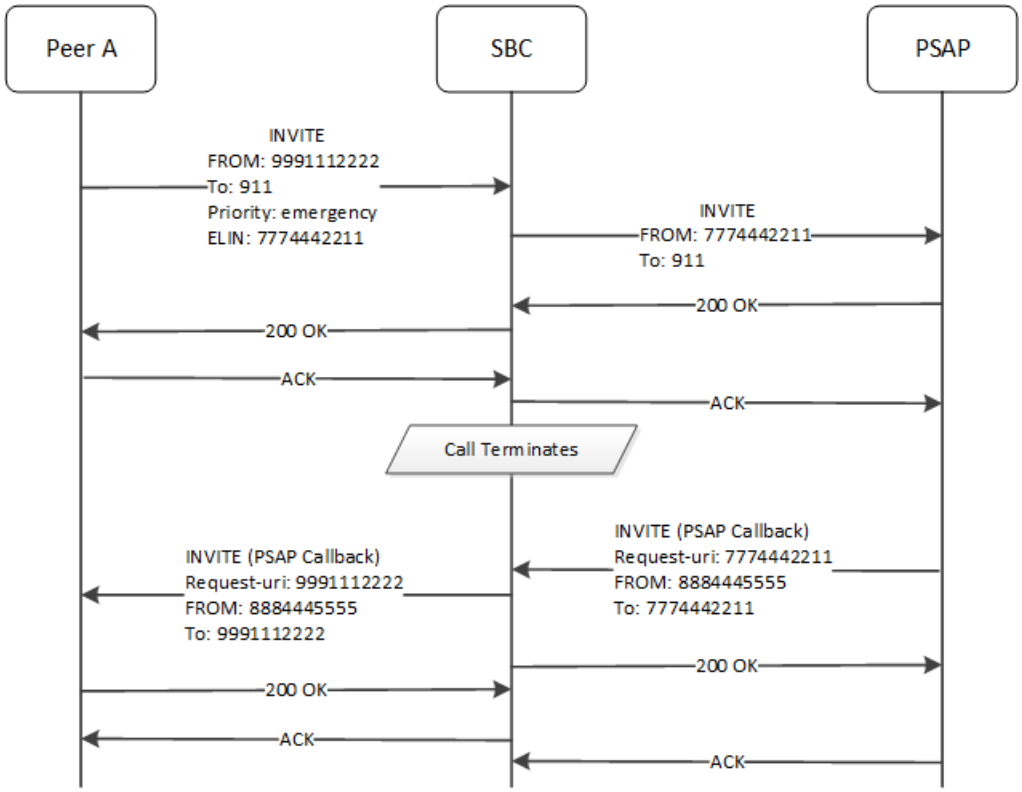
This call flow would succeed if both of the following were true:

- You had enabled the **Elin-Ignore-PSAP-Source** SPL option, which would have ignored the PSAP from and proceeded with performing an ELIN mapping table lookup.
- You had enabled the **Elin-Modify-RURI** SPL option, which would have replaced the request-uri with 9991112222.

PSAP Callback with Both Options Enabled

This next call flow depicts the system forwarding a PSAP call back using both the **Elin-Ignore-PSAP-Source** and **Elin-Modify-RURI** SPL options.

This flow is the same as the example directly above, with the exception that you have enabled both SPL options.



The SBC receives the PSAP callback with 8884445555 as the FROM. The SBC ignores this FROM because you have enabled **Elin-Ignore-PSAP-Source**. The SBC proceeds with PSAP callback handling. It finds a match for 7774442211 in the ELIN mapping table and proceeds. The SBC changes the request-uri in its subsequent INVITE to 9991112222 because you have enabled **Elin-Modify-RURI**. Standard ELIN handling also changes the TO header in its subsequent INVITE to 9991112222.

The use of both of these options allows this call flow to succeed.

Configure the ELIN Gateway Options

To enable an Emergency Location Identification Number (ELIN) gateway to support connections to an emergency service provider, you must configure the ELIN gateway option on the (SBC). Oracle delivers the SBC pre-configured with the 911 and 112 Public Safety Answering Point (PSAP) callback numbers. You can add more PSAP numbers, as needed. You can also specify the length of time that you want the SBC to retain ELIN mappings.

- Determine the preferred length of time, in minutes, that you want the SBC to retain ELIN mappings.
- Determine whether or not you want to add more PSAP callback numbers.
- The **Elin-Ignore-PSAP-Source** and **Elin-Modify-RURI** spl-options are available in the ElinGateway.1.8.spl version and higher. Determine your current plugin version to determine whether you need to remove any existing ELIN plugin configuration from the SBC and reboot before performing this configuration.

The SBC requires ELIN configuration at the global level, rather than at the session-agent, realm-config, or sip-interface level. Select the spl-config option under system for this ELIN configuration. In the following configuration you can set the time limit for retaining ELIN mappings and you can add more PSAP callback numbers.

To configure the ELIN Gateway option:

1. Access the **spl-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# spl-config
ORACLE(spl-config)#
```

2. Type **select**.
3. Type `spl-options +Elin-Gateway=<value>`.

   Valid values: 30 or 60.

```
ORACLE(spl-config)# spl-options +Elin-Gateway=60
```

4. (Optional) Type spl-options +Elin-Add-PSAP="<value>", where <value> is one or more PSAP numbers, and press Enter.

   For multiple numbers, place the numbers within quotes, separate the numbers with a comma, and use no spaces. A single number does not require enclosure in quotes. Examples: `+Elin-Add-PSAP=999` and `+Elin-Add-PSAP="999,000,114"`.

5. (Optional) Type spl-options +Elin-Ignore-PSAP-Source to ignore the source of the incoming request for a PSAP callback.
6. (Optional) Type spl-options +Elin-Modify-RURI to change the ELIN number in the request-URI of the outgoing INVITE to the FROM in the original INVITE and avoid route lookup problems.
7. Save and activate the configuration.

Comfort Noise Generation SPL

Comfort noise (CN) is the noise in a Real Time Transport Protocol (RTP) message (defined in RFC 3389) that is played to prevent a user from hearing completely dead silence on the connection. The Session Description Protocol (SDP) negotiates this RTP message containing the comfort noise using payload type 13 and an rtpname of "CN".

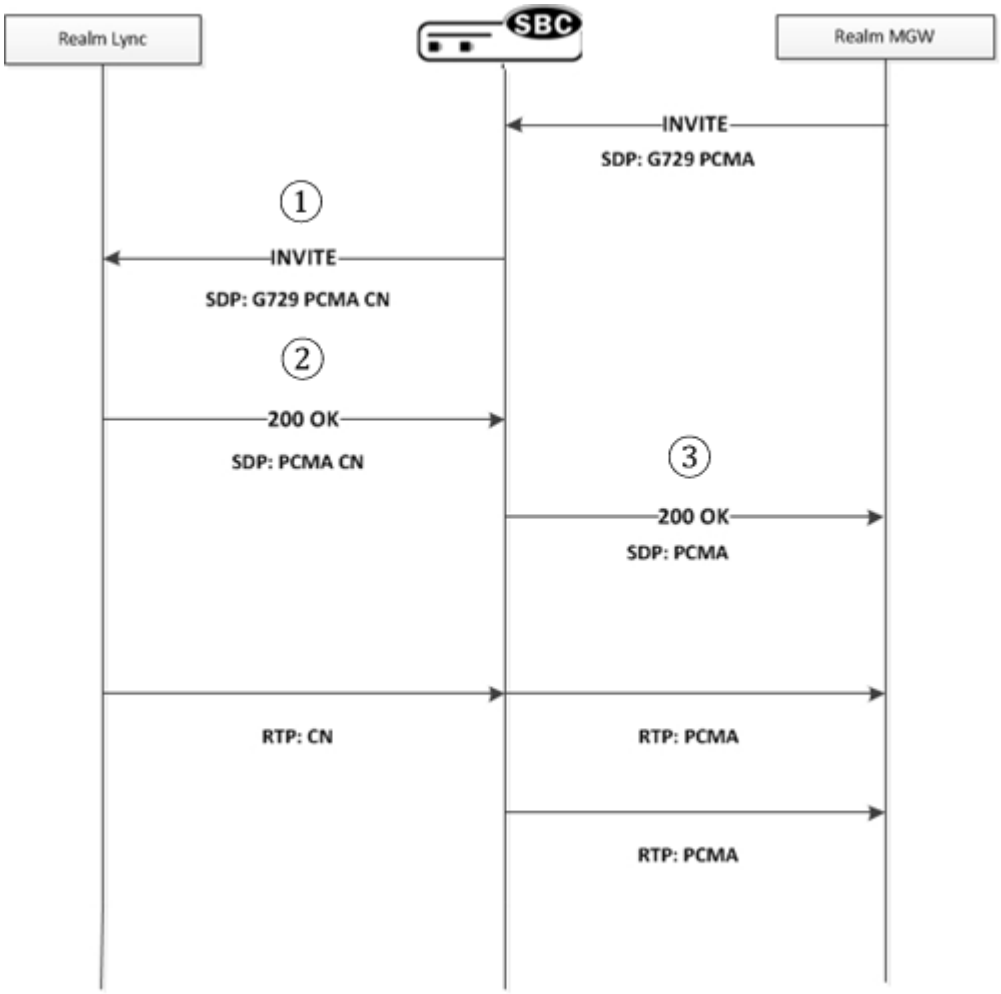
However, when CN is received, normal RTP ceases. Thus, with no RTP traffic, guard timers may trigger and cause the call to be terminated. To correct this, you can load a Comfort Noise Generation SPL that allows the Oracle Communications Session Border Controller (SBC) to generate "noise" RTP using the normal audio codec when it receives a CN indication.

After loading the SPL on the SBC, comfort noise is added and removed from the SDP to allow for proper negotiation. If properly negotiated in SDP, CN interworking facility (IWF) is enabled in the media flows allowing the SBC to generate noise RTP when CN is received.

> **Note:**
>
> CN generation configuration is supported on realms only.

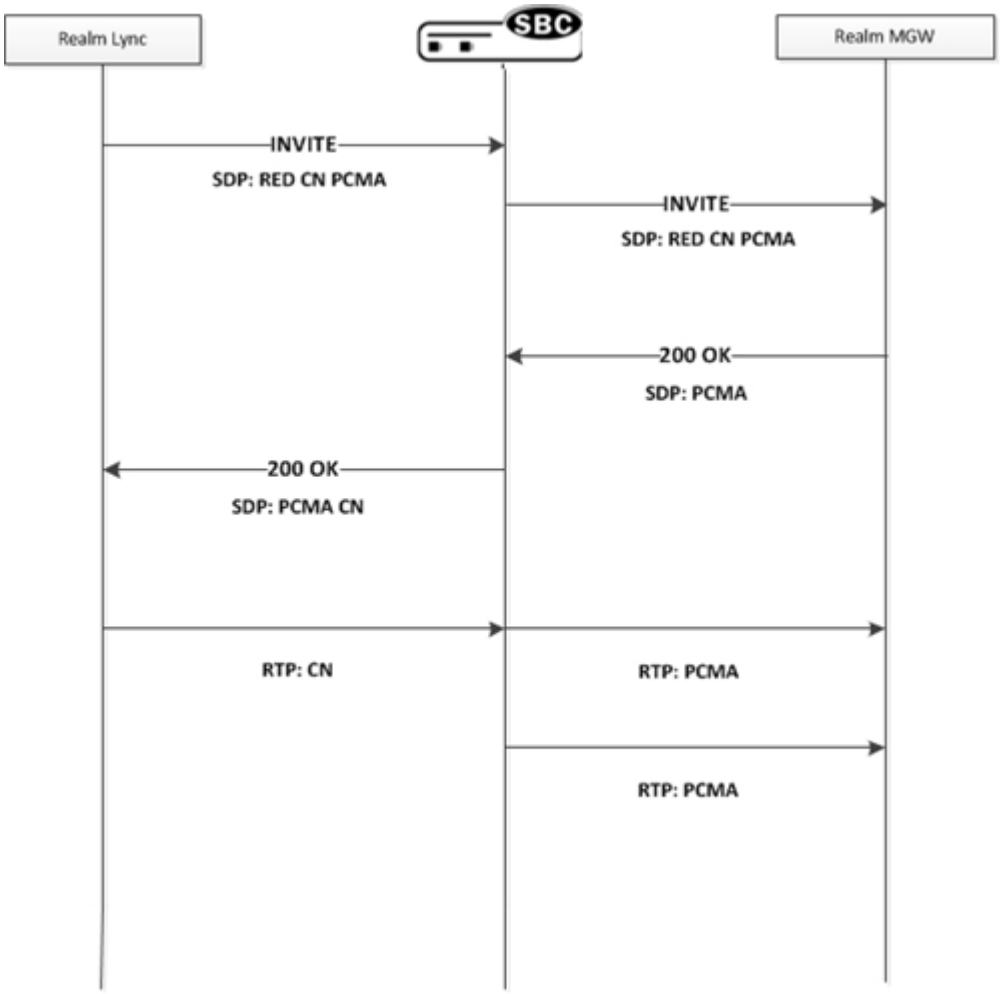
The following describes two different cases of how the SBC performs the SDP manipulation using the CN Generation SPL.

Case 1



1. SDP offer received from the realm that has comfort-noise-generate enabled. If the SDP offer contains CN, no IWF is required. If it does not contain CN, and at least one of the offered audio codecs is PCMU or PCMA, CN is added in outgoing SDP offer.
2. If SDP Answer contains the CN codec and topmost audio codec is PCMU or PCMA, SBCenables CN IWF.
3. SBC strips CN from outgoing SDP Answer.

Case 2



1. SDP offer is sent to a realm that has comfort-noise-generate enabled. If CN was not offered, IWF cannot be performed. If CN is in the offer, the SBC forwards the offer to the outbound side.
2. SDP Answer is received from a realm that has comfort-noise-generate enabled. If it contains CN, no IWF is required because both sides support CN. If there is no CN in the Answer, and the topmost audio codec is PCMU or PCMA, the SBC enables CN IWF.
3. If CN IWF is enabled, the SBC adds CN to the outgoing SDP Answer.

Configure the Comfort Noise Generation SPL

Use the following procedure to configure the CN Generation SPL on the SBC.

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Type **spl-options +comfort-noise-generate** and press Enter.

```
ORACLE(realm-config)# spl-options +comfort-noise-generate
```

3. Type **done** to save your work.

Example Configuration

The following is an example of a CN Generation SPL configuration.

```
media-manager
    realm-config
        identifier              SP
        addr-prefix             172.16.0.0/16
        network-interfaces      Core1:0
        mm-in-realm             enabled
        spl-options             comfort-noise-generate
```

High Availability (HA) Support

High Availability (HA) supports the use of Comfort noise. The codec encoding (PCMU/PCMA), codec ptime, and CN generation enabled/disabled state is exchanged with the standby in Middlebox Control Daemon (MBCD). If CN generation occurs in a call flow, and a switchover occurs, the CN generation stops until the next CN message is received.

Licensing Information

The following licensing applies to the CN Generation SPL.

Process	Description
Package name	G711
License category	Open Source
Package version	N/A
Vendor	Sun Microsystems
Applicable license	This source code is a product of Sun Microsystems, Inc. and is provided for unrestricted use. Users may copy or modify this source code without charge. SUN SOURCE CODE IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. Sun source code is provided with no support and without any obligation on the part of Sun Microsystems, Inc. to assist in its use, correction, modification or enhancement. SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY THIS SOFTWARE OR ANY PART THEREOF. In no event will Sun Microsystems, Inc. be liable for any lost revenue or profits or other special, indirect and consequential damages, even if Sun has been advised of the possibility of such damages. Sun Microsystems, Inc. 2550 Garcia Avenue Mountain View, California 94043

Process	Description
Software builds	SC[z] BC[z]640
Purpose	Conversion from linear samples to alaw/ulaw
Used in modified or unmodified form	Unmodified
Used internally or distributed	Distributed with Product Binaries
Location in source code tree	linux/kernel/modules/acme
Used by itself or in combination with other software	Used exclusively and tightly integrated into acme.ko. The code is compiled as part of acme.ko.
Link to website hosting Software	N/A
License requires notice provided in product documentation?	No

Request Validation SPL

You can configure the SBC to validate a specific set of requests and respond to these requests with the behaviors presented here when you enable the **ntt-request-valid** SPL option. This validation works using Surrogate Register SPL options within SurrogateRegister.spl and in conjunction with other NTT Message Converter SPL options. This processing compares values within the request, and only processes the call if they match. If they do not match, the SBC replies with responses specific to each scenario.

To implement this feature, you use your preferred configuration method to enable the following mandatory options in the applicable realm, which are available from the SurrogateRegister.spl:

- access realm—dyn-contact-start
- core realm—dyn-contact-method=randomseed, ntt-request-valid

You enable the **ntt-request-valid** option in the core **realm-config** to validate requests and, if that validation fails, reject requests within the following scenarios:

- An INVITE or re-INVITE received from NTT has a different user part in the Request-URI user info than the SBC received in the contact header of the 200 OK in the most recent REGISTER from NTT. If this check fails, the SBC sends a 404 response to the sender.
- Any of the following requests received from NTT have a different user part in the Request-URI user info than the SBC received in the 200 OK of the most recent REGISTER from NTT:
  - PRACK
  - CANCEL
  - ACK
  - BYE

  If this check fails, the SBC drops the call without any response.

- The top Via header in an INVITE or re-INVITE received from NTT does not use the IP to which surrogate registration was sent. If this check fails, the SBC responds with 403 response.

Additional detail on each of these request scenarios is provided below.

INVITE and Re-INVITE Check Priority

In addition to the checks implemented as a part of this feature, the SBC performs an IP check on the layer 3 source IP address. When configured, the SBC performs these checks on INVITEs and Re-INVITEs in a specific order, and takes action based on whether or not the IPs match.

1. Top Via header check—If this check fails, the SBC drops the call without any response.
2. L3 source IP check—If the layer 3 source address is not the IP to which surrogate registration was sent, the SBC sends a 403 response to the sender.
3. Request-URI check—If this check fails, the SBC sends a 404 response to the sender.

If all checks succeed, the SBC proceeds with the call.

Configuration

You enable this feature by setting the following three spl-options:

- access realm: **dyn-contact-start**
- core-realm: **dyn-contact-method=randomseed**, **ntt-request-valid**

```
ORACLE(realm-config)#spl-options +ntt-request-valid
ORACLE(realm-config)#spl-options +dyn-contact-start
ORACLE(realm-config)#spl-options +dyn-contact-method=randomseed
```

Validating the Request-URIs in INVITEs and re-INVITEs

This validation compares the contact header in the 200 OK of the sender's REGISTRATION with the RURI of the INVITE message from NTT. You configure the following **spl-options** to generate an effective random contact value in the SBC.

- access-realm: **dyn-contact-start**
- core-realm: **dyn-contact-method=randomseed**

If you do not configure the **ntt-request-valid** option, the SBC drops a call without any response when the effective random contact generated in the 200 OK's contact header does not match the RURI of the INVITE message from NTT. When you configure this option and the validation fails, however, the SBC sends a 404 response to NTT without a reason header.

If the values match, the SBC continues to process the call.

> **Note:**
>
> Based on standard SBC operation, when the top via and the sender do not match, the SBC adds a "received= <source ip>" to the via header internally. In a scenario where the L3 source IP is NTT and the top via is not NTT IP, the via header check still passes because the "received=" field contains and NTT IP.

Validating the Request-URIs in PRACKs, CANCELs, ACKs and BYEs

This validation compares the Request-URI user info in the request and the 200 OK of the REGISTER. These request validations require that you configure only the **ntt-request-valid** option in the SurrogateRegister.spl, but applicable deployments typically also have all the NTT Message Converter options configured.

If you do not configure the **ntt-request-valid** option, the SBC performs no validation and does not reject requests regardless of the user-info. When you configure this option and the R-URIs do not match, however, the SBC drops requests that fail this validation and does not provide any response.

If the values match, the SBC continues to process the call.

> **Note:**
>
> If the SBC drops a BYE, the PBX ultimately sends its own BYE to the SBC, which terminates the call.

Validating the Via Header in INVITEs and re-INVITEs

This validation compares the top VIA header with the sender's IP in INVITEs or re-INVITEs received from NTT. The check is independent of the INVITE/re-INVITE R-URI validation described above. These request validations require that you configure only the **ntt-request-valid** option in the SurrogateRegister.spl, but applicable deployments typically also have all the NTT Message Converter options configured.

If you do not configure the **ntt-request-valid** option, there is no validation check and the SBC simply proceeds with the call. When you configure this option and the VIA header does not match, however, the SBC drops requests that fail this validation and does not provide any response.

The SBC identifies the sender's IP as follows:

- You have configured the sender's **session-agent** with static IP.
- The **hostname** for this **session-agent** is configured in the **registerHost** parameter in the **surrogate-agent**.

Consider the example configuration:

- You have configured the **register-host** parameter on the applicable **surrogate-agent** to the value "ipvoice.jp".
- You have configured the **hostname** parameter on the applicable **session-agent** to the value "ipvoice.jp", and the **ip-address** parameter to 10.180.111.1.

The SBC identifies the **session-agent** using the **register-host** parameter of the **surrogate-agent** and the **ip-address** of that **session-agent**, 10.180.111.1, as the sender's address.

> **Note:**
>
> For NTT deployments, this sender's address is often called "NTT". which is typically a single **surrogate-agent**.

Suppression of Extraneous 18x Messages

You can configure the SBC to suppress some provisional 18x, specifically 180 or 183, messages from a UAS within call setup transactions to reduce excess signaling traffic. The system suppresses all of these 18x messages until it receives a 200 OK from the UAC. You configure this feature using an SPL option within the SuppressAdditionalProvisional SPL.

After the SBC receives an initial 18x message, subsequent 18x messages may only increase the amount of traffic on the UAC network without furthering a call setup. When configured, however, the SBC is capable of forwarding the first 18x to the client and responding to subsequent 18x, specifically 180 or 183, messages locally, thereby offloading the network from that traffic.

Examples of environments that benefit from this configuration includes MS Teams Direct Routing and AVAYA IP-PBX environments, which typically issue multiple 18x responses towards the caller while waiting for subsequent call setup signaling. Without this feature, the SBC forwards all of these 18x responses.

You enable this feature by configuring the SPL option **Drop-Additional-Provisional** within a **session-agent**, **realm-config**, **sip-interface**, and/or the **system** elements. If the SBC encounters any enabled **Drop-Additional-Provisional** SPL option, it triggers the feature for that agent, realm or interface. If you enable the SPL option at the **system** level, you enable the feature for all traffic.

With respect to provisional responses, the SBC normally monitors this traffic for 180 responses that include the 100rel header. When you enable this feature, it performs a similar function, forwarding the first 180 response (with 100rel) to the other side, then suppressing the subsequent 180 responses (with 100rel). In addition, the SBC sends a PRACK towards the UAS side using its +100rel interworking feature.

Additional operational behavior includes:

- The SBC suppresses all subsequent 18x responses with or without 100rel, sends a PRACK to the UAS, and accepts the subsequent 200OK for 100rel 180s.
- If the SBC receives a 18x message that it needs to suppress includes a 100rel tag in a Required header, the SBC sends a PRACK for that message with the 100rel Required tag, assuming proper configuration.
- Deployments using this feature can accept a 200 OK with a different ‘To’ tag than the one sent in the first 180 Ringing. For example, consider the scenario wherein the SBC sends a 18x to a UAC with ‘to’ tag = t1, followed by subsequent 18x messages with the 'To' tags – ‘t2’, ‘t3’, ‘t4’. This feature suppresses these subsequent 18x messages. If a 200 OK then arrives at the SBC with the 't4' ‘To’ tag = ‘t4’, both the SBC and the UAC accept this 200 OK.

Required Conditions

This feature imposes the following requirements for proper operation:

- The incoming INVITE must include SDP.
- The egress side of the SBC must send SDP in the 200 OK, even if SDP was included in the 18x.
- Early media is not supported within the context of this feature operation because applicable 18x messages may include SDP.
- This feature does not work within forked call flows.

Feature Configuration

When you enable the **Drop-Additional-Provisional** SPL option, the system automatically loads the SuppressAddionalProvisional.spl.

```
ORACLE(session-agent)#spl-options +Drop-Additional-Provisional
```

Use the syntax below to disable this feature.

```
ORACLE(session-agent)#spl-options -Drop-Additional-Provisional
```

Related Configuration

The following additional features may be required for this feature:

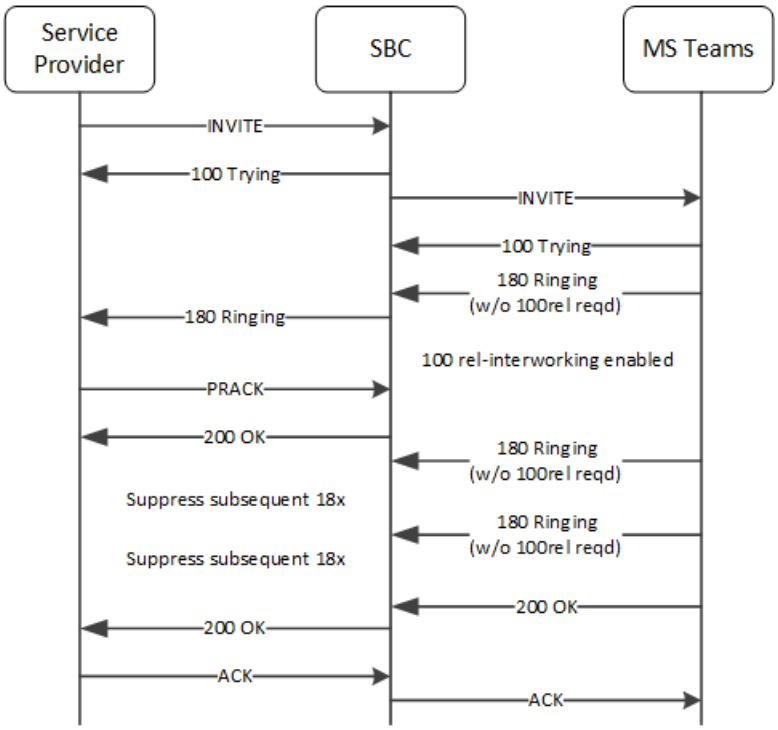
- If you need to support 18x suppression for scenarios that include the 100rel tag in a Require header, you must enable the **100rel-interworking** option on the applicable **sip-interface**.
- If you are using the SuppressAdditionalProvisional SPL loaded on an SBC version that does not support this feature, and are upgrading to a version that does, remove this suppression SPL manually and reboot your system before you perform this upgrade.

Call Flows for 18x Suppression

This section provides call flow diagram on this 18x suppression feature.

Scenario-1: MS Teams – Media Bypass Enabled, single device login

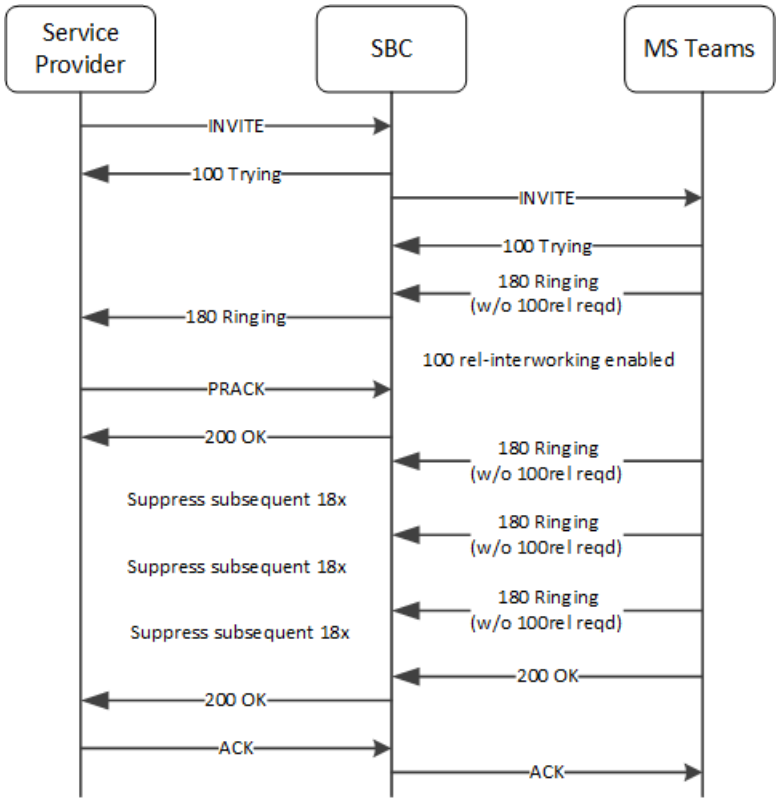
This scenario includes the SBC forwarding the first 180 response to the UAC side (Service Provider) and then suppressing subsequent 180 responses received from the UAS side (MS Teams). This scenario requires that you enable the **Drop-Additional-Provisional** configuration in the **spl-options** on the egress side.



Scenario-2: MS Teams – Media Bypass Enabled, multiple device login

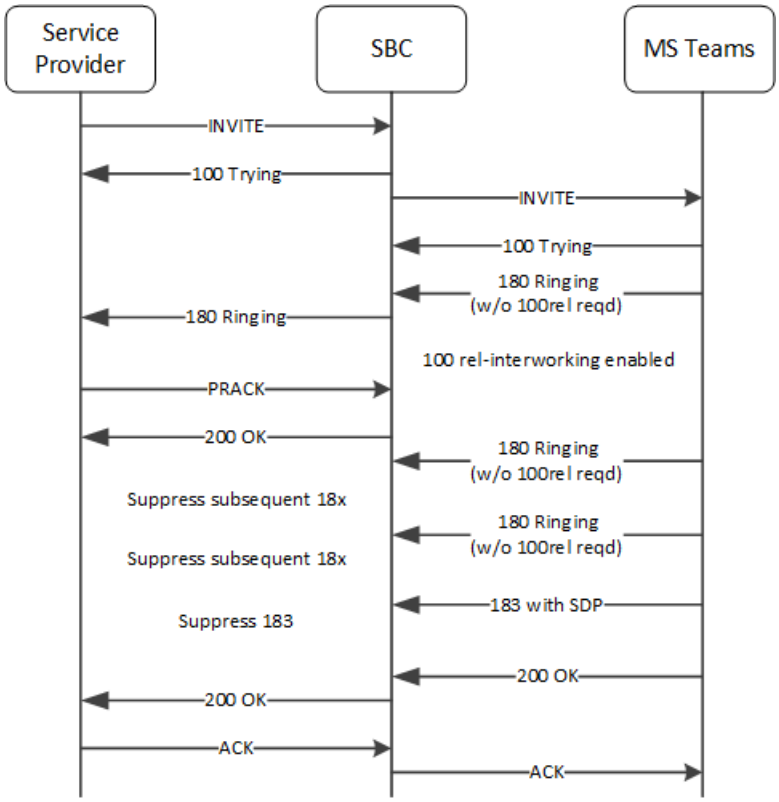
In this scenario, the SBC forwards the first 180 to the PSTN from MS-Teams and suppresses all subsequent 180 messages. In this case, there are multiple endstations responding to the INVITE within the MS Teams domain.

Any device that previously send a 180 can send the 200 OK. If a 200 OK comes from the device where the respective 180 ringing was suppressed, the SBC forwards the 200 OK to the service provider as soon as it receives it from MSTeams. In this case, the service provider must be configured to be aware that the 200 OK may have a “to:” tag that does not match the previous 180/183 received.



Scenario-3: MS Teams – Media Bypass Disabled

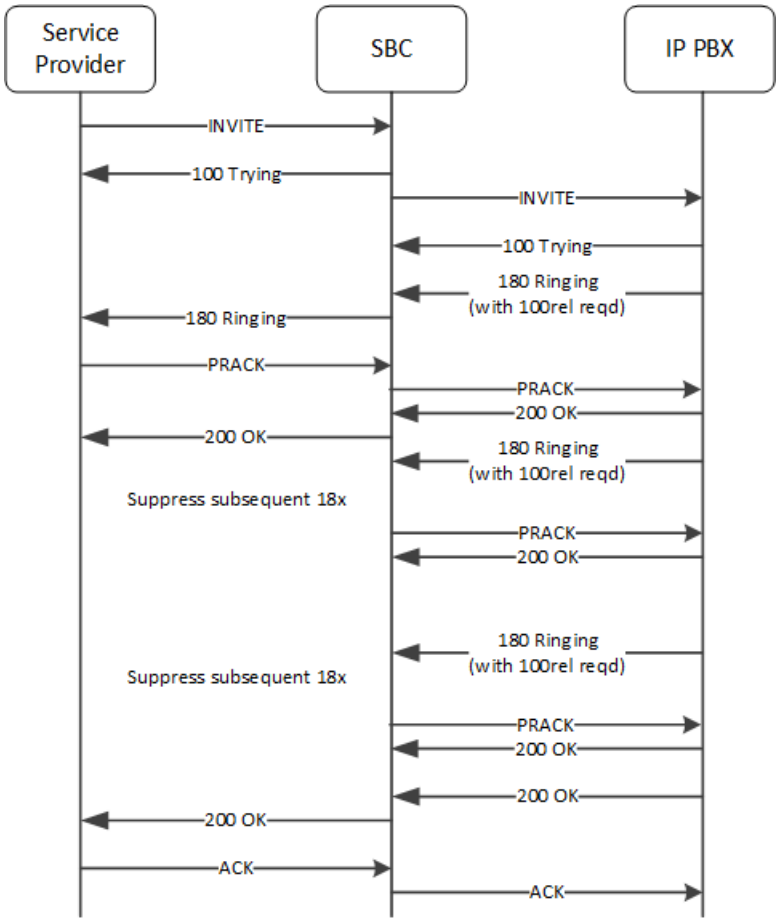
This scenario includes the SBC forwarding the first 180 response to the UAC side (Service Provider) and then suppressing subsequent 180 responses, including 183 response from the UAS side (MS Teams). This scenario requires that you enable the **Drop-Additional-Provisional** SPL option on the egress side.



Scenario-4: 180 Response received with 100rel required

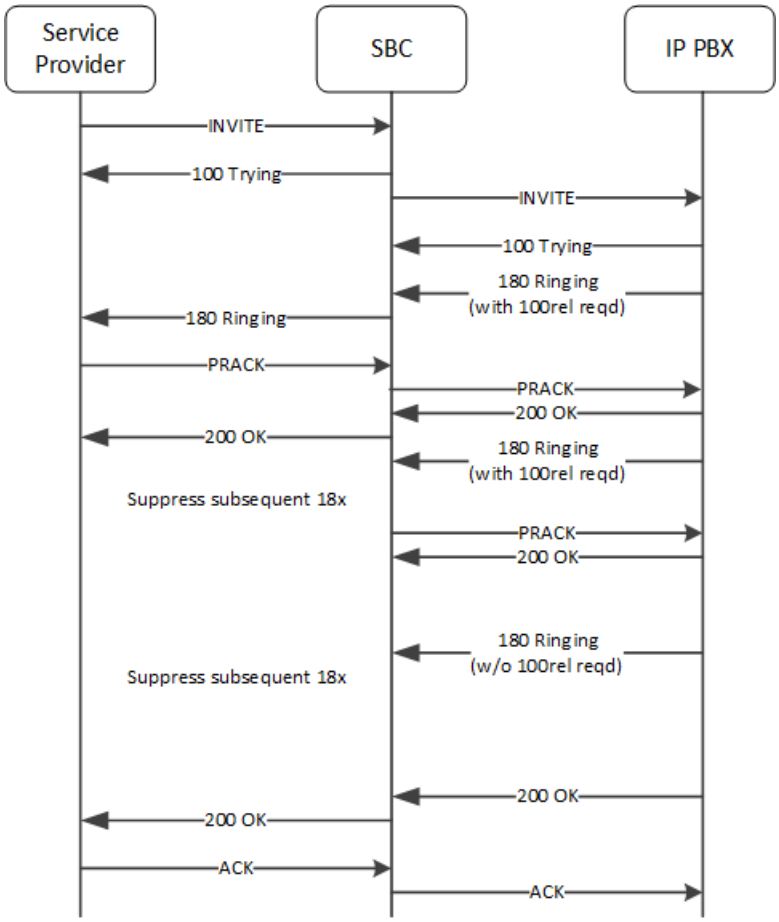
In this scenario, the SBC forwards only the first 180 with 100rel required while suppressing the subsequent 180s. In addition, the system sends PRACKs to the IP-PBX for the suppressed 180s, and accepts the 200 OKs for those PRACKs.

This scenario requires that you enable **100rel-interworking** on the egress **sip-interface** to manage the 180/PRACK sequences with the IP PBX. Similarly, you must enable **100rel-interworking** on the ingress **sip-interface** to manage the 180/PRACK transaction with the service provider.



Scenario-5: 180 Responses received with and without 100rel required

This flow is the same as Scenario 4 with the exception that it includes 180 messages with and without 100rel required. The difference between the scenarios is that the SBC does not need to manage the PRACK/200 OK transactions when the 180 does not require 100rel. The configuration requirements are the same as for Scenario 4.


Service Provider
SBC
IP PBX
INVITE
100 Trying
INVITE
100 Trying
180 Ringing
(with 100rel reqd)
180 Ringing
PRACK
PRACK
200 OK
200 OK
180 Ringing
(with 100rel reqd)
Suppress subsequent 18x
PRACK
200 OK
180 Ringing
(w/o 100rel reqd)
Suppress subsequent 18x
200 OK
200 OK
ACK
ACK

20 Transcoding

Transcoding is the ability to convert between media streams that are based upon disparate codecs. The Oracle Communications Session Border Controller supports IP-to-IP transcoding for SIP sessions, and can connect two voice streams that use different coding algorithms with one another.

This ability allows providers to:

- Handle the complexity of network connections and the range of media codecs with great flexibility
- Optimize bandwidth availability by enforcing the use of different compression codecs
- Normalize traffic in the core network to a single codec
- Enact interconnection agreements between peer VoIP networks to use approved codecs

By providing transcoding capabilities at the network edge rather than employing core network resources for the same functions, the Oracle Communications Session Border Controller provides cost savings. It also provides a greater degree of flexibility and control over the codec(s) used in providers' networks and the network with which they interconnect.

In addition, placing the transcoding function in the Oracle Communications Session Border Controller and at the network edge means that transcoding can be performed on the ingress and egress of the network. The Oracle Communications Session Border Controller transcodes media flows between networks that use incompatible codecs, and avoids back-hauling traffic to a centralized location, alleviating the need for multimedia resource function processors (MRFPs) and media gateways (MGWs) to support large numbers of codecs. This maximizes channel density usage for the MRFPs and MGWs so that they can reserve them for their own specialized functions.

Transcoding Resources

The computing resources used to transcode media come in one of three forms. When deploying an SBC image, only one of these transcoding resource types may be used per system.

The forms of transcoding resources are:

- Hardware-based transcoding for physical SBC platforms
- Software-based transcoding for vSBC platforms (Genuine Intel CPUs only)
- Artesyn PCIe card-based transcoding for vSBC platforms.

See the Transcoding Support section in the *Release Notes* for a list of the codecs supported by each of the 3 resource types, and any limitations.

Hardware-based Transcoding Resources

Acme Packet hardware is provisioned with DSP resources that enable transcoding on the Oracle Communications Session Border Controller. Transcoding capacity depends on the codecs in-use and the number of transcoding modules installed in the system. Capacity scales

linearly with each additional transcoding module installed. The number of DSP modules that can be installed is platform-dependent.

- Acme Packet 6300 and 6350: maximum of 48 DSP modules per system; 1 or 2 (24 DSP) TCUs may be installed in each system
- Acme Packet 4600: maximum of 12 DSP modules
- Acme Packet 3950/4900: maximum of 8 DSP modules
- Acme Packet 3900: maximum of 5 DSP modules

Transcodable Codecs

Refer to the Release Notes' Transcoding Support topic for a list of the transcodable codecs in this release and which platforms they are supported on. The codec names listed in the table in the Release Notes reflect the default media profiles for their given names.

When creating an override media profile for a codec, the system ignores case sensitivity. Also, GSM = GSM-FR.

Transcodable Codec Details

The following table lists the supported codecs, bit rates, RTP payload type, default ptime, and supported ptimes. See the Release Notes for which transcoding platforms support specific codecs.

vSBCs support a subset of the codecs in the table below. Refer to your version's Release Notes to see which codecs are transcodable on vSBCs. In addition, the maximum supported ptime for the vSBC is 60 msec. The ptimes in the table below that are greater than 60 msec, for example for the G723, G729, AMR, AMR-WB, OPUS and SILK codecs, apply only to the SBC when deployed over physical platforms.

Codec	Supported Bit Rate (kbps)	RTP Payload Type	Default Ptime (ms)	Supported Ptime (ms)
G.711 PCMU	64	0	20	10, 20, 30, 40, 50, 60
G.711 PCMA	64	8	20	10, 20, 30, 40, 50, 60
G.722	48, 56, 64	9	20	10, 20, 30, 40 For Acme Packet 1100, 3900 and virtual platforms, supported Ptimes include 20, 40 and 60 only
G.723.1	5.3, 6.3	4	30	30, 60, 90
G.726	16, 24, 32, 40	2, 96-127	20	10, 20, 30, 40, 50
iLBC	13.33	96-127	30	20, 30, 40, 60
	15.2	96-127	20	20, 30, 40, 60
G.729/A/B	8	18	20	10, 20, 30, 40, 50, 60, 70, 80, 90
AMR	4.75, 5.15, 5.90, 6.70, 7.40, 7.95, 10.2, 12.2	96-127	20	20, 40, 60, 80, 100
AMR-WB (G.722.2)	6.6, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05, 23.85	96-127	20	20, 40, 60, 80, 100
GSM FR	13	3	20	20

Codec	Supported Bit Rate (kbps)	RTP Payload Type	Default Ptime (ms)	Supported Ptime (ms)
T.38	4.8, 9.6, 14.4	N/A		10, 20, 30
Opus	For PNF or HW-SBC, the supported bitrate is: • 8-12 kbit/s for NB speech • 16-20 kbit/s for WB speech For VNF or vSBC, the full range of bitrate is supported for Opus, from 8 to 128 kbit/s	104	20	10, 20, 40, 60, 80, 100
EVRC	0.8, 4.0, 8.55	96-127	20	20, 40, 60
EVRC0	0.8, 4.0, 8.55	96-127	20	20
EVRC1	4.0, 8.55	96-127	20	20
EVRCB	0.8, 2.0, 4.0, 8.55	96-127	20	20
EVRCB0	0.8, 2.0, 4.0, 8.55	96-127	20	20
EVRCB1	4.0, 8.55	96-127	20	20
SILK	6.0 to 40	96-127	20	20, 40, 60, 80, 100
EVS Primary mode	5.9 to 128	96 - 127	20	20, 40 and 60
EVS AMR-WB IO mode	6.6 to 23.85	96 - 127	20	20, 40, and 60

Note:

If you set the value for the transcoding capacity as 0, then SBC considers the transcoding capacity as unlimited. But if you set the value greater than zero, Oracle Communications Session Border Controller considers the transcoding capacity as maximum license capacity. This is applicable for all supporting transcoding codecs.

See the EVS Supported Options section for EVS information.

See theSILK Codec Transcoding Support section for SILK information.

T.38 FAX Support

The Oracle Communications Session Border Controller (SBC) supports T.38 FAX relay (Version 0) conversion to T.30 over G.711 and supports FAX modulation schemes up to 14400 kbps V.17. The SBC does not support V.34 modulation, at this time.

Software-based transcoding

The Oracle Communications Session Border Controller supports media transcoding on virtual platforms. Refer to the Transcoding Support section of the Release Notes for the list of codecs which may be transcoded with software-based transcoding resources.

Transcoding is the process of converting voice audio streams from one encoding format (codec) to another. In addition to conversion between codecs, the SBC can also reframe compressed audio streams from one packet size to another (e.g. 10ms G.729 reframed to 30ms G.729) according to packetization times specified in session establishment. The SBC

may then convert between any supported codecs and frame size combination to another supported codec and frame size combination.

Software-based transcoding is configured identically to hardware-based transcoding, and is invoked when codec policies are configured but no transcoding hardware is recognized in the system. Software-based transcoding on virtual platforms is only supported on Intel CPU infrastructure.

Software-based transcoding alarms and traps

SNMP Traps

The apSysMgmtGroupTrap trap is sent with the MIB OID apSysXCodeG729Capacity to alert you of high G.729 Royalty codec usage. This MIB object is defined in `ap-smgmt.mib`. It is sent when utilization rises above 95% of licensed capacity. It is cleared when utilization falls below 80% of licensed capacity. The MIB object appears as:

```
apSysXCodeG729Capacity OBJECT-TYPE
        SYNTAX          SysMgmtPercentage
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
                "The percentage of licensed G729 transcoding utilization"
        ::= { apSysMgmtMIBGeneralObjects 35 }
```

Alarms

The G729 transcoding utilization alarm is triggered when utilization rises above 95% of licensed capacity. It is cleared when utilization falls below 80% of licensed capacity. The alarm appears as follows on the ACLI:

```
ID      Task      Severity      First Occurred          Last Occurred
131159  527739792       6       2011-10-11 10:11:49     2011-10-11 10:11:49
Count   Description
1       G729 Transcoding capacity at  97 (over threshold of 95)
```

Debugging log files

The `log.media` log file records host based transcoding events based upon logging level.

Adaptive Jitter Buffers for Transcoding Flows on vSBCs

The processing of transcoded flows on the SBC uses an adaptive jitter buffer. This feature allows the transcoding function to adapt to changes in network conditions and packet jitter. But if necessary, the jitter buffer feature (on virtual SBC platforms only) can also be adjusted to better align to specific network conditions.

The feature establishes two operation modes for sessions that include transcoding, Low-Jitter and High-Jitter. All sessions start in Low-Jitter mode, and can transition to High-Jitter mode and back again based on network conditions. Each mode has a minimum and maximum buffer depth (measures in milliseconds) that drives jitter buffer operation.

During voice calls, the system transitions between low-jitter mode and high-jitter mode when it detects:

- A short burst of very high-jitter packets (low-to-high level transition)

- A longer trend of high-jitter packets (low-to-high level transition)
- A very-long trend of low-jitter packets (high-to-low level transition)

The system avoids cases wherein packet loss occurs without any jitter, or cases wherein certain codecs use silence suppression and may not transmit packets over extended intervals.

Jitter Buffer Override Options

If necessary, you can override the Low-Jitter and High-Jitter minimum and maximum buffer depths. A larger/deeper jitter buffer can handle more jitter, but increases end-to-end latency of the call audio, which can result in noticeable conversational difficulty. Oracle recommends changing default jitter levels in conjunction with Oracle support.

The SBC provides options configurations for changing default jitter buffer levels, which reside in the **media-manager**. Adaptive Jitter Buffer override settings include:

- **xcode-jitter-buffer-low-min**—Default: 60 — Range: 60 to 120ms
- **xcode-jitter-buffer-low-max**—Default: 120ms
- **xcode-jitter-buffer-high-min**—Default: 60ms
- **xcode-jitter-buffer-high-max**—Default: 180ms
- **xcode-jitter-buffer-adaptive**—Default: enable. Enables or/disables the adaptive feature. If disabled, there is no adaptation of the jitter buffer, regardless of the values of the low/high/min/max settings.

These four settings set the min/max for both the low-level and high-level operation. An example of the syntax for these options is presented below.

```
ORACLE(media-manager)# options +xcode-jitter-buffer-low-min 75
ORACLE(media-manager)# options +xcode-jitter-buffer-high-min 100
```

If you type the option without the plus sign, you overwrite any previously configured options. To append new options to an element's options list, prepend the new option with a plus sign as shown in the example.

> **Note:**
>
> These options replace the older **xcode-jitter-buffer-min** and **xcode-jitter-buffer-max** options. Depending on your system's version, the SBC supports either the new or older options.

> **Note:**
>
> Oracle recommends you remove any **xcode-jitter-buffer-min** and **xcode-jitter-buffer-max** options settings prior to upgrading from a version that does not support this feature to a version that does.

Reporting on Adaptive Jitter

The vSBC includes reporting fields for the counts and transitions associated with this feature at the bottom of the **show xcode session-byid** command. They are available for each transcoding session.

```
ORACLE show xcode session-byid
Channel 1:
HSP device = 0
Egress Interface = 0
FLOWID = 11
FLOWID-RTCP = 11
Src IP:Port = 192.168.0.193:10004
Dst IP:Port = 192.168.0.21:6000
Src RTCP IP:Port = 192.168.0.193:10005
Dst RTCP IP:Port = 192.168.0.21:6001
…

Adaptive Stats:
JitterState 2
JitterCurrent (ms) 8.5
JitterMaximum (ms) 10.4
JitterAverage (ms) 6.74
TotalHighMaxPkts 51
TotalHighMinPkts 425
TotalLowPkts 3
TotalHighLowDetects 0
TotalLowHighMaxDetects 0
TotalLowHighMinDetects 1
TotalHighLowResets 0
TotalLowHighResets 0
```

The "Adaptive Stats" are the applicable statistics, showing the current, maximum, and average calculated jitter for the specified transcoded voice session. The "Detects" counts track how many threshold transitions have occurred. Jitter buffer operation does not change until a jitter buffer reset occurs, which the system tracks as "Resets".

PCIe Transcoding Accelerator Cards

PCIe transcoding accelerator cards enable high-density media processing using the same DSP hardware as physical Acme Packet platform hardware-based transcoding. A PCIe transcoding accelerator card in conjunction with a vSBC provides functional parity with engineered SBC platforms.

- PCIe transcoding accelerator cards work seamlessly with vSBCs, once initialized and recognized by the hypervisor. In this way there are no unique configuration or maintenance tasks to be aware of.
- Provisioning any transcoding cores for software-based transcoding is incompatible with a PCIe card.
- If the PCIe transcoding accelerator card fails after the SBC starts, an alarm is raised and the system is rebooted.
- When vSBCs are configured in a redundant pair, both systems must have the same population of PCIe cards and installed DSPs.

Refer to the Release Notes document for all system prerequisites and limitation. Check My Oracle Support for the latest application notes on how to initialize PCIe cards on supported hypervisors.

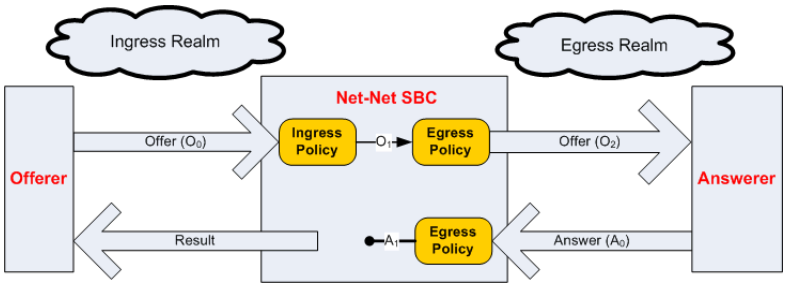
Transcoding Processing Overview

Transcoding processing is viewed in terms of the ingress and egress realms. The ingress realm is where the SDP offer is received by the Oracle Communications Session Border Controller. The egress realm is where the SDP offer is sent, and where the SDP answer is expected to be received from (i.e., the answerer's realm). A call is defined as transcodable if an egress or ingress policy exists for the session and if the session is not subject to media release, as specified in the realm configuration.

To understand the details of transcoding processing, refer to the following diagram. An SDP offer, O0, is received by the Oracle Communications Session Border Controller in the ingress realm. The ingress codec policy is applied here and the SDP offer is transformed into O1. O1 is then passed to and processed by the egress codec policy. This SDP message is then forwarded to the answerer as O2. The answerer replies with A0 to the Oracle Communications Session Border Controller, which is subjected to the egress codec policy again and transformed into A1.

When policy dictates not to transcode the call, the Result SDP sent back to the offerer is based on the common codecs shared between A1 and O1. The Oracle Communications Session Border Controller first constructs the list of codecs that are present in both in O1 and A1. Then, the Oracle Communications Session Border Controller maintains the codec order from A1 for the Result as it is sent to the offerer.

When policy dictates to transcode the call, the top transcodable codec in O1 is used in the ingress realm and the top non-signaling codec in A1 is used in the egress realm.



Unoffered Codec Reordering

According to RFC 3264, the answerer can add codecs that were not offered to the Answer. The answerer may add new codecs as a means of advertising capabilities. RFC 3264 stipulates that these unoffered codecs must not be used. The RFC does not dictate where in the m= line these codecs can appear and it is valid that they may appear as the most preferred codecs.

In order to simplify the answer processing, the Oracle Communications Session Border Controller moves all unoffered codecs in A0 to the back of the SDP answer before any other answer processing is applied.

Non-transcoded Call

The decision to transcode is based on the top non-signaling codec in A1. If the top A1 codec is present in O1, the call proceeds, non-transcoded. This is the rule for non-signaling codecs (i.e., not RFC 2833 nor FAX).

Transcoded Call

The following two conditions must then be met to transcode the call's non-signaling media:

- The top A1 codec is not present in the O1 m= line
- The top A1 codec was added by the egress policy

If these rule are met, the Oracle Communications Session Border Controller will transcode between the top A1 codec and the top transcodable, non-signaling O1 codec.

> **Note:**
>
> During an iLBC call establishment, when force-time is enabled and the offered and answered ptime are the same but with different iLBC mode, Oracle Communications Session Border Controller will disable transcoding and force the call into a transparent call.

Post Processing

If any errors are encountered during the Ingress and Egress policy application, or other violations of RFC3264 occur, the call is rejected. If O2 does not contain any enabled m= lines at the conclusion of the initial call setup, the call is rejected. If O2 does not contain any enabled m= lines at the conclusion of a reINVITE, the reINVITE is rejected and the call reverts back to its previous state.

Voice Transcoding

The following examples use the ingress and egress codec policies listed at the top of each scenario. The examples use changing SDP offers and answers, which contribute to unique results, per example. The effects of the SDP offers and answers are explained in each example.

Voice Scenario 1

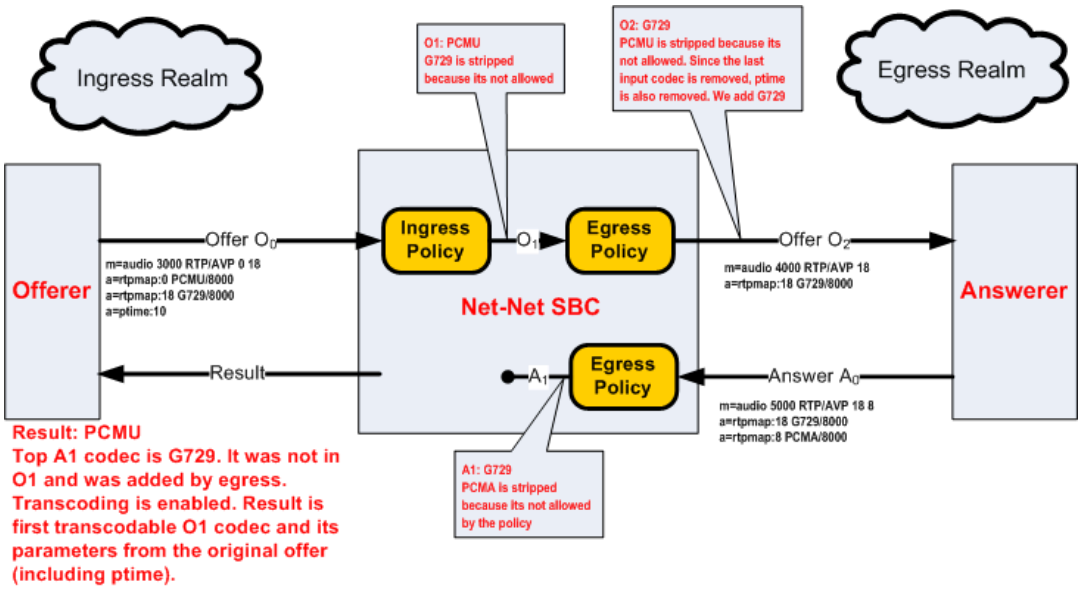
The following ingress and egress policies are used for scenario 1.

Ingress Policy	Ingress Policy	Egress Policy	Egress Policy
allow-codecs	PCMU GSM	allow-codecs	G729 GSM G722
add-codecs-on-egress	PCMU	add-codecs-on-egress	G729
order-codecs	N/A	order-codecs	N/A
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A

> **Note:**
>
> The codec in the ingress policy's add-codecs-on-egress parameter has no effect in the following examples. Its presence would have an effect if a reINVITE was initiated from egress realm, effectively reversing the roles of the codec policies.

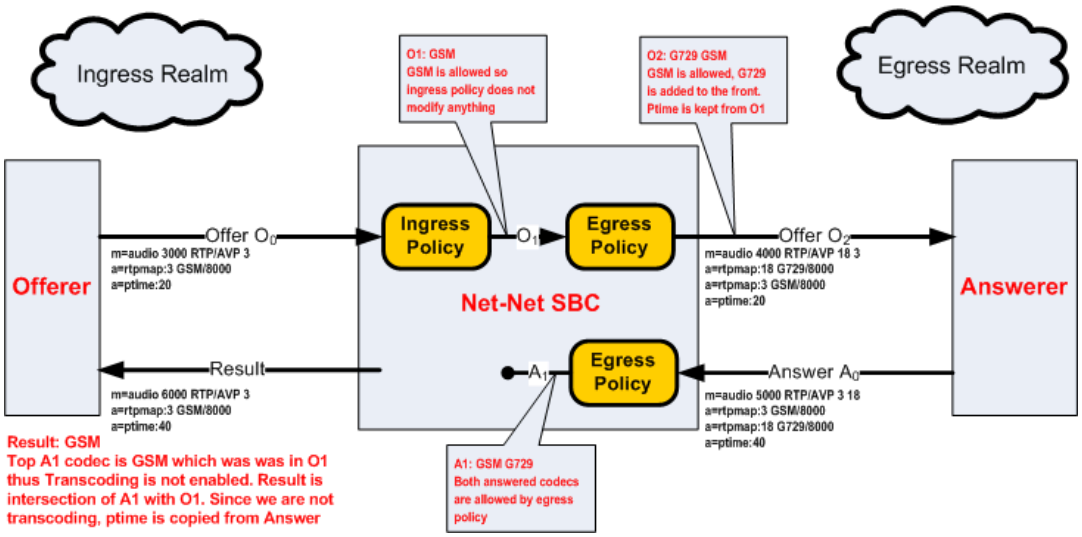
1. In the following diagram, PCMU and G729 are offered. Ingress policy removes G729 and allows PCMU. The egress policy adds G729 and strips PCMU from offered SDP and forwards it on to the answerer (ptime is also removed because the last codec was removed).

   The SDP answer agreed to use G729 and adds PCMA. The egress policy then strips PCMA from the SDP answer. At this point, the top codec in A1, G729 is checked against O1. Since G729 is not present in O1, it is transcoded to PCMU.

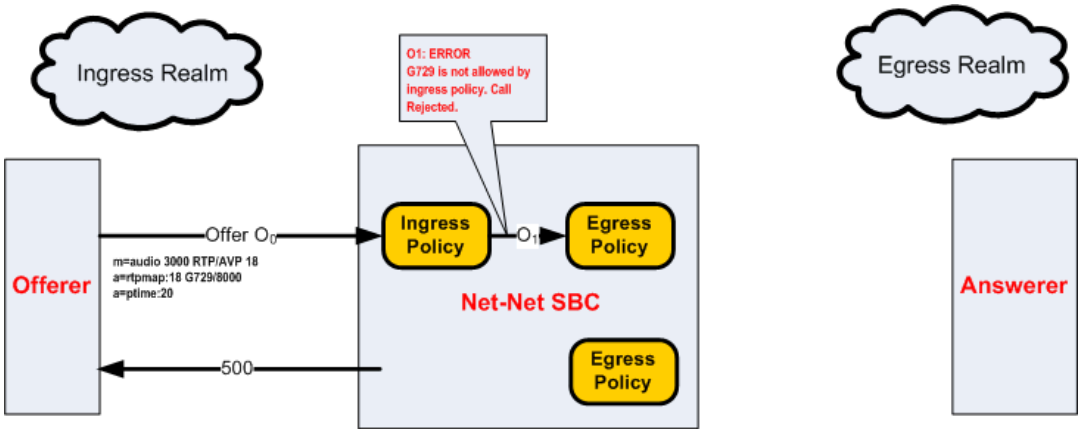


2. In the following diagram, GSM is in the original SDP offer. It is then passed through to O1. Egress policy adds G729 and retains ptime from GSM and sends this to the answerer as O2.

   The SDP answer agrees to use G729 and GSM, but prioritizes GSM. The egress policy allows both codecs through, unchanged. Because A1 and O1 both have GSM, it is used for the non-transcoded call. Ptime is copied from A0 to the result.

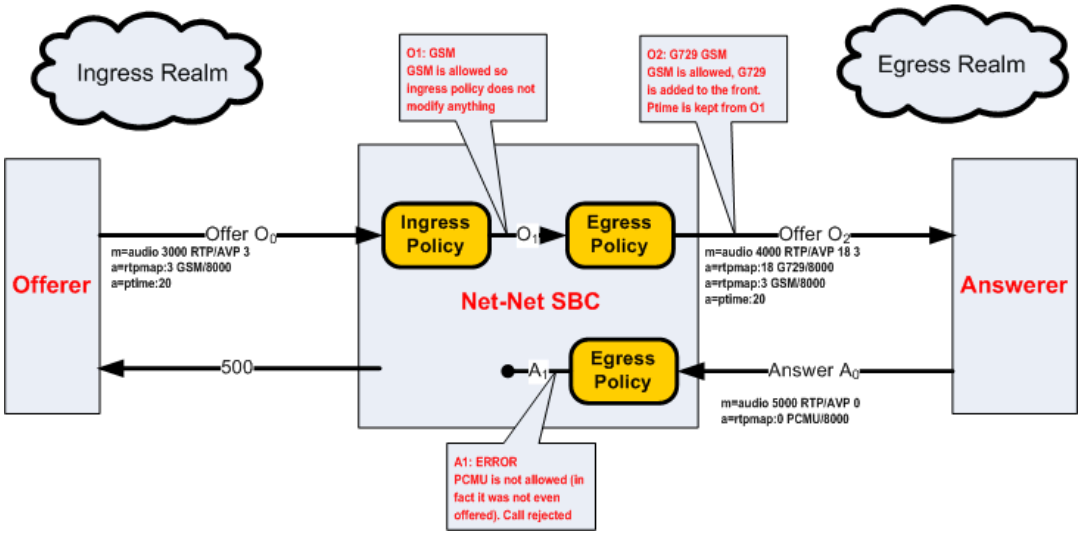


3. In the following diagram, G729 in the original SDP offer. Because once G729 is removed, no non-signaling are left in O1, thus the call is rejected.



4. In the following diagram, GSM is in the original SDP offer. It is then passed through to O1. Egress policy adds G729 and retains ptime from O1 and sends this to the answerer as O2.

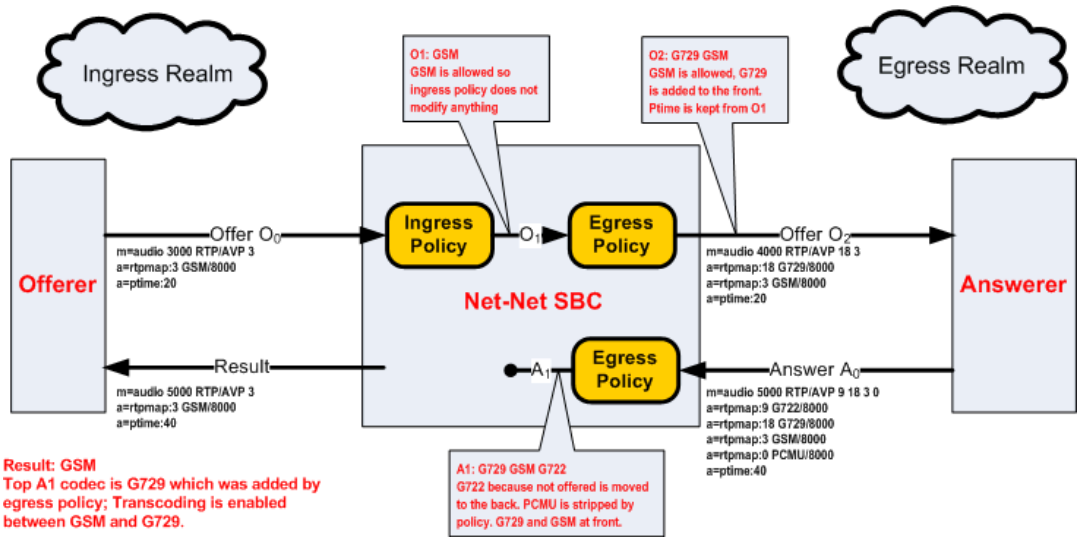
   The SDP answer states that the answerer wants to use PCMU. This is a violation of the RFC3264. Therefore the call is rejected.



In this example, when the negotiation fails, the Oracle Communications Session Border Controller sends a 500 message to the offerer and a BYE message to the answerer.

5. In the following diagram, GSM is in the original SDP offer. It is then passed through to O1. Egress policy adds G729 and retains ptime from O1 and sends this to the answerer as O2.

   The SDP answer replies with G722 G729 GSM and PCMU. PCMU is stripped by policy, G722 is moved to the back of the answer because it was not offered. The top A1 codec was not in O1, and was added by egress policy, therefore the call is transcoded between GSM and G729.

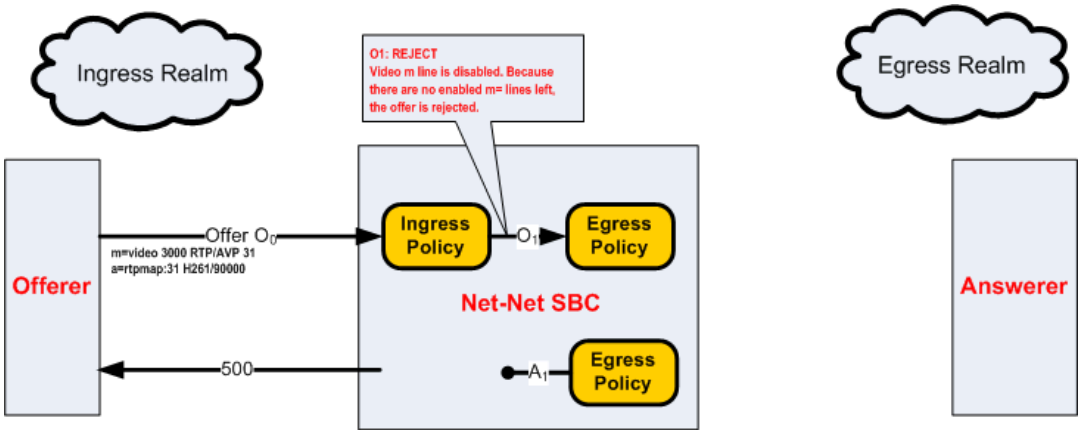


Voice Scenario 2

The following ingress and egress policies are used for scenario 2.

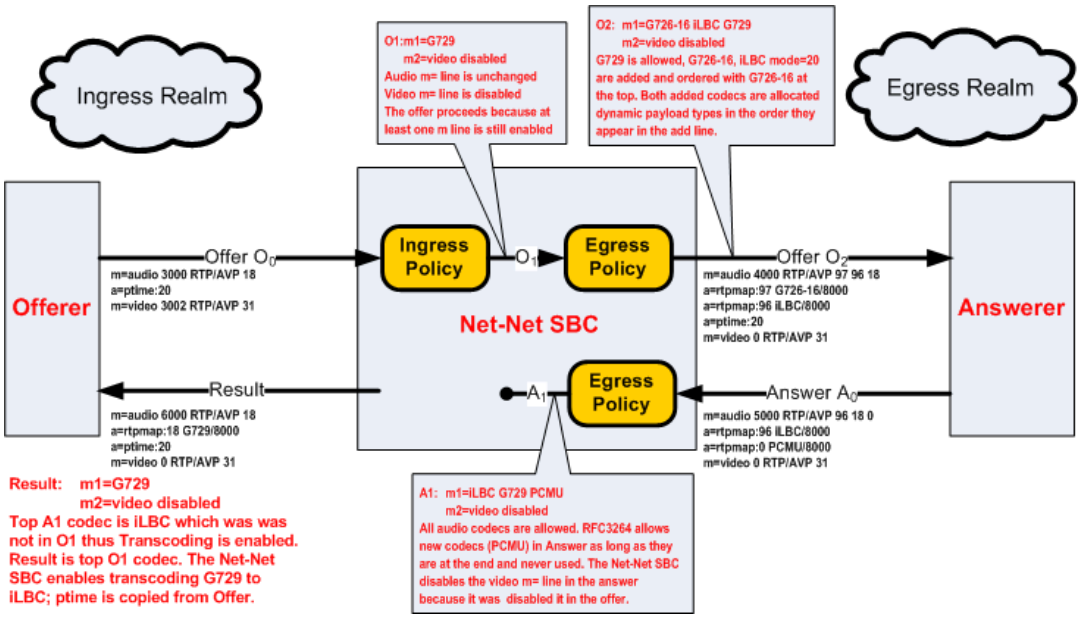
Ingress Policy	Ingress Policy	Egress Policy	Egress Policy
allow-codecs	Video:no PCMU:force * PCMA:force	allow-codecs	* PCMA:no
add-codecs-on-egress	N/A	add-codecs-on-egress	iLBC G726-16
order-codecs	N/A	order-codecs	G726-16 * PCMU
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A

1. In the following diagram, a video m= line is offered. The ingress policy disables the video m= lines. With no enabled m= lines left, the call is rejected.



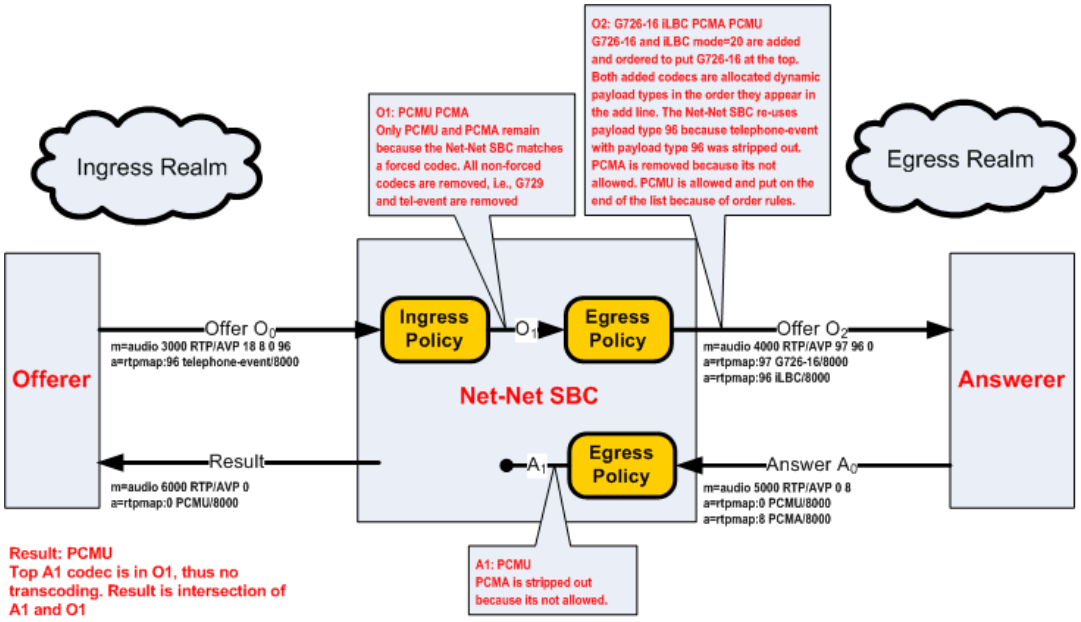
2. In the following diagram, G729 and video are offered to the Oracle Communications Session Border Controller. Ingress policy allows G729 and disables the video m= line. The egress policy adds iLBC and G726-16, and then orders the codecs according to the order-codecs parameter. The ptime is maintained between O0 and O2. Both added codecs are allocated dynamic payload types in the order they appear in their m= line. A disabled Video m= line is passed on to the answerer.

   The SDP answer agreed to use iLBC, G729, and adds PCMU, and reorders them as stated. The disabled video m= line is maintained. At this point, the top codec in A1, iLBC is used and transcoded with the top codec in O1, G729.



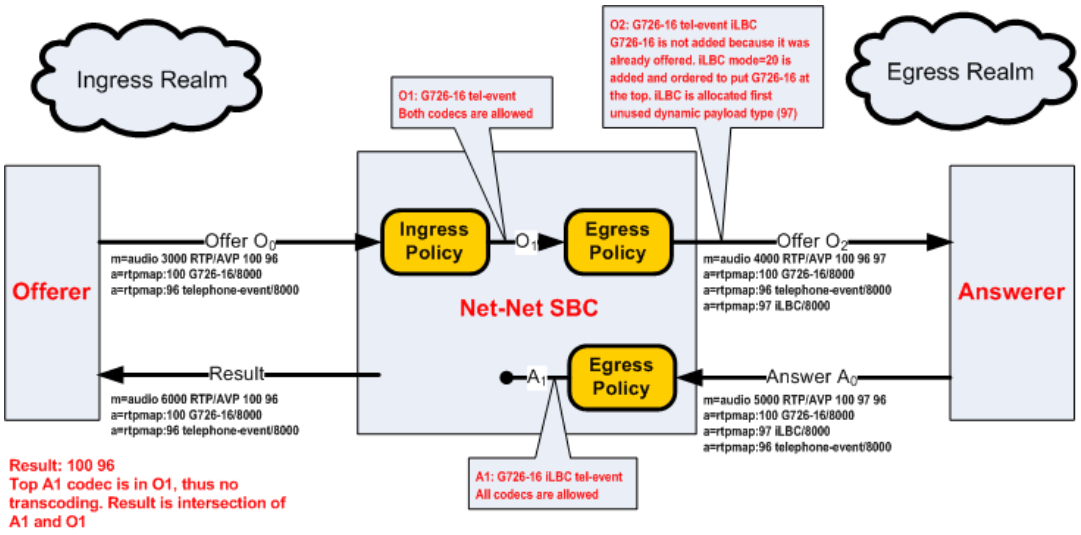
3. In the following diagram, G729 and video are offered to the Oracle Communications Session Border Controller. Ingress policy allows G729 and disables the video m= line. The egress policy adds iLBC and G726-16, and then orders the codecs according to the order-codecs parameter. The ptime is maintained between O0 and O2. Both added codecs are allocated dynamic payload types in the order they appear in their m= line.

   The SDP answer only wants to use PCMU and PCMA. The egress policy removes PCMA and passes only PCMU to the offerer. Because PCMU was in O1 and is now the only codec in A1, it is used, and no transcoding is used between the endpoints.



4. In the following diagram, G726-16 and telephone-event are offered to the Oracle Communications Session Border Controller. Ingress policy allows both codecs. The egress policy adds iLBC, and then orders the codecs according to the order-codecs parameter.

   The SDP answer agreed to use all codecs, but reorders them with G726-16 in the top position. Because G726-16 is the top codec in A1, and it is also present in O1, it is used for this call without any transcoding.



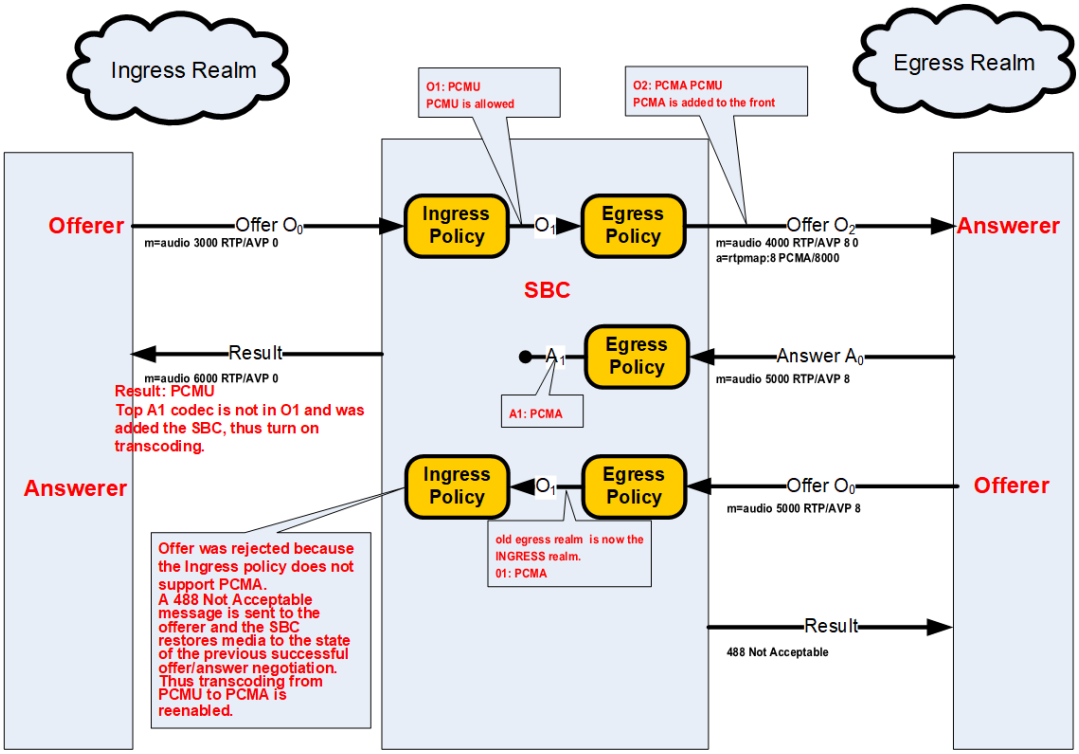
Voice Scenario 3

Voice scenario 3 involves reINVITEs. The following ingress and egress policies are used for scenario 3.

Ingress Policy		Egress Policy	
allow-codecs	PCMU G729	allow-codecs	*

Ingress Policy		Egress Policy	
add-codecs-on-egress	N/A	add-codecs-on-egress	PCMA
order-codecs	N/A	order-codecs	N/A
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A

- In the following diagram, the answerer sends a reINVITE after a previous transcoding session was established. The original offerer and answerer swap roles. The new offerer is rejected by Oracle Communications Session Border Controller and the call reverts to the state negotiated in the original SDP negotiation.



RFC 2833 Transcoding

RFC 2833 defines an RTP payload that functions interchangeably with DTMF Digits, Telephony Tones and Telephony Signals. The Oracle Communications Session Border Controller can monitor audio stream for in-band DTMF tones and then can convert them to data-based telephone-events, as sent in RFC2833 packets. This section explains how the Oracle Communications Session Border Controller transcodes between these RTP-based telephone events and in-band DTMF tones carried by G711. DTMF tones can only be transported in non compressed codecs. The Oracle Communications Session Border Controller supports two DTMFable non-compressed codecs: PCMU (G711µ) and PCMA (G711A).

> **Note:**
>
> The following line is added to SDP whenever telephone-event is added on egress: a=fmtp:101 0-15

The following two scenarios describe when telephone-event to DTMF transcoding takes place:

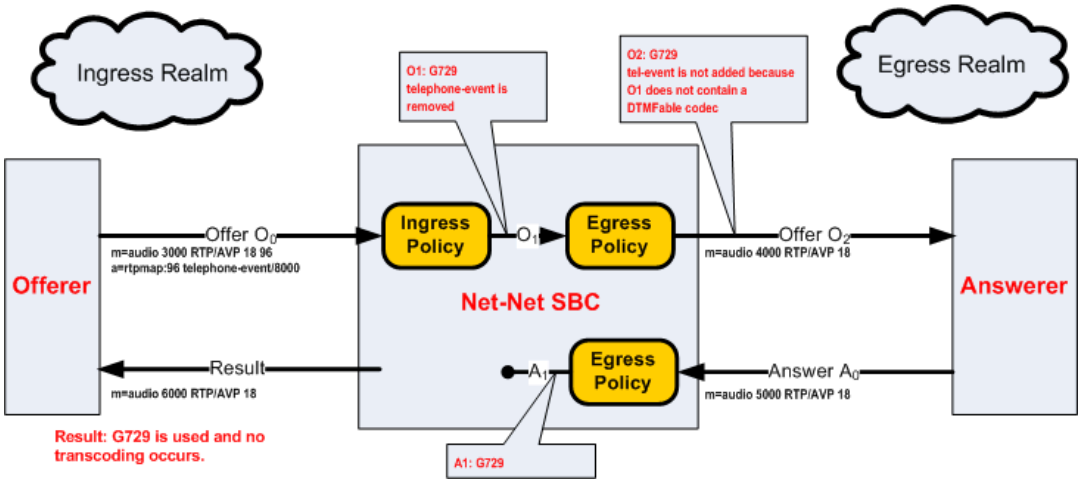
RFC 2833 Scenario 1

The following ingress and egress policies are used for scenario 1.

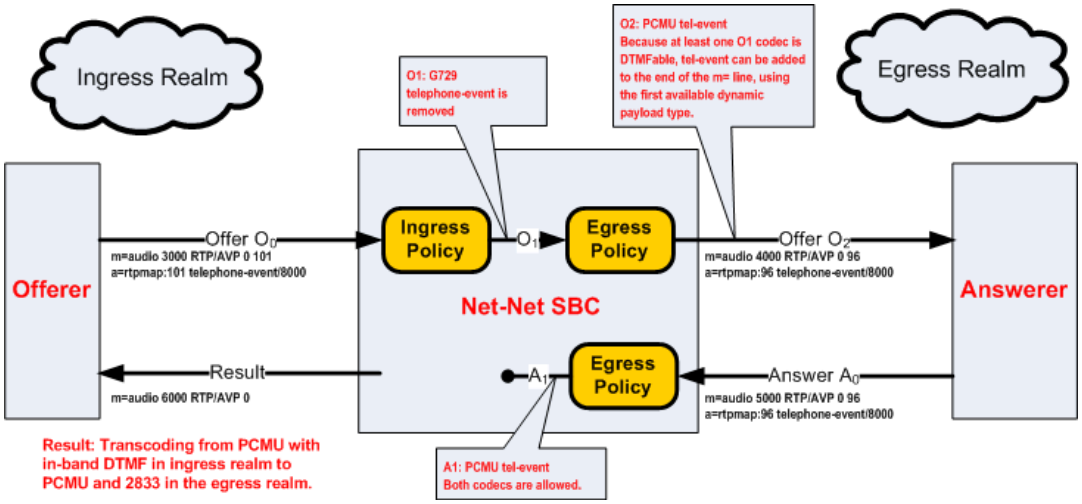
Ingress Policy		Egress Policy	
allow-codecs	* telephone-event:no	allow-codecs	* PCMA:no
add-codecs-on-egress	N/A	add-codecs-on-egress	telephone-event
order-codecs	N/A	order-codecs	N/A
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A
dtmf-in-audio	preferred	dtmf-in-audio	preferred

1. In the following diagram, telephone event was offered by the offerer but was stripped by ingress policy. telephone-event was not added by the egress policy because the remaining audio codec in O1 was not DTMFable. G729 was the only codec forwarded on to the answerer.

   The SDP answer agreed to use the remaining audio codec, G729. A0 is unaltered by egress policy, and forwarded as the Result to the offerer. Therefore, G729 is used in both the ingress and egress realms, the call does not support RFC 2833, and the call is not transcoded.

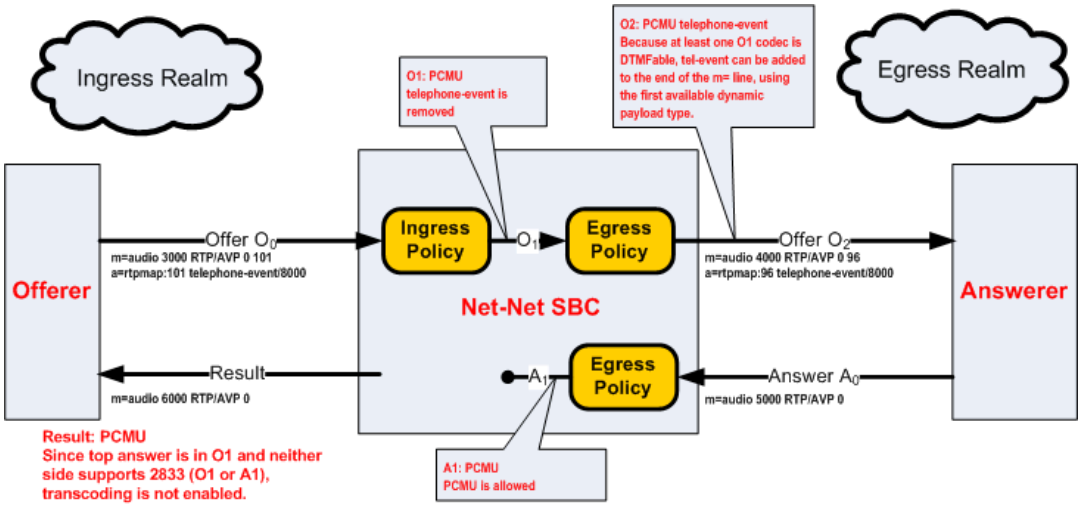


2. In the following diagram, telephone event was offered by the offerer but was stripped by ingress policy. telephone-event was added by the egress policy because the remaining audio codec in O1 was DTMFable. PCMU and telephone-event are then forwarded on to the answerer. Note that the telephone-event payload type is added with the lowest available dynamic type number.



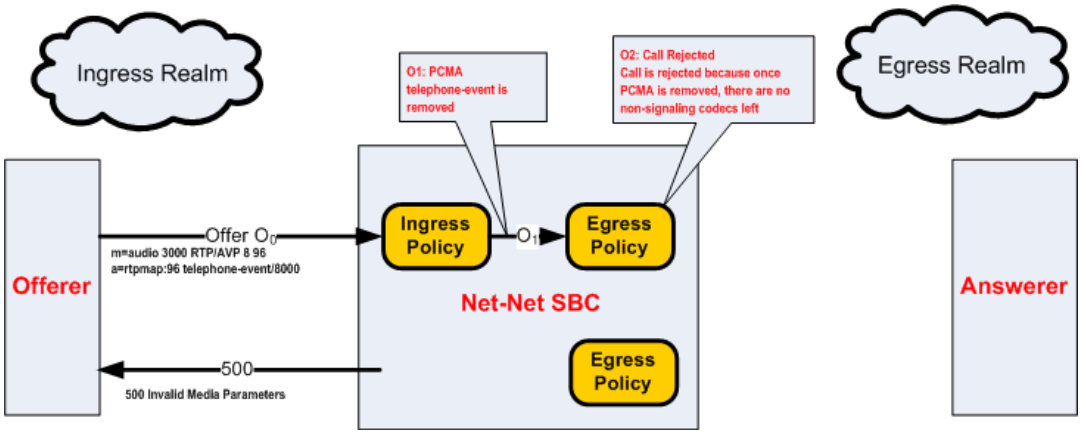
This case illustrates when the answerer supports audio and RFC 2833, but the offerer supports audio with inband DTMF. The Oracle Communications Session Border Controller transcodes between RFC2833 in the egress realm to in-band DTMF on the ingress realm.

3. In the following diagram, telephone event was offered by the offerer but was stripped by ingress policy. telephone-event is added by the egress policy because the remaining audio codec in O1 was DTMFable. PCMU and telephone-event are then forwarded on to the answerer. Note that the telephone-event payload type is added with the lowest available dynamic type number.



The SDP answer only agreed to use PCMU. When A0 reaches the egress policy, it is passed along through the Oracle Communications Session Border Controller to the offerer. Because telephone-event was not answered by the answerer and not supported in O1, it can't be used. Transcoding is therefore not used for this call.

4. In the following diagram, telephone event was offered by the offerer and was stripped by ingress policy. Since PCMA was also stripped by the egress policy, leaving no non-signaling codecs, the call is rejected. A 500 message is sent back to the offerer.

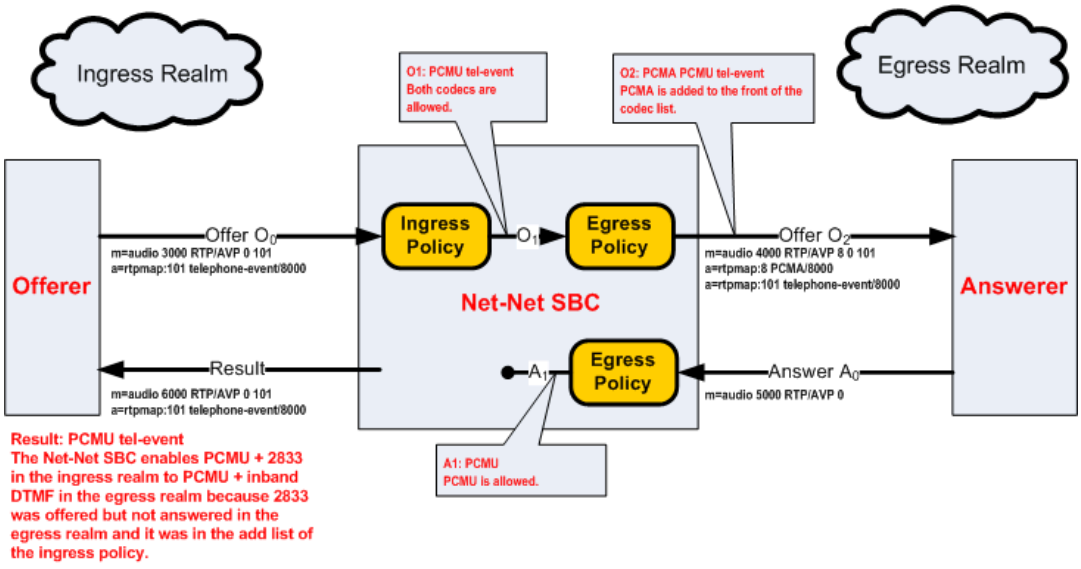


RFC 2833 Scenario 2

The following ingress and egress policies are used for RFC2833 scenario 2.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*
add-codecs-on-egress	telephone-event	add-codecs-on-egress	PCMU
order-codecs	N/A	order-codecs	N/A
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A
dtmf-in-audio	preferred	dtmf-in-audio	preferred

1. In the following diagram, telephone event and PCMU are offered by the offerer. They are both passed to O1, and PCMA is added as it is sent to the answerer. The SDP answer, A0 disables all codecs but PCMU.

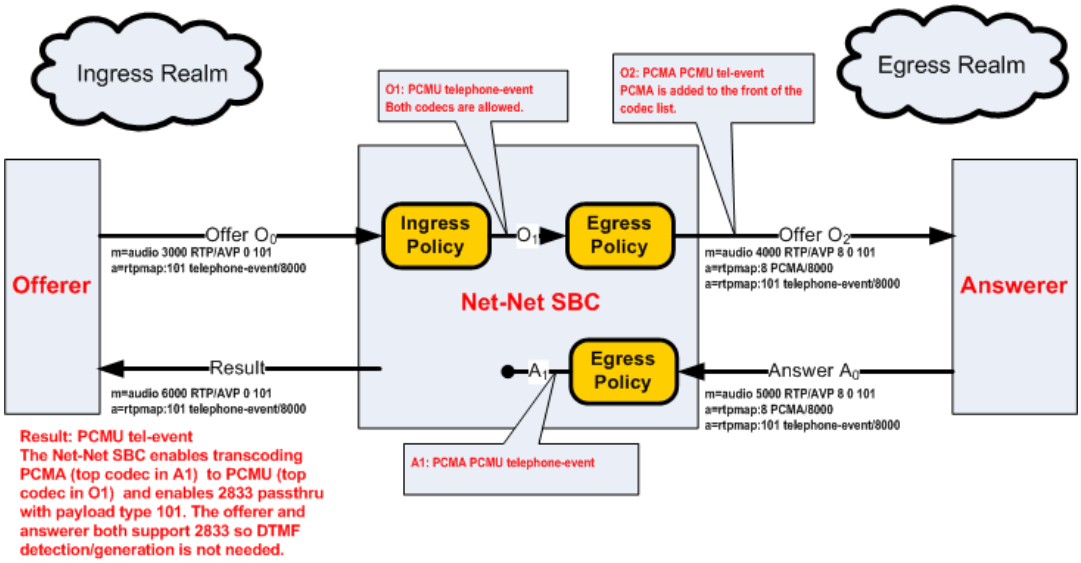


The Oracle Communications Session Border Controller adds telephone-event to the result because it is listed on the ingress policy's add-codecs-on-egress parameter and present in the offerer's SDP.

> **Note:**
>
> This is the only time the add list of an ingress policy is utilized as a check.

The result SDP includes PCMU and telephone-event in the ingress realm, which is transcoded to PCMU with in-band DTMF in the egress realm.

2. In the following diagram, telephone event and PCMU are offered by the offerer. They are both passed to O1, and PCMA is added as it is sent to the answerer. The SDP answer supports all three codecs offered, with PCMA added on top.



The answerer responds with PCMA as the preferred codec in A0. The Oracle Communications Session Border Controller compares A1 to O1 to make the transcoding decision. PCMA is the top codec in A1 and is transcoded to PCMU, the top codec in O1. Also, because telephone-event is supported by both sides of the call, it is passed through without any transcoding necessary.

This case illustrates when both endpoints are capable of sending and receiving telephone-event. Regardless of whether the audio portion of the call is transcoded, the telephone-event messages are passed through the system untouched, thus not requiring transcoding resources. This is known as telephone-event pass-through.

FAX Transcoding

FAXes are transmitted in a call as either T.30 or T.38 media. T.30 FAX is binary in-band media carried over G.711. The Oracle Communications Session Border Controller (SBC) can transcode between T.38 and a faxable codec. The supported faxable codecs are PCMU and PCMA.

T.30 can be transported only in non-compressed codecs. The two non-compressed codecs supported by the SBC are PCMU (G711µ) and PCMA (G711A). If a transcoding realm does not support an uncompressed codec, T.30 cannot be supported in that realm. Alternatively, G711FB may be allowed specifically for FAX only.

The SBC uses an internal codec called G711FB (G711 - Fall Back) that is an umbrella codec of all FAXable codecs. G711FB will default to PCMU for the purpose of offering a faxable codec. You can re-map G711FB to PCMA by configuring the media-profile for it appropriately. G711FB's only use is for FAX transcoding.

FAX transcoding is triggered when you configure the add on egress parameter with either T.38 or G711FB. In a FAX scenario, when the codec policy adds either T.38 or G711FB, a new m= line is added to the SDP. When adding T.38, the new m= line specifies the T.38 codec. When adding G711FB, the new m= line specifies PCMU (or alternatively PCMA).

When added, m= lines cannot be deleted in the context of a call. The SBC maintains all m= lines between itself and an endpoint throughout the course of call. All m= lines not in use can be disabled by setting their receive port to 0, but they cannot be removed from the SDP.

Defining G711FB

G711 Fall Back (G711FB) is an internal codec that encompasses PCMU and PCMA for carrying fax information in FAXable codecs. You must configure the G711FB codec for either PCMU or PCMA for when the Oracle Communications Session Border Controller inserts a FAXable codec in SDP. G711FB is used only for FAX transcoding scenarios.

To define G711 FB, create a media profile configuration element named g711fb and set the payload-type to 0 or 8.

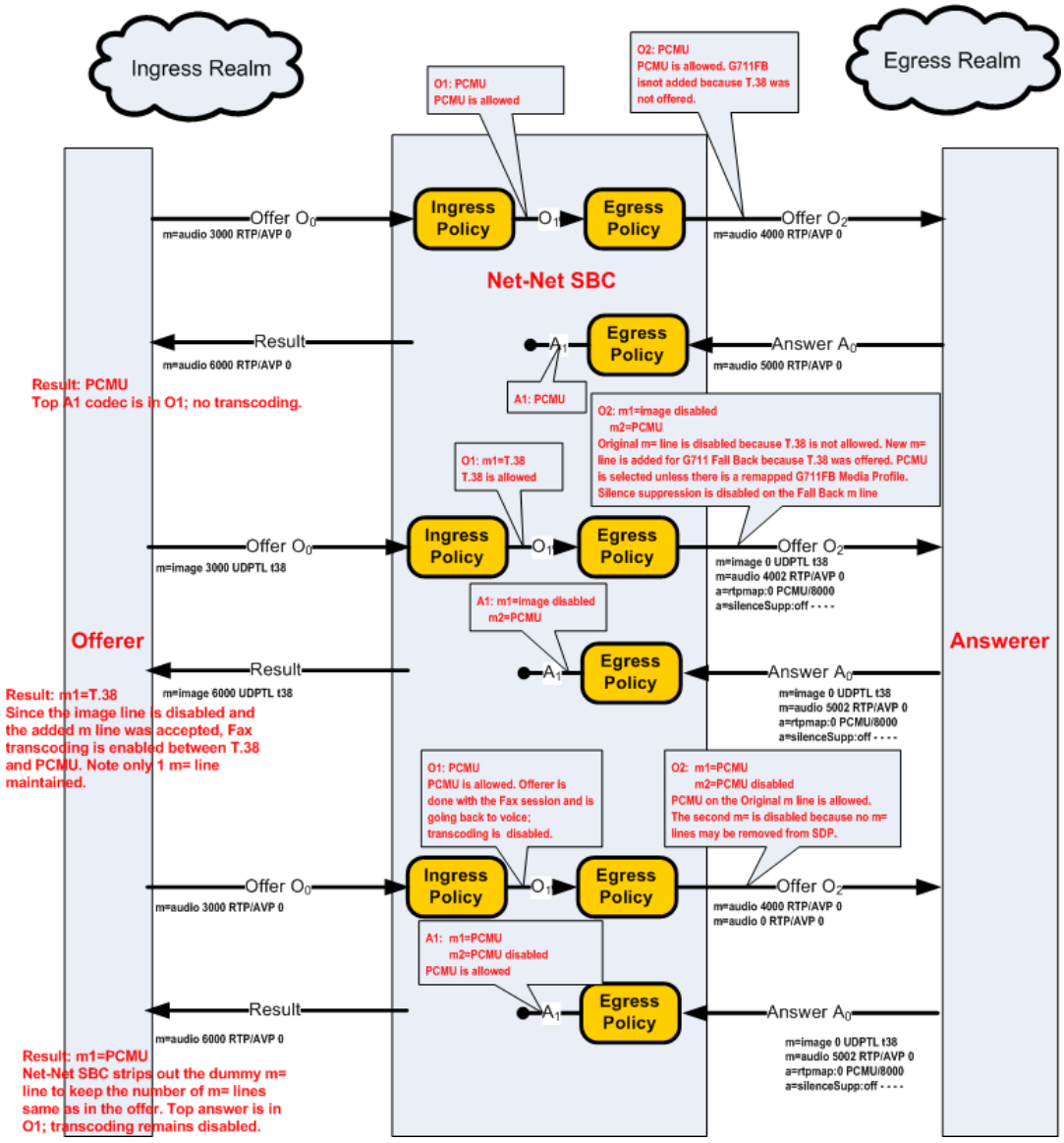
Codec (supported bit rates)	RTP Payload Type	Default Ptime (ms)	Supported Ptime (ms)
T.38	N/A	30	10, 20, 30
G711FB (64 kbps)	0, 8	30	10, 20, 30

FAX Scenario 1

The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	T.38:no
add-codecs-on-egress	N/A	add-codecs-on-egress	G711FB
order-codecs	N/A	order-codecs	N/A
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A

- In the following diagram, there are three offer-answer exchanges. Initially a PCMU-to-PCMU session is negotiated. Next, a T.38 to PCMU session is negotiated. Finally, the session reverts to non-transcoded PCMU to PCMU state.



FAX Scenario 2

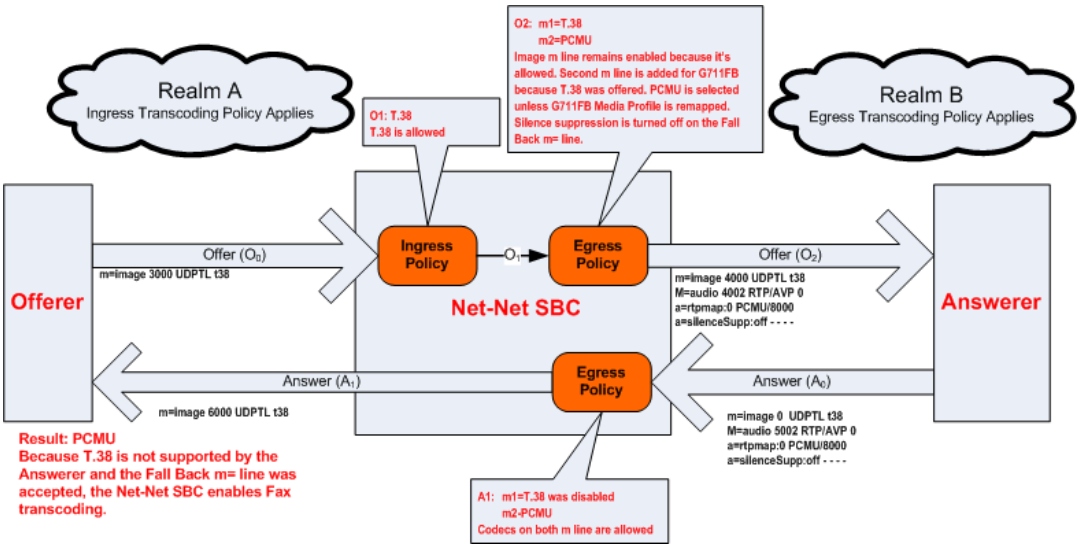
The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*
add-codecs-on-egress	N/A	add-codecs-on-egress	G711FB
order-codecs	N/A	order-codecs	N/A
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A

1. In the following diagram, T.38 is offered to the Oracle Communications Session Border Controller. A second m= line was added to O1 that included a G711FB codec (PCMU).

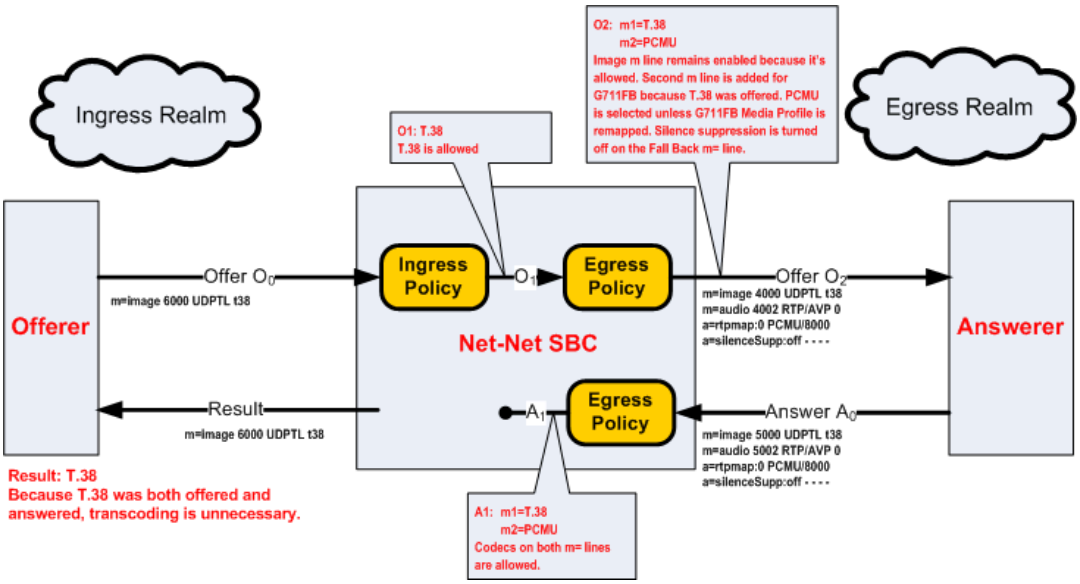
   The SDP answer agreed to PCMU, but disabled T.38. When the Oracle Communications Session Border Controller forwarded the SDP in A1 to the answerer, it stripped the second

m= line. Because A1 rejects T.38 m= line, but accepts the PCMU m= line, FAX transcoding is enabled.



2. In the following diagram, T.38 is offered to the Oracle Communications Session Border Controller. A second m= line was added to O1 that included a G711FB codec (PCMU).

   The SDP answer agreed to PCMU and T.38. Because both O1 and A1 support T.38, the call proceeds without transcoding.



FAX Scenario 3

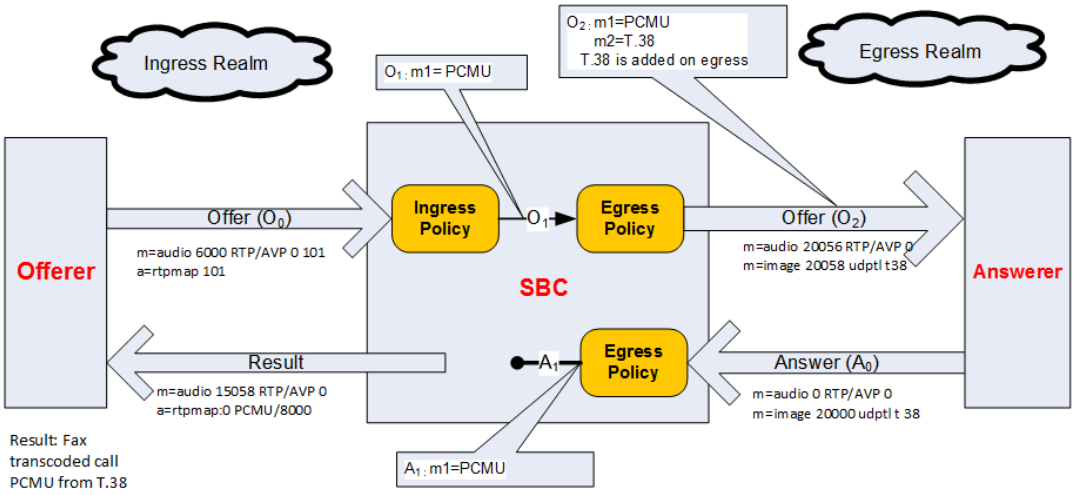
The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*

Ingress Policy		Egress Policy	
add-codecs-on-egress	N/A	add-codecs-on-egress	PCMU, G729 ,T.38
order-codecs	N/A	order-codecs	N/A
force-ptime	disabled	force-ptime	disabled
packetization-time	N/A	packetization-time	N/A

1. In the following diagram, PCMU and telephone-event codecs are received by SBC .The egress codec policy has PCMU, G729 and T.38 **add-codecs-on-egress**.

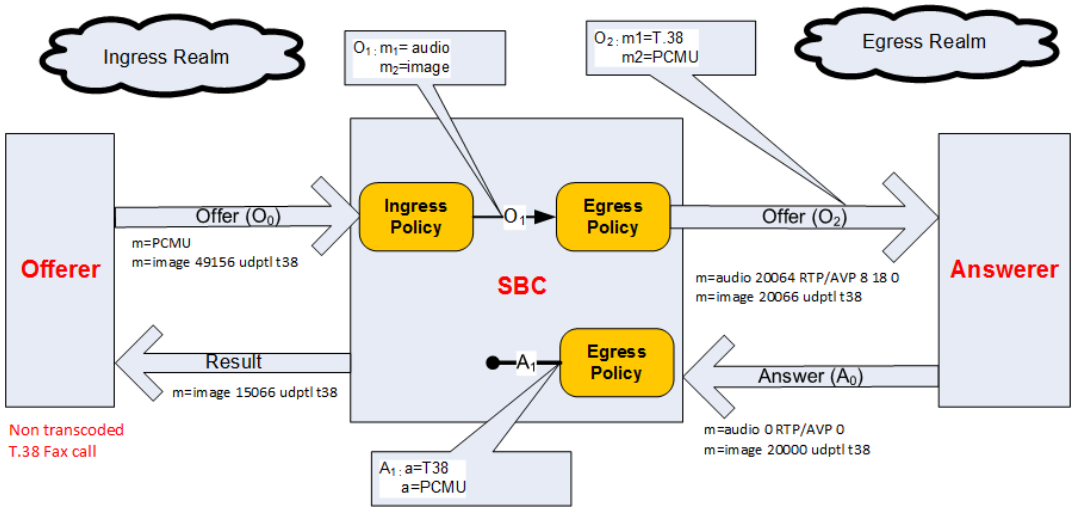
   Since there is a faxable codec in the SDP offer and T.38 in **add-on-egress**, the non-faxable codec G729 is stripped and T.38 is added to egress offer.
   In the A0 answer the audio m-line is zero indicating disabled and the port for the image m-line is non-zero (20000) so the called party has selected T.38. Hence PCMU in realm A is transcoded to T.38 in realm B.



2. In the following diagram, PCMU and T38 are received by the Oracle Communications Session Border Controller. The egress codec policy has PCMU, G729, and T.38 **add-codecs-on-egress**.

   T.38 is present in the O0 offer, and therefore will not be explicitly added by the egress codec policy to the O2 Offer. The logic to remove non-faxable codecs is only invoked when T.38 is added by the **add-codecs-on-egress** parameter. In this example, T.38 is not added since is already present in SDP. With PCMU and T.38 received, G729 as a non-faxable codec is not removed. Therefore, PCMU, T.38, and G729 are all present in the O2 SDP offer sent to the answerer.

   In the A0 answer, the audio m-line port is 0 indicating that audio is disabled, and the port for the image m-line is nonzero (20000), thus the called party has selected T.38. In the A1 answer the SBC send the audio m-line port as 0 indicating that audio is disabled, and the port for the image m-line is nonzero (20000). This set-up results in a non-transcoded T.38 - SBC -T.38 fax call.



Transrating

The Oracle Communications Session Border Controller can transrate media as it exits the Oracle Communications Session Border Controller into the network. Transrating is also known as forced packetization time (ptime), and is used to enforce a configured ptime within a realm. Transrating is often desirable when devices in a realm can only accept media with a specific ptime, or to optimize bandwidth.

If this feature is configured, the media portion of a call is transrated regardless of which codecs are ultimately chosen for each realm as long as they are transcodable. This allows realms that have devices that can only use a single packetization interval to interwork with devices that may or may not have the same packetization capabilities.

You must enable force-ptime in the egress codec policy and then specify the packetization time to force. When force ptime is enabled, it implicitly masks all codecs not of the specified packetization time that are listed in that codec policy's allow codecs and add codecs on egress parameters. For example, if force ptime is enabled with a packetization time of 20 ms, then no G723 codecs (which are only available at 30, 60, and 90 ms) may be active via codec policy in that realm.

Transrating occurs when forced-ptime is enabled and the offered and answered ptimes do not match and the top non-Signaling codec of A1 and top non Signaling codec of O1 are Transcodable.

> **Note:**
>
> Answered ptime A1 does not have to be equal to the ptime inserted into the outgoing offer O2, it just has to be different than the offer the Oracle Communications Session Border Controller received (O1).

Transrating Scenario 1

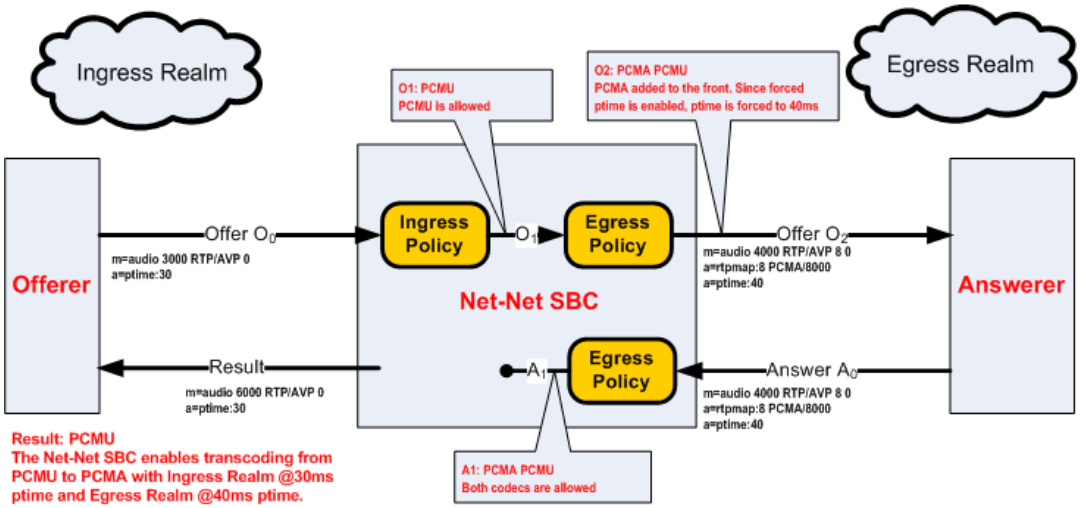
The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*

Ingress Policy		Egress Policy	
add-codecs-on-egress	N/A	add-codecs-on-egress	PCMA
order-codecs	G723 *	order-codecs	N/A
force-ptime	disabled	force-ptime	enabled
packetization-time	N/A	packetization-time	40

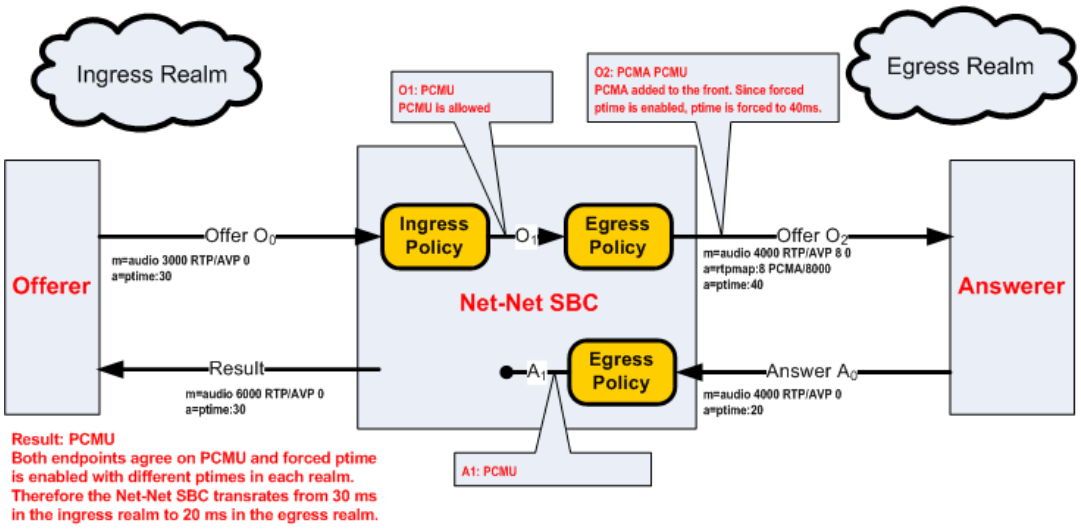
1. In the following diagram, PCMU is offered in the ingress realm with 30ms ptime, and the egress realm is forced to use 40ms ptime. PCMA is added as the top codec for the egress realm.

   The Oracle Communications Session Border Controller enables transcoding between the ingress realm (PCMU) and the egress realm (PCMA) and the ptimes as negotiated are also maintained.



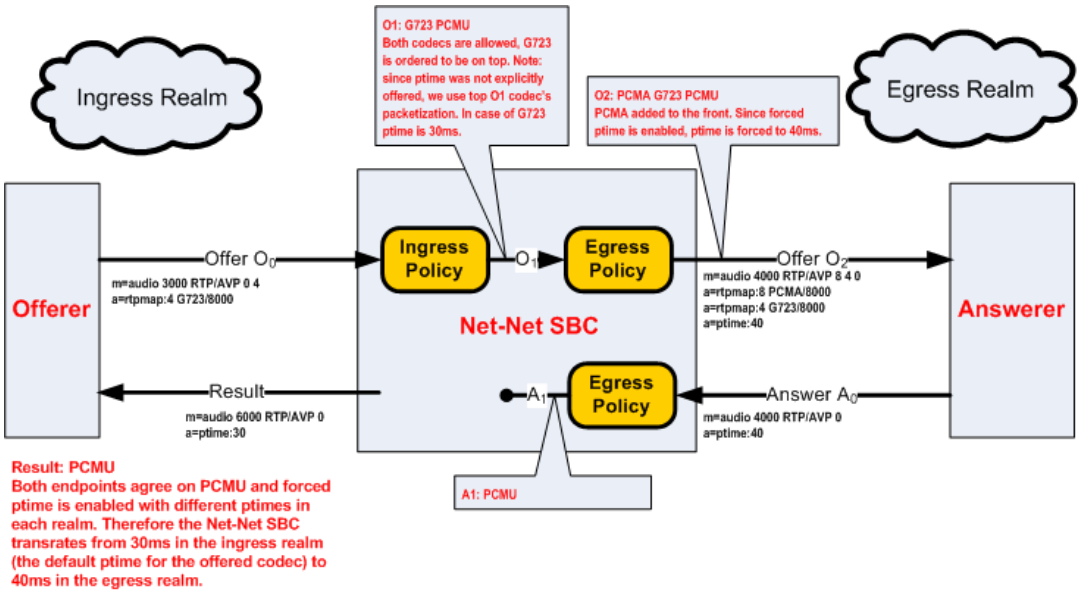
2. In the following diagram, PCMU is offered in the ingress realm with a ptime of 30ms, and forced to 40 ms in the egress realm by policy.

   The answerer chooses to use PCMU with a 20 ms ptime. Thus the call is not transcoded, but it is transrated from 30ms in the ingress realm to 20ms in the egress realm.



3. In the following diagram, PCMU and G723 are offered in Realm A. The top codec's ptime (30ms) is implied as the one for the ingress realm. The Oracle Communications Session Border Controller adds PCMA to the SDP offer with a 40ms ptime.

   The answerer chooses to use PCMU with a 40 ms ptime. Thus the call is transrated from 30ms in the ingress realm to 40ms in the egress realm.



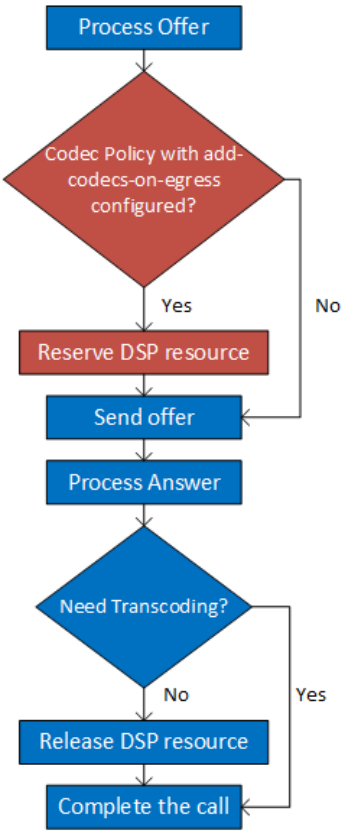
Reactive Transcoding

When setting up transcoded calls the Oracle Communications Session Border Controller (SBC) reserves a Digital Signaling Processor (DSP) resource for the incoming SDP offers that need transcoding. The default behavior is to pre-book the DSP resource and use it when the system's egress policy qualifies the SDP offer for transcoding.

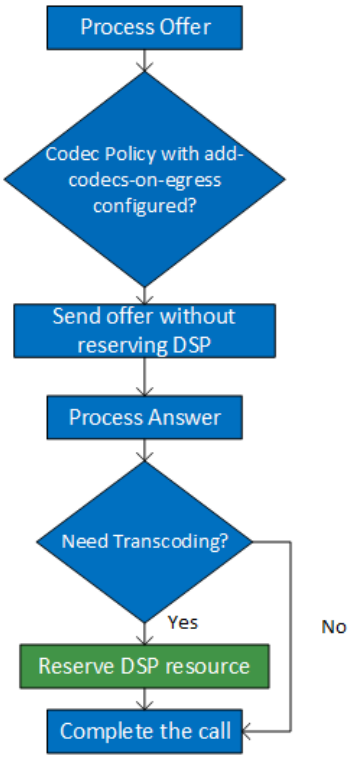
Alternatively, the SBC offers a **reactive-transcoding** option where the DSP resource is reserved after the SDP answer is received. Reactive transcoding is enabled by setting **media-manager-config**, **reactive-transcoding** to **enabled**. The advantage of this process is for every call that comes in the DSPs need not be pre-booked and instead can be dynamically reserved. The ideal situation for this behavior is when only a few calls need transcoding and the network traffic pattern is well known.

The SBC does not perform reactive transcoding on all transcoding call flows. When you configure support for the following features, the SBC reserves DSP resources:

- Pooled-transcoding
- dtmf-in-audio detection
- 140-Baudot transcode
- Fax call
- RTCP generation
- FAX tone detection

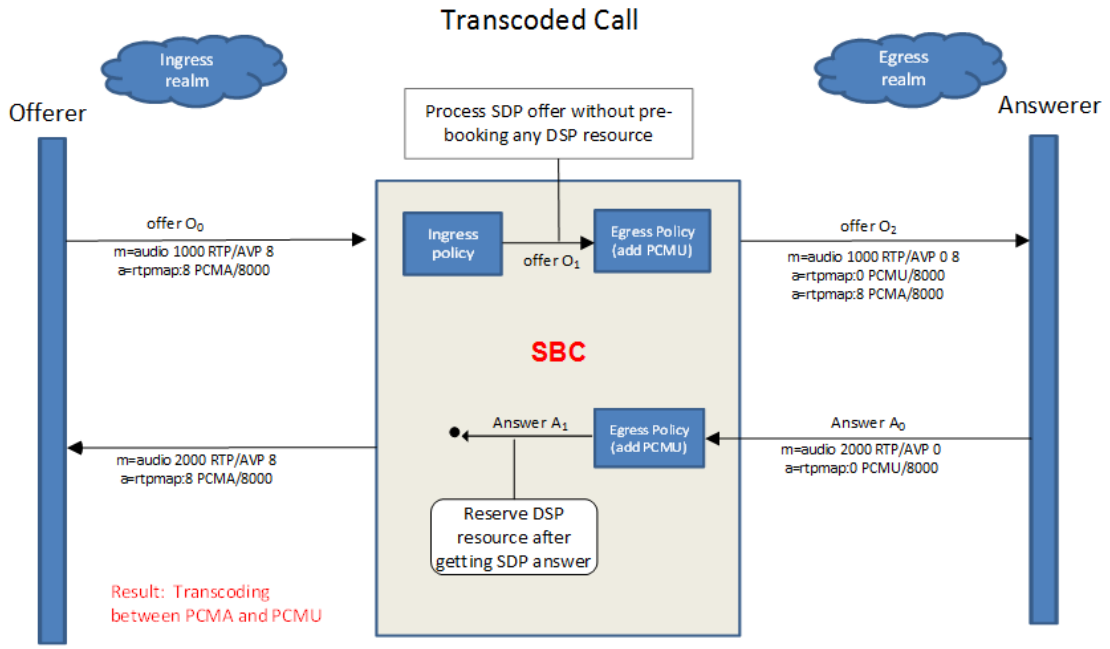


Current system behavior



System behavior when **reactive-transcoding** is enabled

The figure below illustrates the transcoding process when reactive-transcoding is enabled.

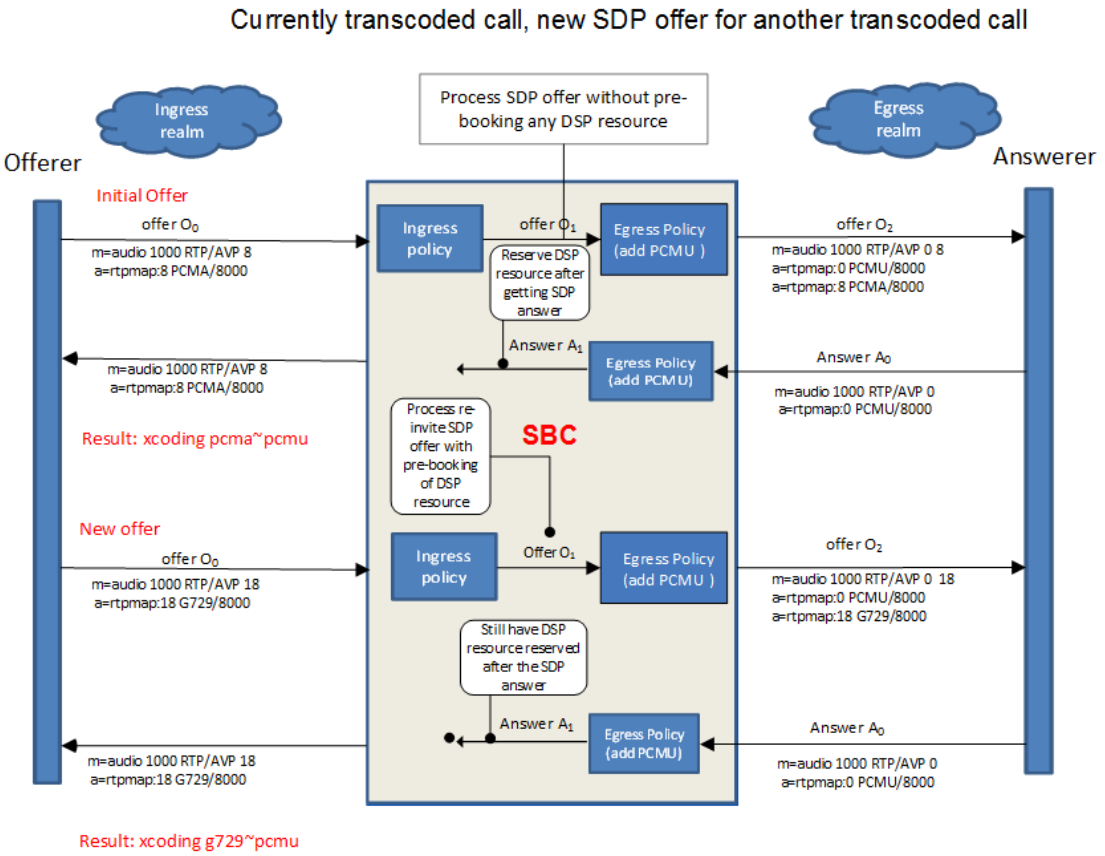


> **Note:**
>
> The DSPs will be reserved only for the call currently being transcoded. This avoids DSP exhaustion when there are multiple incoming offers.

Reactive Transcoding Examples

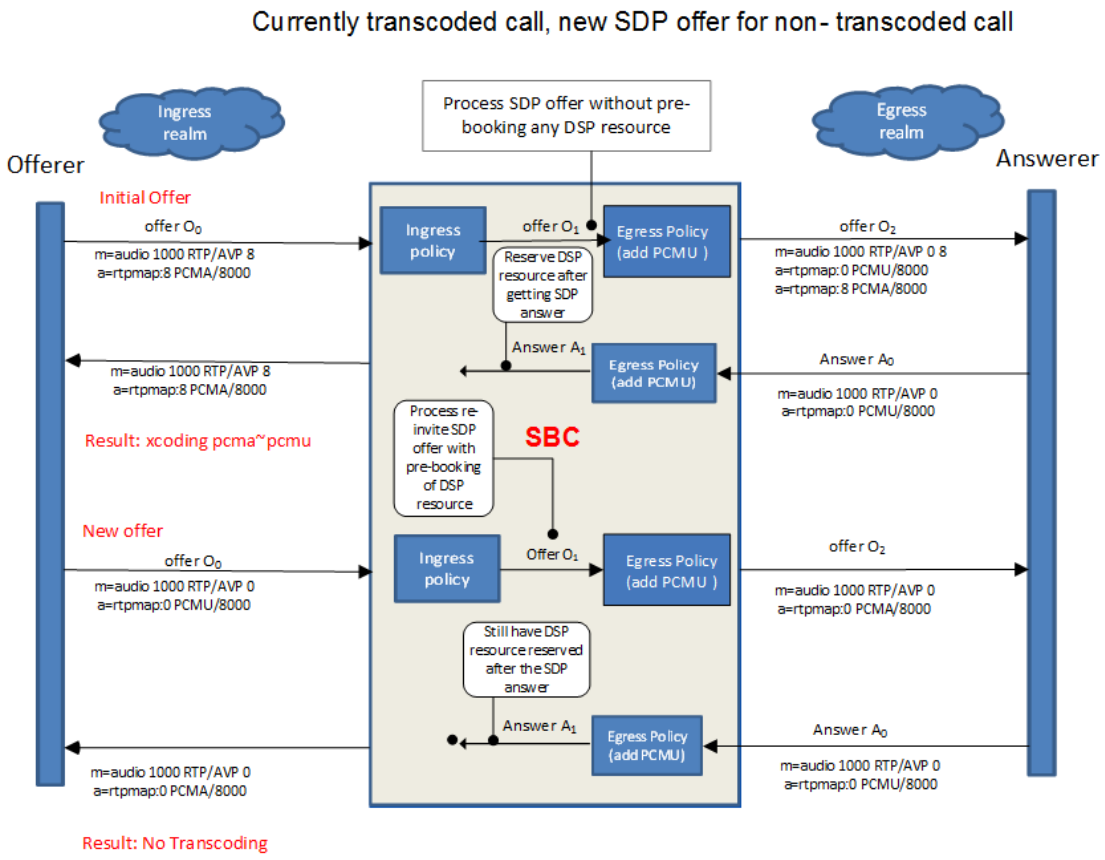
Scenario 1: Current Call : Transcoded, New offer: Transcoded

When a new offer is processed, the Oracle Communications Session Border Controller checks if the call is transcoded. Oracle Communications Session Border Controller pre-books DSP resources while processing new a SDP offer. After getting the SDP answer if the Oracle Communications Session Border Controller finds that the call still needs transcoding, it continues to keep the DSP resources.



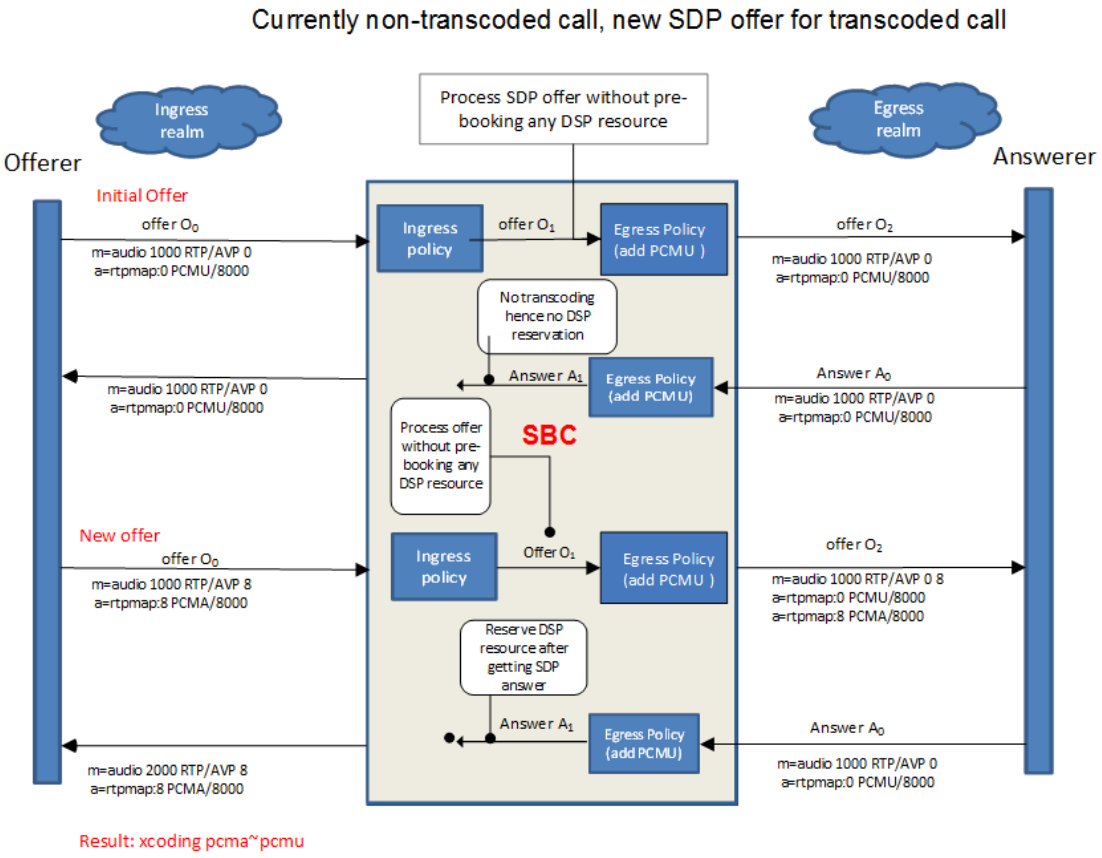
Scenario 2: Current Call : Transcoded, New offer: Non-Transcoded

When a Re-INVITE offer is processed, the Oracle Communications Session Border Controller checks if the call is already transcoded. Oracle Communications Session Border Controllerpre-books DSP resources during SDP offer of re-invite to avoid failure to grab DSP after SDP answer during DSP exhaustion case. After getting the SDP answer, if the Oracle Communications Session Border Controller finds that the call does not need transcoding any more, it releases the DSP resources.



Scenario 3: Current Call : Non-Transcoded, New offer: Transcoded

If a Re-INVITE is received when the reactive-transcoding mode is enabled, it will be handled as a regular offer. The DSP resource will only be reserved after receiving the SDP answer, as necessary.



Reactive Transcoding Mode Configuration

This procedure is used to configure the Oracle Communications Session Border Controller to dynamically reserve DSPs:

1. Access the **media-manager-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

2. Type select to begin editing.

```
ORACLE(media-manager-config)# select

ORACLE(media-manager-config)#
```

3. **reactive-transcoding**— Set this parameter to **enabled** for the SBC to perform reactive transcoding.

```
ORACLE(media-manager-config)#reactive-transcoding enabled
```

4. Type **done** to save your configuration.

Transcoding Configuration

The Oracle Communications Session Border Controller (SBC) uses codec policies to describe how to manipulate SDP messages as they cross the SBC. The SBC bases its decision to transcode a call on codec policy configuration and the SDP. Each codec policy specifies a set of rules to be used for determining what codecs are retained, removed, and how they are ordered within SDP.

Codec Policy Configuration

Codec policies describe how to manipulate SDP messages as they cross the Oracle Communications Session Border Controller (SBC). The SBC bases its decision to transcode a call on codec policy configuration and the SDP. Each codec policy specifies a set of rules to be used for determining which codecs are retained or removed, and how they are ordered within SDP.

When configuring transcoding, you create a codec policy and associate the policy to a realm. In the codec policy, you specify:

- Which codecs to allow and which codecs to deny within a realm.
- Which codecs to add to the SDP m= lines for an egress realm.
- The preferred order of codecs shown in an SDP m= line.
- The packetization time to enforce within a realm for transrating.

Terms Used in Codec Policies

Understand the following definitions for terms used in codec policies.

DTMF Codecs—Uncompressed codecs capable of properly transmitting a Dual-tone Multi-frequency (DTMF) waveform. Supported codecs: PCMU and PCMA.

FAX Codecs—Uncompressed codecs capable of properly transmitting a T.30 waveform. Supported codecs: PCMU and PCMA.

Signaling Codecs—Non-audio codecs interleaved into a media stream, and cannot be used on their own. Supported Codecs: Telephone-event and Comfort Noise (CN).

Comfort Noise Codecs—Interoperable codecs used to transcode silence to Comfort Noise on the SBC. Supported codecs: PCMA, PCMU, and G.726.

Disable an m= line in an SDP message—Use to set the m= line port to 0 (RFC 3264).

Enable an m= line in an SDP message—Use to set the m= line port to non-0 (RFC 3264). The m= line's mode attribute (for example, sendrecv, inactive, and rtcponly) is not considered.

Ingress Policy

Incoming SDP is first subject to the ingress codec policy. If no codec policy is specified in the realm config for the ingress realm, or the m= lines in the SDP offer are disabled (by a 0 port number), the SDP is transformed to O1 unchanged.

The ingress codec policy first removes all un-allowed codecs, as configured in the allow-codecs parameter by setting their port to 0 or removing the codecs from a shared m-line. For example, if two codecs share an m-line and one of them is un-allowed, the resulting m-line will

not include the un-allowed codec and its attribute lines will be removed. If a single codec is used, the resulting m-line will include the codec, but its port will be set to 0 and its attribute lines will remain. Next, the remaining codecs are ordered with the **order-codecs** parameter. Ordering is when the codec policy rearranges the codecs in the SDP m= line. This is useful to suggest the codec preferences to impose within the egress realm. O1 is then processed by the egress codec policy after a realm is chosen as the destination.

In practical terms, the ingress policy can be used for filtering high-bandwidth codecs from the access realm. It can also be used for creating a suggested, prioritized list of codecs to use in the ingress realm.

Egress Policy

The Oracle Communications Session Border Controller (SBC) applies the egress codec policy to the SDP that was processed by the ingress policy. The SBC applies the egress policy the SDP exits the system into the egress realm. When no egress codec policy is defined, or the SDP's m= lines are disabled (with a 0 port), the SDP is passed untouched from the ingress policy into the egress network.

The egress codec policy first removes all disallowed codecs in the **allow-codecs** parameter (<codec>:no). Codecs on the **add-codecs-on-egress** list are not removed from the egress policy regardless of the how the **allow-codecs** parameter is configured. If the result does not contain any non-signaling codecs, the ptime attribute is removed from the SDP. Codecs not present in O1 that are configured in the **add-codecs-on-egress** parameter are added to the SDP, only when O1 contains one or more transcodable codecs.

> **Note:**
>
> Transcoding can only occur for a call if you have configured the add-codecs-on-egress parameter in an egress codec policy.

If codecs with dynamic payload types (those between 96 and 127, inclusive) are added to the SDP, the lowest unused payload number in this range is used for the added codec.

The following rules are also applied for egress policy processing:

- When O1 contains at least one transcodable codec the system adds the codecs listed in the Egress policy to the SDP, as follows:
  - telephone-event—added only when O1contains at least one DTMF-suppported codec.
  - comfort-noise—added only when O1 contains at least one non-signaling transcodable audio codec, and when O2 contains a comfort noise interoperable codec (either present in O1 and allowed, or also added on egress).
  - T.38—added when there is no T.38 and there is at least one FAX-supported codec (G711Fall Back (FB)) in O1. T.38 added as a new m= image line to the end of SDP. When the egress policy does not allow G711FB, the SBC disables the m= line with the FAX-supported codec. Otherwise when G711FB is allowed; pass it through the regular offer processing allowing/adding only FAX-supported codecs.
  - G711FB, added when there is no G711FB and there is T.38 in O1. The system adds G711FB as a new m= audio line to the end of SDP.
    When the egress policy does not allow T.38, the SBC disables the m= image line. Otherwise if T.38 is allowed, passing it through the regular offer processing.
- When adding a codec on the egress side, the SBC checks to see if the payload presented is already in that direction's codec list. The system does this by matching all parameters for

that codec. This check includes the codec type itself. If present, the system uses that PT. If not, the system generates a new payload for the new codec.

When the result of the egress policy does not contain any non-signaling codecs (audio or video), the m= line is disabled by setting the port number to 0.

The m= line is ordered according to the rules for the order-codecs parameter.

All attributes, a= lines, ptime attribute, and all other unrecognized attributes are maintained from O1. Appropriate attributes for codecs added by the add-on-egress parameter are added to SDP. The rtpmap and fmtp parameters are retained for codecs not removed from the original offer. The result is O2.

You can use codec policies to normalize codecs and packetization time in the core realm, where the network conditions are clearly defined.

You can also use codec policies to force the most bandwidth-conserving codecs anywhere in the network.

Post Processing

If any errors are encountered during the Ingress and Egress policy application, or other violations of RFC3264 occur, the call is rejected. If O2 does not contain any enabled m= lines at the conclusion of the initial call setup, the call is rejected. If O2 does not contain any enabled m= lines at the conclusion of a reINVITE, the reINVITE is rejected and the call reverts back to its previous state.

allow-codecs

Use the **allow-codecs** parameter to configure the codecs that you want to allow and remove from the SDP. A blank list allows nothing, ***** allows all codecs, **none** removes all codecs, **:no** blocks the specific codec or class of media, and **:force** removes all non-forced codecs.

Use the following settings to configure the **allow-codecs** parameter:

- <codec>**:no**—blocks the specific codec
- *****—allow all codecs.
- <codec>**:force—**If any forced codec is present in an SDP offer, all non-forced codecs are stripped from the m- line.
- **audio:no**—audio m= line is disabled
- **video:no**—video m= line is disabled

For example, if you configure PCMU in the allow-codecs parameter, the PCMU codec, received in an SDP message is allowed to go on to the next step of transcoding processing, and all other codecs are removed.

The following list describes the order of precedence for removing codecs according to the codec policy:

1. <codec>**:no**—Overrides all other allow-codecs parameter actions.
2. **audio:no** and **video:no**. An allow-codecs line such as “allow-codecs PCMU audio:no” disables the PCMU m= line because audio:no has a higher precedence than the specific codec.
3. <codec>**:force**
4. <codec> Specific codec name and those codecs configured in the add-codecs-on-egress list.

5. ***—The lowest precedence of all flags. For example "**allow-codecs * PCMU:no**" allows all codecs except PCMU.

order-codecs

Use the **order-codecs** parameter to re-order the codecs in the m= line as the SDP is passed on to the next step. This parameter overwrites the order modified by the **add-codecs-on-egress** command, when relevant. Use the following syntax for this parameter:

- <blank>—Do not re-order codecs
- *—You can add a <codec> before or after the ***** which means to place all unnamed codecs before or after (the position of the *) the named codec. For example:
- <codec> *—Puts the named codec at the front of the codec list.
- * <codec>—Puts the named codec at the end of the codec list.
- <codec1 > * <codec2>—Puts <codec1> first, <codec2> last, and all other unspecified codecs in between them.
- <codec>—When the * is not specified, it is assumed to be at the end.

Any codec name is allowed in the **order-codecs** parameter, even those not defined or not transcodable. An * tells the order-codecs parameter where to place unspecified codecs with respect to the named codecs. Refer to the following examples.

- <blank>—Do not reorder m= line
- PCMU *—Place PCMU codec first, all others follow
- * PCMU—Place PCMU codec last, all others proceed PCMU
- G729 * PCMU—Place G729 codec first, PCMU codec last, all others remain in between
- PCMU—If * is not specified, it is assumed to be at the (PCMU *).

Add on Egress

Use the **add-codecs-on-egress** parameter to add a codec to the SDP's m= line only when the codec policy is referenced from an egress realm (except in one 2833 scenario). The codecs you enter for this parameter are added to the front of the m= line. Signaling codecs are added to the end of the m= line.

Transcoding can only occur if this parameter is configured. There is a special scenario for 2833 support where the add-codecs-on-egress parameter is configured for an ingress realm. See RFC 2833 Scenario 2 for details.

Packetization Time

Use the **packetization-time** parameter to specifiy a media packetization time in milliseconds (ms) to use within the realm referencing this codec policy. You must also enable the force ptime parameter to enable transrating in conjunction with configuring the packetization time. See Transrating for more information.

Set a User-Defined Ptime per Codec

To change the default packetization time (ptime) on the Oracle Communications Session Border Controller for a specific codec, you must create a media profile configuration element. In the **parameter** parameter, you can set the ptime to the value you want.

- Confirm that the system is in Superuser mode.

If you are adding ptime to a pre-existing media profile, then you must select the configuration that you want to edit (using the ACLI select command). If you are adding ptime to an undefined media profile, you must create the profile first.

> **Note:**
>
> The frames-per-packet parameter in the media profile configuration element is not used for setting a user defined ptime for that codec.

To configure a new ptime value for a codec:

1. Access the **media-profile** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

2. **name**—Type the name of the codec for which you are creating a new default ptime.

   ```
   ORACLE(media-profile)# name pcmu
   ```

3. **payload-type**—Enter the well-known payload type for this codec.

   ```
   ORACLE(media-profile)# payload-type 0
   ```

4. **parameter**—Set the ptime by typing **parameter**, a Space, **ptime=**, the new ptime value. Then press Enter. For example:

   ```
   ORACLE(media-profile)# parameter ptime=40
   ```

5. Type **done** to save your configuration.

Name a Codec Policy

The name of the codec policy is important not only because it uniquely identifies the policy, but because it is the name you will enter into the **codec-policy** parameter in your realm configuration. It is important to apply the correct policy to the appropriate realm.

To set the codec policy's name:

- **name**—Set the name for this codec policy, and note it for future reference when you apply codec policies to realms. This parameter is required, and has no default.

  ```
  ORACLE(codec-policy)# name private
  ```

Order Codecs

In the codec policy, you can specify the order that codecs appear in the SDP offer or answer.

To configure an order which codecs appear in the offer:

- **order-codecs**—Enter the order in which you want codecs to appear in the SDP offer or answer. See "order-codecs" for the possible ways to set the order.

```
ORACLE(codec-policy)# order-codecs G711 * G729
```

Allow, Remove, and Add Codecs for Transcoding

The Oracle Communications Session Border Controller (SBC) allows and removes codecs by way of the following parameters in codec-policy.

allow-codecs

The SBC uses the list that you create in the **allow-codecs** parameter in codec-policy to pass through the codecs that you want by allowing them to remain in the SDP for the next step. The SBC removes codecs that do not match entries on the list. This parameter is required.

Enter a list of codecs that are allowed to pass through the SBC. Use the syntax in the Transcodable Codecs section of this chapter. To allow all codecs, enter an asterisk (*).

```
ORACLE(codec-policy)# allow-codecs *
```

When multiple items are added, enclose them in quotes. For example:

```
ORACLE(codec-policy)# allow-codecs G729 G711 AMR
```

add-codecs-on-egress

The SBC uses the list that you create in the **add-codecs-on-egress** parameter in codec-policy to set the codecs that the SBC adds to an offer when they are not present in the offer. This parameter applies only to the egress policy.

Enter the codecs that you want added to the SDP offer for the egress codec policy. If you leave this parameter blank, the SBC does not add codecs to the SDP answer. You cannot use this parameter as a wildcard.
To remove items from the **allow-codecs** list, simply replace the **add** command you see in these example with **delete** and remove the items you want.

```
ORACLE(codec-policy)# add-codecs-on-egress G729
```

Remove Codecs

To remove items from the **allow-codecs** and **add-codecs-on-egress** lists, replace the **add** command with **delete** and remove the items you want.

```
ORACLE(codec-policy)# delete-codecs G729
```

```
ORACLE(codec-policy)# delete-codecs-on-egress G729
```

> **Note:**
>
> When you need to modify the list of configured codecs, you must enter the complete list at one time.

Configure a Codec Policy

- Confirm that the system is in Superuser mode.

You can use a single codec policy for any number of realms.

1. Access the **codec-policy** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# codec-policy
ORACLE(codec-policy)# select
```

2. Do the following:

Name	Description
Name	Type a name for this policy.
Allow codecs	Type the name of the codecs that you want to allow on ingress, any exceptions. See "allow-codecs."
Add codecs on egress	Type the names of the codecs that you want to add on egress. See "Add on Egress."
Order codecs	Type the names of the codecs in the order you want them processed. See "order-codecs."
Packetization time	Set the time, in milliseconds, that you want the system to wait before enforcing packetization on the outgoing SDP offer. See "Packetization Time." Requires enabling Force Ptime. Default: 20. Range: 0-4294267295.
Force ptime	Enable to enforce the packetization time.
Secure dtmf cancellation	Enable to completely remove all DTMF tones on ingress to make them undetectable on egress. Default: Disabled.
Dtmf in audio	Set how you want the system to handle DTMF in audio streams. Default: Disabled. Valid values: Disabled | Preferred | Dual.
Tone detect renegotiate timer	Set the time in milliseconds that you want the system to wait before sending a REINVITE if the SBC does not receive a REINVITE from the endpoint. Default: 500. Range: 50-32000.
Reverse fax tone detection reinvite	Enable to force the SBC to send a REINVITE to a realm other than the one on which the FAX tone detection is enabled. Default: Disabled.
Evrc tty baudot transcode	Enable to transcode EVRC, TTY, and TDD calls to Baudot in EVRC-G7.11 transcoded calls.

3. Type **done** to save your configuration.

Configure Transrating

The following procedure explains how to configure transrating for a codec policy. You must apply this codec policy as an egress codec policy.

- Confirm that the system is in Superuser mode.

If you are adding support for this feature to an existing configuration, you must select the specific configuration instance using the ACLI **select** command.

To configure forced ptime for a codec policy:

1. Access the **codec-policy** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# codec-policy
ORACLE(codec-policy)# select
```

2. Select the **codec-policy** object to edit.

```
ORACLE(codec-policy)# select
<name>:
1:  name=cp1
2:  name=cp2

selection: 2
ORACLE(codec-policy)#
```

3. **force-ptime**—Set this parameter to **enabled** to enable forced ptime for this codec policy.
4. **packetization-time**—Enter the ptime in milliseconds (ms) to use in the realm where this codec policy is active. Default: 20ms. Valid values: 10, 20, 30, 40, 50, 60, 70, 80, 90 ms.
5. Type **done** to save your configuration.

Apply a Codec Policy to a Realm

After you configure a codec policy, you apply it to a realm by policy name.

- Confirm that the system is in Superuser mode.

To apply a codec policy to a realm:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0
```

```
selection: 1
ORACLE(realm-config)#
```

3. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. This value is the same as the one you entered in the name parameter for the codec policy you want to use for this realm. There is no default for this parameter.

```
ORACLE(realm-config)# codec-policy private
```

4. Type **done** to save your configuration.

Secure DTMF Cancellation

For security and privacy reasons, you can remove all Dual-Tone Multi-Frequency (DTMF) information that the Oracle Communications Session Border Controller (SBC) processes by enabling `secure-dtmf-cancellation` within the codec-policy. For example, you might want to cancel all DTMF tones when processing credit card numbers, debit card numbers, PIN numbers, and other DTMF-based passwords that you want to remove from the media stream. When an incoming call requires DTMF cancellation, the SBC uses the built-in detection and cancellation mechanism to completely remove all sixteen DTMF tones on ingress making the tones undetectable on egress. (0-9, *, #, A,B,C,D)

The SBC supports secure DTMF cancellation for all use cases and call scenarios that include in-band DTMF detection, such as DTMF in-band to SIP-INFO, DTMF in-band to RFC2833, DTMF in-band to None and DTMF over RFC2833. Oracle recommends that you enable this feature for codec policies that use codecs that support reliable in-band DTMF detection on ingress, such as PCMU.

Standard DTMF cancellation can leave some residual signal energy at the beginning and ending of each DTMF digit. The practical result of this is that an important piece of data, such as a card number, is still detectable within the egress stream. The SBC performs a second function with `secure-dtmf-cancellation` to remove this leftover signaling from the media stream. To do this, the SBC uses process timing as opportunities to identify any immediately identified or residual DTMF and cancels it.

The result is silence for the entire duration of each DTMF digit and the elimination of all stray DTMF data.

This feature works on all flows, including TLS/SRTP.

In addition to enabling `secure-dtmf-cancellation` in the codec policy, you must set `dtmf-in-audio` in the codec policy such that it is not disabled.

> **Note:**
>
> An endpoint configured as secure at the session startup stays in the secure mode throughout the call because you cannot re-configure the DTMF cancellation mode during the call.

> **Note:**
>
> The **secure-dtmf-cancellation** option does not remove DTMF bleed unless transcoding is enabled for the call.

Enable Secure DTMF Cancellation

When you want to completely remove all traces of Dual-Tone Multi-Frequency (DTMF) information that the Oracle Communications Session Border Controller (SBC) processes, enable secure-dtmf-cancellation to make the DTMF tones undetectable.

Secure-dtmf-cancellation requires that you also enable dtmf-in-audio. If dtmf-in-audio is not already enabled, you can enable both attributes in the following procedure.

1. Access the **codec-policy** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# codec-policy
ORACLE(codec-policy)# select
```

2. Select the **codec-policy** object to edit.

```
ORACLE(codec-policy)# select
<name>:
1:  name=cp1
2:  name=cp2

selection: 2
ORACLE(codec-policy)#
```

3. Enable the **secure-dtmf-cancellation** parameter.

```
ORACLE(codec-policy)#secure-dtmf-cancellation enabled
```

4. Enable the **dtmf-in-audio** parameter.

```
ORACLE(codec-policy)#dtmf-in-audio dual
```

5. Type **done** to save your configuration.

Default Media Profiles

The Oracle Communications Session Border Controller (SBC) contains a set of default media profiles that define characteristics of well-known IANA codecs. You cannot view the default media profiles configurations, but you can override them by configuring identically-named media profile configuration elements.

Transcodable codecs are a subset of the default media profiles, which the SBC can transcode between.

Preferred Default Payload Type

When the Oracle Communications Session Border Controller (SBC) adds a codec with a dynamic payload type to SDP, it uses the lowest unused payload number. You can configure a preferred payload type for a dynamic codec by creating an override media profile. The override makes the SBC use your preferred payload type for insertion into SDP. If you configure a

dynamic codec to use a preferred payload type and that payload type is already in use, the codec will still be inserted into SDP, but with the first available dynamic payload type.

For example, suppose you create a media profile for telephone-event with a payload type of 101. If telephone-event is added to SDP, and payload type 101 is already in use in the SDP, the SBC will use the first available payload type in the 96-127 range when adding telephone-event.

Redefining Codec Packetization Time

You can configure a media profile with a packetization time (ptime) that overrides the default ptime of the codec. Transcoding functions look up and use default ptimes when not specified in offered or answered SDP. The default ptime for most audio codecs is 20ms, but some are 30ms.

To change the default ptime for a codec, you must create a media profile that overwrites the default ptime parameter with your new ptime. When SDP is received with no 'a= ptime' attribute or when adding the codec to egress SDP, the system uses the newly configured ptime.

To set a new default ptime for a media profile, type "ptime=<x>" in the parameters parameter, where <x> is the new default ptime.

mptime Support for Packet Cable

The SDP specification lacks the ability to specify unique packetization times (ptime) per codec when more than one codec is listed in an m= line. The ptime attribute is not related to a specific codec but to the entire m= line. When multiple codecs appear on a single m= line, the PacketCable mptime attribute can specify different packetization times for each codec.

The Oracle Communications Session Border Controller (SBC) adheres to PKT-SP-NCS1.5-I01-050128 and PKT-SP-EC-MGCP-I06-021127 for processing and generating mptime. The mptime line uses an integer to indicate the ptime for each corresponding codec in the m= line. The dash character, "-", on an mptime line is used for non-packetized codecs, such as CN or telephone-event.

When the SBC receives an invalid mptime, it is ignored and removed. When a valid mptime is received in the incoming SDP, the SBC uses its values for the ptimes of each corresponding codec and sends a valid mptime line in the outgoing SDP.

Valid:

```
m=audio 10000 RTP/AVP 0 96 8
a=mptime:20 - 30
a=rtpmap:96 telephone-event/8000
```

Valid: 'ptime' attribute is ignored

```
m=audio 10000 RTP/AVP 0 8
a=mptime:20 30
a=ptime:30
```

Invalid: dash cannot be first mptime value

```
m=audio 10000 RTP/AVP 96 0
a=mptime: - 20
```

When SBC includes an mptime in an outgoing SDP, it always adds a ptime attribute with the value of the most preferred codec. This is done to increase the interoperability with devices that do not support mptime.

Media Profile Configuration

Media profiles must be created and then defined when you want to override the Oracle Communications Session Border Controller's default media profiles.

Media Type Subnames

You can define multiple versions of a media profile for a single codec by using the subnames feature. You can then reference the new media profile by a combination of the media profile name and media profile subname.

Some media types are not unique per just their value in an SDP m= line, they must be uniquely identified by looking at additional SDP parameters. For example, you can define a media profile for G729, when only the parameter and value **annexb=yes** is present in the SDP. By creating a media profile + subname that defines both a media type and parameter, you can perform various operations on G729 only when **annexb=yes** is encountered.

Some applications of media type subnames are:

- maintaining different versions of the same codec with different bandwidth ceilings
- maintaining different versions of the same codec with different ptimes
- grouping codecs by using customer as a subname
- grouping codecs by using realm as a subname

SDP Parameter Matching

This feature matches parameters in the **a=fmtp**, codec-specific SDP **a=** line. It does not try to match a global **m=** line attribute like **a=ptime**.

Using Subnames with Codec Policies

Media profiles are defined and referenced in the ACLI by a name and subname in the following format

```
<name>::<subname>
```

If no subname has been created for a media profile, you may continue using the media profile name without any subname specifier.

For example, to remove a media profile and subname configured as PCMU::customer1 from all SDP entering the egress realm, you would configure the codec policy **allow-codecs** parameter as follows:

```
allow-codecs PCMU::customer1:no
```

media-profile subtype Configuration Restrictions

media-profiles are subject to stringent configuration restrictions. You must avoid creating a **media-profile** with configured **subtype** parameter that does not substantively differ (in all additional parameters) from the default (unconfigured) media profile. An example of an invalid configuration is **media-profile**, **name** of g729, and a **media-profile**, **subname** of g729, with no additional parameter configurations other than the default values. Such configuration can cause unexpected behaviors and must be avoided.

Subname Syntax With the Wildcard Character

You can use the wildcard character in one or both portions (name and subname) of a media type and subname pair:

- When you use the wildcard character the **name** portion of the value, you can provide a specific subname that the Oracle Communications Session Border Controller (SBC) uses to find matching media profiles.
- When you use the wildcard character in the subname portion of the value, you can provide a specific **name** that the SBC uses to find matching media profiles.

The following table defines and explains using a subname with the wildcard character and shows the syntax:

Syntax	Example Value	Description
<name>	PCMU	Matches any and all media profiles with the name value configured as PCMU. This entry has the same meaning as a value with this syntax: <name>::*.
<name>::	PCMU::	Matches a media profile with the name with the name value configured as PCMU with an empty subname parameter.
<name>::*	PCMU::*	Matches any and all media profiles with the name value configured as PCMU with any and all subname configured.
<name>::<subname>	PCMU::64k	Matches a media profiles with the name with the name value configured as PCMU with the subname parameter set to 64k.
*	*	Matches anything, but does not have to be a defined media profile.
::	*::*	Matches any and all media profiles, but requires the presence of media profile configurations.
*::<subname>	*::64k	Matches all media profiles with this subname. You might have a group of media profiles with different names, but the same subname value.
*::	*::	Matches any media profiles with an empty subname parameter.
::	::	Invalid
::*	::*	Invalid

Wildcard Character in add-codecs-on-egress Limitation

It is important to note that you may not configure **add-codecs-on-egress** with a wildcard character in a subname in a codec policy. You may only add a specific instance of a media type.

Valid:

```
add-codecs-on-egress PCMU
add-codecs-on-egress PCMU::customer1
```

Invalid:

```
add-codecs-on-egress PCMU::*
```

Configure Media Type and Subname

To use media type subnames with a codec policy, you must first configure a media profile and subname. Then you can configure a codec policy with a media type and subname pair for your application

To use configure a media type and subname:

1. Access the **media-profile** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

2. **name**—Type the name of the codec for which you are creating a new default ptime.

   ```
   ORACLE(media-profile)# name g729
   ```

3. **subname**—Enter a description for the use of this subname

   ```
   ORACLE(media-profile)# subname annexb-yes
   ```

   You may now configure this subname's unique attributes. PCMU is created with ptime of 30 in this example.

4. **parameters**—Set the ptime by typing **parameter**, a Space, **ptime=**, the new ptime value. Then press Enter. For example:

   ```
   ORACLE(media-profile)# parameter annexb=yes
   ```

   > **Note:**
   >
   > Remember to configure all additional, required media profile parameters, or they will inherit default values.

5. Type **done** to save your configuration.

Configure a Codec Policy with a Media Type with a Subname

To configure a codec policy with a media type with subname:

1. Access the **codec-policy** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# codec-policy
ORACLE(codec-policy)# select
```

2. Select the **codec-policy** object to edit.

```
ORACLE(codec-policy)# select
<name>:
1:  name=cp1
2:  name=cp2

selection: 2
ORACLE(codec-policy)#
```

3. **allow-codecs**—Enter a list of codecs that this codec policy allows or denies from passing through the Oracle Communications Session Border Controller. To allow all codecs, enter an asterisk (*).

```
ORACLEcodec-policy# allow-codecs g729::annexb-yes:no
```

4. Type **done** to save your configuration.

Updating the b=AS line for Transcoded Calls

The Oracle Communications Session Border Controller (SBC) can evaluate transcoding call scenarios and mitigate between end stations and policy servers to request appropriate bandwidth. You can configure a specific value to be included in the SDP's b=AS line to ensure that it requests the correct bandwidth for transcoded calls. Depending on the transcoding scenario, this value may or may not be used.

The SBC can change the egress answer b=AS line in a call to a value based on the media profile configuration of the codec to which the media is being transcoded. This ensures that the SBC makes bandwidth requests that are applicable to the leg on which that codec is used.

Use the following syntax to specify 124 kbps for the b=AS line a in media profile.

```
(media-profile)#as-bandwidth 124
```

The default value is zero, meaning it is disabled. The range is from 0 to 4294967295, measured in kbps.

The SBC can modify a b=AS: line at both the SDP media level and SDP session level. Session level b=AS modification does not use an **as-bandwidth** setting.

The SBC updates a media level b=AS: line to the configured **as-bandwidth** value only for the Egress SDP Answer, not the offer.

The SBC modifies a session level b=AS: value only if every m-line in the SDP also has a b=AS: value. When an incoming message has SDP with a session level b=AS value, and every m-line in the SDP has a b=AS value, then the SBC updates the outgoing message session level b=AS value to the sum of the transcoding/IPv4-IPv6 adjusted media level b=AS values in the outgoing message. When an incoming message has SDP with a session level b=AS value, but not every m-line in the SDP has a b=AS value, the SBC does not modify the session level b=AS value in the outgoing message.

When the SBC updates a b=AS: line in the egress answer's SDP to a configured **as-bandwidth** value, and the original ingress offer SDP already had a b=AS: line, the SBC changes the egress answer SDP's b=AS: back to the ingress offer's b=AS: value, not to the configured **as-bandwidth** value.

Important operational considerations include:

- The value can be understood as IP neutral, meaning the system recognizes when the call is IPv4 to IPv6 interworking, and adjusts the value of the b=AS line to compensate for the IP version of the applicable leg. The SBC does this for both transcoded and non-transcoded calls in both the egress offer and answer.
- The configuration has no effect when the initial message received has no b=AS line.
- The configuration has no effect when the m line is not transcoded.
- The system does not change a session b=AS value when the SDP has a media line with b=AS value of 0.

Codec and Conditional Codec Policies for SIP

The Oracle Communications Session Border Controller (SBC) can add, strip, and reorder codecs for SIP sessions. This builds on the SBC pre-existing abilities to route by codec and re-order one codec in an SDP offer by allowing you to configure the order of multiple codecs and to remove specific codecs within the media descriptions in SDP offers.

You can enable the SBC to perform these operations on SDP offers by configuring codec policies. Codec policies are sets of rules that specify the manipulations that you want the SBC to perform on SDP offers. The SBC applies the policies on an ingress and egress basis using the realm and session agent configurations.

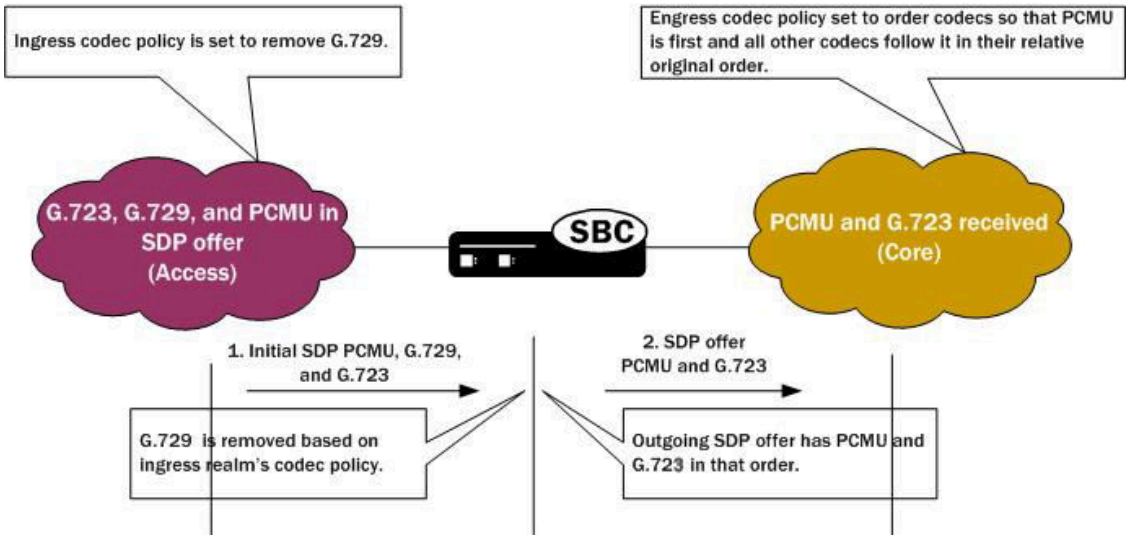
SBC supports the following types of codec policies:

- Ingress policy—applies to the SDP offer for incoming traffic
- Egress policy—applies to the SDP offer for traffic leaving the SBC
- Conditional policy—applies to the SDP offer for traffic leaving the SBC. A conditional policy differs from an egress policy in providing the capability to perform standard codec manipulations (add, strip and re-order) dynamically, based on the codec list and associated parameters contained in the original SDP offer. See Conditional Codec Policies for specific details regarding the use and construction of conditional policies.

The SBC applies codec policies during the offer phase of media format negotiation. When you enable codec manipulation, the SBC performs the modification according to the specific policy and forwards the traffic.

For example, when the SBC receives a SIP INVITE with SDP, it refers to the realm through which the INVITE arrived and performs any manipulations specified by the ingress codec policy that you assigned to the ingress realm. With the media description possibly changed according to the ingress codec policy, the SBC passes the SDP offer to the outgoing realm and applies the egress codec policy. Note that the SDP to be evaluated by the egress codec policy

may match the original SDP, or it may have been changed during transit through the ingress realm. After applying the egress coded policy, the SBC forwards the INVITE.



Because the offer-answer exchange can occur at different stages of SIP messaging, the assigned ingress and egress roles follow the media direction rather than the signaling direction. It might be, for example, that the offer is in an OK that the SBC modifies.

You can apply codec policies to realms and to session agents. Note that codec policies configured in session agents take precedence over those applied to realms. The system does not require both an ingress policy and an egress policy for either realms or for session agents. When either one is unspecified, no modifications take place on that side. When you specify neither an ingress nor egress policy, the SBC forwards SDP offers as received.

Codecs in Relationship to Media Profiles

For each codec that you specify in a codec policy, there must be a corresponding media profile configuration on the Oracle Communications Session Border Controller. You configure media profiles in the ACLI by way of the session-router configuration. In the profiles, you can specify codec type, transport protocol, required bandwidth, and a number of constraints.

Manipulation Modes in Codec Policies

You can configure a codec policy to perform several different kinds of manipulations:

- Allow—List of codecs that are allowed for a certain codec policy. When a codec does not appear on the allow list, the Oracle Communications Session Border Controller (SBC) removes it. You can use the wildcard character (the asterisk *) in this list, to allow all codecs. You can create the following exceptions to the allow list that contains a wildcard.
  - You make an exception to the wildcard list of codecs by entering the codecs that are not allowed with a **no** attribute. This tells the SBC to allow all codecs except the ones you specify.

    ```
    ORACLE(codec-policy)# allow-codecs (* PCMA:no)
    ```

  - You can also create exceptions to allow lists such that audio or video codecs are removed. However, when the allow list specifies the removal of all audio codecs and an INVITE arrives at the SBC with only audio codecs, the SBC behaves in accordance

with RFC 3264. This means that the resulting SDP contains one attribute line, with the media port for the media line set to 0. The terminating side supplies new SDP in its reply because the result of the manipulation is the same as an INVITE with no body.

```
ORACLE(codec-policy)# allow-codecs (* audio:no)
```

- Order—List of the codecs where you specify their preferred order in the outgoing media offer. The SBC arranges matching codecs according to the rule you set, and any remaining codecs are added to the list in the same relative order as in the incoming media offer. When your list specifies a codec that is not present, the ordering proceeds as specified but skips the missing codec.
  You can use an asterisk (*) as a wildcard in this list, too. The placement of the asterisk is important, as shown in the following examples:
  - For an order rule set this way

    ```
    ORACLE(codec-policy)# order (A B C *)
    ```

    codecs A, B, and C will be placed at the front of the codec list in the order specified. All other codecs in the offer will follow A, B, and C in the same relative order as in the original SDP offer.
  - For an order rule set this way:

    ```
    ORACLE(codec-policy)# order (* A B C)
    ```

    codecs A, B, and C will be placed at the end of the codec list in the order specified. All other codecs in the offer will come before A, B, and C in the same relative order as in the original SDP offer.
  - For an order rule set this way

    ```
    ORACLE(codec-policy)# order (A * B C)
    ```

    codec A will be placed at the beginning of the codec list, followed by all other codecs in the offer in the same relative order as in the original SDP offer. B and C will end the list.
- Force—An attribute you can use in the allow list with one codec to specify that all other codecs are stripped from the outgoing offer. You can specify multiple forced codecs in your rules.
  - When you set multiple codecs in the allow list and one of them is forced, the outgoing offer contains the forced codec.
  - When you set multiple codecs in the allow list and the one that is forced is not present in the offer, the SBC selects a non-forced codec for the outgoing offer.

    ```
    ORACLE(codec-policy)# allow (PCMU G729:force)
    ```

    You cannot use the force attribute with an allow list that contains a wildcard.

> **Note:**
>
> The Force attribute can only be applied to ingress realms receiving the offers, and is not applied to egress realms.

- No—An attribute that allows you to strip specified codecs or codec types from an allow list that contains a wildcard.

```
ORACLE(codec-policy)# allow (* PCMA:no)
```

In-Realm Codec Manipulation

In addition to applying codec policies in realms, the realm configuration supports a setting for determining whether or not you want to apply codec manipulation to sessions between endpoints in the same realm.

You can use In-realm codec manipulation for simple call flows that traverse two realms. When the originating and terminating realms are the same, the Oracle Communications Session Border Controller (SBC) checks to see if you enabled this capability. If you enabled it, the SBC performs the specified manipulations. When you do not enable this capability, or when you disable the realm's media management in realm (**mm-in-realm**) setting, the SBC does not perform codec manipulations.

For more complex call scenarios that involve call agent or re-initiation of a call back to the same realm, the SBC does not perform in-realm codec manipulation.

Conditional Codec Policies

A codec policy performs actions conditionally when any of its parameters includes a conditional value. A conditional value includes a target codec paired with a requirement for executing the action. The Oracle Communications Session Border Controller manipulates SDP according to this value pair when the ingress SDP or a previous manipulation to the SDP meets the condition criteria. You can configure conditional manipulation by extending upon the syntax of the following core parameters in the codec-policy configuration element:

- allow-codecs,
- add-codecs-on-egress, and
- order-codecs.

The system establishes conditions on a codec policy as a sequence of allowing, adding, and re-ordering. Each step in this sequence can occur with or without conditions. Allowing is required. Any applied policy, whether or not it is conditional, without an allow blocks all traffic. Allow all, using the wildcard asterisk character is a typical setting. Adding applies only to egress policies.

To establish conditions, you configure the parameter with pairs that consist of target codecs followed by the conditions that trigger the action. Each policy parameter can include one or more of these pairs. When configuring a parameter with multiple values, you enclose them within parenthesis, whether or not there are conditions.

The system processes all policies serially, regardless of whether any includes a condition. The system first determines which codecs to allow, and then which to add (none on ingress), then the codec order. The system also processes parameter values serially. You must configure all parameter values based on what may have been changed previously. This is particularly important when using both ingress and egress codec policies. Consistent with this serial

concept, egress policies operate on what the system presents to them, which includes the results of any ingress policies.

Note that conditional **order-codecs** are often done used in conjunction with **add-codecs-on-egress** to define the location of codecs you add to the list presented at egress. When **order-codecs** is not used, the system places all added codecs at the front of the list in the order they were added. It is often useful to place an added codec in a different position, using **order-codecs**.

Conditional Codec Lists

Conditional codec policies are constructed using existing ACLI configuration commands — **allow-codecs**, **add-codecs-on-egress**, and **order-codecs** — in conjunction with keywords and operators. Conditions are defined by a continuous character string (no spaces allowed) that starts with the **:(** character sequence and is terminated by a closing parenthesis. For example,

```
ORACLE(codec-policy)# allow-codecs PCMU:(!G729)
```

which can be interpreted as — allow PCMU if the G729 codec is not in the SDP offer after ingress codec policy processing.

An example of using **add-codecs-on-egress** is:

```
ORACLE(codec-policy)# add-codecs-on-egress PCMU:(PCMA)
```

which can be interpreted as — add PCMU if PCMA codec is in the SDP offer after ingress codec policy processing.

If PCMA is in the SDP offer after ingress codec policy processing, the **add-codecs-on-egress** ACLI command is treated as **add-codecs-on-egress PCMU**. If PCMA is not in the SDP offer after ingress codec policy processing, **add-codecs-on-egress** is treated as empty.

Both the conditioned codec and/or the condition itself can contain subnames. For example,

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::TEST0)
```

which can be interpreted as — add AMR::ONE if AMR::TEST0 codec is in the SDP offer after ingress codec policy processing.

Codecs contained in the condition can be wildcarded. For example,

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::*)
```

which can be interpreted as — add AMR::ONE if any AMR codec is in the offer after ingress codec policy processing.

An example of using **order-codecs** is:

```
ORACLE(codec-policy)# order-codecs (PCMU:(PCMU) *)
```

which can be interpreted as — set the codec order to PCMU followed by the list after ingress codec policy processing if PCMU is not present. When the system adds a codec, the codec

goes to the end of the offered list. This conditional format may be used to place an added codec at the front of list.

Conditional Codec Operators

Three logical operators are available to construct conditional lists

the OR operator (|)

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::*|PCMU)
```

which can be interpreted as — add AMR::ONE if any AMR:: codec is in the SDP offer after ingress codec policy processing, or if PCMU is in the SDP offer after ingress codec policy processing.

the AND operator (&)

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::*&PCMU)
```

which can be interpreted as — add AMR::ONE if any AMR:: codec is in the SDP offer after ingress codec policy processing, and if PCMU is in the SDP offer after ingress codec policy processing.

the NOT operator (!)

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(!AMR::*)
```

which can be interpreted as — “add AMR::ONE if no AMR:: codec is in the SDP offer after ingress codec policy processing.

Each operator applies only to the codec immediately following it. Operators are processed left to right until all conditions have been tested. The condition result is accumulated as each of the conditions is processed. For example:

```
ORACLE(codec-policy)#add-codecs-on-egress AMR::ONE:
(!AMR::TEST1&AMR::TEST0|AMR::TEST2)
```

- Test SDP offer for AMR::TEST1 (a NOT operation).
  If AMR:TEST1 is NOT present, set accumulated result to TRUE.

  If AMR:TEST1 is present, set accumulated result to FALSE.
- Test SDP offer for AMR::TEST0 (an AND operation).
  If AMR is present, no change to accumulated result.

  If AMR is not present, set accumulated result to FALSE.
- Test SDP offer for AMR::TEST2 (an OR operation).
  If AMR is present, set accumulated result to TRUE.

  If AMR is not present, no change to accumulated result.

Multiple conditions can be concatenated; in this case, individual conditions are separated by SPACE characters and the ACLI command argument is bracketed with double quotation marks ...). For example:

```
ORACLE(codec-policy)#add-codecs-on-egress (PCMU G729:(G726) G723:(PCMA))
```

- PCMU is unconditionally added to the egress codec list.
- Process the first condition — G729:(G726)
  If G726 is present, the result is TRUE; add G729 to the egress codec list.

  If G726 is not present, the result is FALSE; do not add G729 to the egress codec list.
- Process the second condition — G723:(PCMA)
  If PCMA is present, the result is TRUE; add G723 to the egress codec list.

  If PCMA is not present, the result is FALSE; do not add G723 to the egress codec list.

Codec Policies Instructions and Examples

The following topics contain instructions and examples show how to configure codec policies and apply them to realms and session agents. The instructions and examples also show you how to configure settings for in-realm codec manipulation.

Create a Codec Policy

- Confirm that the system is in Superuser mode.

To create a codec policy:

1. Access the **codec-policy** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# codec-policy
   ORACLE(codec-policy)# select
   ```

2. **name**—Enter the unique name for the codec policy. This is the value you will use to refer to this codec policy when you apply it to realms or session agents. This parameter is required and is empty by default.
3. **allow-codecs**—Enter the list of codecs to allow for this codec policy. In your entries, you can use the asterisk (*) as a wildcard, the force attribute, or the no attribute so that the allow list you enter directly reflects your configuration needs. Enclose entries of multiple values in parentheses (( )). For more information, see Manipulation Modes.

   The codecs that you enter here require corresponding media profile configurations.

   allow-codecs—Use to construct ingress, egress, or conditional codec policies. For details of conditional codec policies, see Conditional Codec Policies.
4. **add-codecs-on-egress**—Enter the codecs that the Oracle Communications Session Border Controller adds to an egress SDP offer when that codec is not already there. This parameter applies only to egress offers. For more information, see Manipulation Modes.

   The codecs that you enter here must have corresponding media profile configurations.

   add-codecs-on-egress—Use to construct ingress, egress, or conditional codec policies. For more information on conditional codec policies, see Conditional Codec Policies.
5. **order-codecs**—Enter the order in which you want codecs to appear in the outgoing SDP offer. You can use the asterisk (*) as a wildcard in different positions of the order to directly reflect your configuration needs. Enclose entries of multiple values in parentheses (( )). For more information, see Manipulation Modes.

   The codecs that you enter here require corresponding media profile configurations.

   order-codecs—Use to construct ingress, egress, or conditional codec policies. For more information on conditional codec policies, see Conditional Codec Policies.

6. Save your configuration.

   The following example shows a codec policy configuration:

```
codec-policy
        name            private
        allow-codecs    g723:no pcmu video:no
        order-codecs    pcmu *
```

Apply a Codec Policy to a Realm

- Confirm that the system is in Superuser mode.

Note that codec policies defined for session agents always take precedence over those defined for realms.

To apply a codec policy to a realm:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. If you are applying a codec policy to a pre-existing realm, you must select (using the ACLI **select** command) the realm that you want to edit.
3. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
4. Save your configuration.

Apply a Codec Policy to a Session Agent

- Confirm that the system is in Superuser mode.

Note that codec policies that are defined for session agents always take precedence over those that are defined for realms.

To apply a codec policy to a realm:

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. If you are a codec policy to a pre-existing session agent, you must select (using the ACLI **select** command) the realm that you want to edit.
3. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
4. Save your configuration.

In-Realm Codec Manipulations

- Confirm that the system is in Superuser mode.

To enable in-realm codec manipulations:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. If you are adding support for in-realm codec manipulations to an existing realm, you must select (using the ACLI **select** command) the realm that you want to edit.
3. **codec-manip-in-realm**—Enter the name of the codec policy that you want to apply to this realm. Default: disabled. Valid values: enabled | disabled.
4. Save your configuration.

Pooled Transcoding

Pooled transcoding refers to a deployment model involving two or more Oracle Communications Session Border Controllers (SBC). The first SBC is an access SBC (referred to as an A-SBC) acting as the P-CSCF, and the others are one or more SBCs equipped with transcoding hardware (referred to as a T-SBC). The T-SBC provides transcoding resources—a pool—that the A-SBC can invoke on-demand.

In the pooled transcoding model, the A-SBC sits between realms or between user endpoints that require transcoding between their preferred codecs. This deployment model conserves resources on both the A-SBC and the T-SBC. While the A-SBC serves as the access function with encryption support, the T-SBC supports transcoding in a tunneling gateway (TG) configuration to meet high-density transcoding requirements.

The following diagram shows an A-SBC positioned between an access UE and the IMS core. The A-SBC compares SDP offers and answers from the elements it sits between, and uses the results to determine whether or not a given session requires transcoding. When a session requires transcoding, the A-SBC invokes the services of the T-SBC. Acting as a B2BUA, the T-SBC uses information from the A-SBC's SIP message to transcode the applicable codecs and then to route SIP signaling back to the A-SBC on the egress.

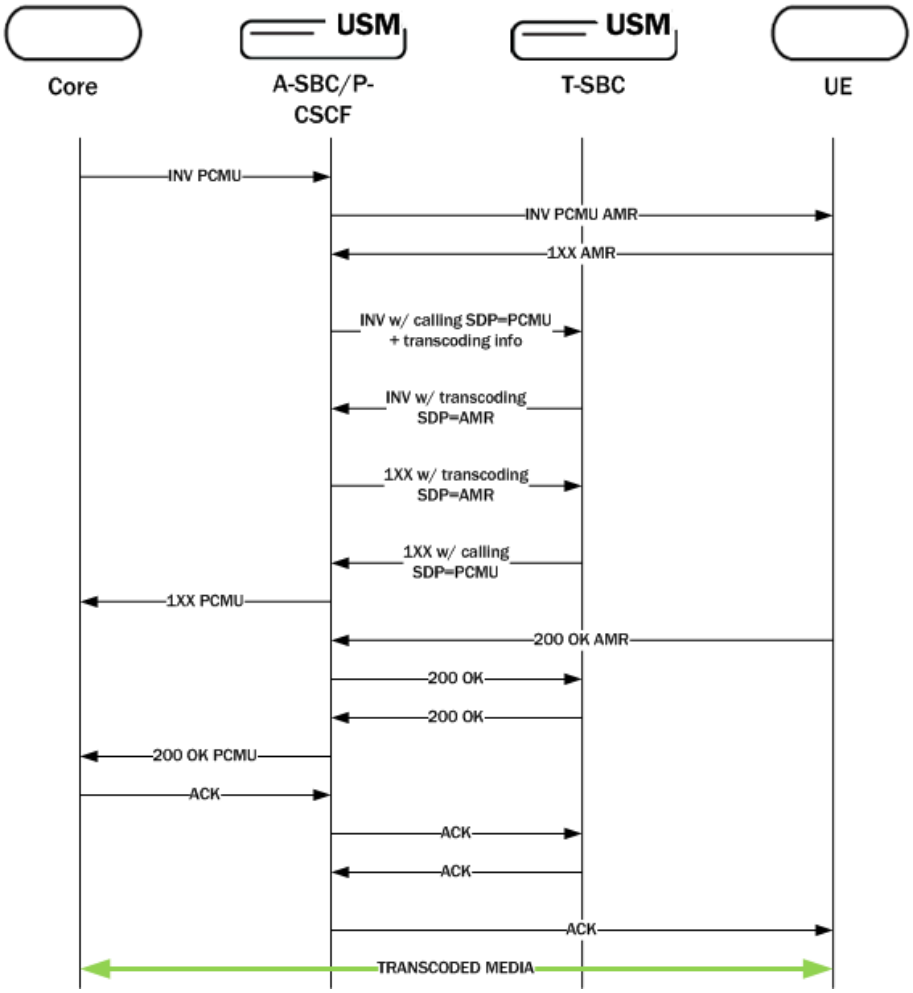
When a call comes in, the system temporarily creates two ports for the caller and callee realms respectively. When the call requires pooled transcoding, the SBC creates four ports on the pooled transcoding realm and two additional ports on the caller realm. The system removes the added ports at the end of the call.

The following example shows the ports before, during, and after for a call between the realms net172 (Offer) and net145 (Answer):

Before pooled transcoding	During pooled transcoding	After pooled transcoding
Net172 -> 2 Ports	Net172 -> 4 Ports	Net172 -> 2 Ports
Net145 -> 2 Ports	Net145 -> 2 Ports	Net145 -> 2 Ports
Net192 (Transcoding Realm) -> 0 Ports	Net192 (Transcoding Realm) -> 4 Ports	Net192 (Transcoding Realm) -> 0 Ports

In the preceding example, the SBC uses four ports on Net192 for communicating with the transcoding SBC. The SBC uses the two newly created ports on Net172 for RTP communication, while the original two ports are not used.

The following diagram shows what a call flow between entities in such a deployment model looks like:



Note:

The A-SBC and the T-SBC do not need to be on the same version of the hardware and the software.

Supported Codecs for Pooled Transcoding

The Oracle Communications Session Border Controller (SBC) supports the following codecs for pooled transcoding. Note that some platforms and software releases may not support all of the codecs in the list, and some codecs must be enabled.

- PCMU
- PCMA
- G722
- G723
- G726-16
- G726-24
- G726-32
- G726-40
- G728
- G729
- GSM
- AMR
- AMR-WB
- EVRC
- EVRC0
- EVRC1
- EVRCB
- EVRCB0
- EVRCB1
- EVS
- Opus
- SILK

The SBC supports the following other types of media for pooled transcoding.

- T.38
- T.140
- Baudot

Hardware and Software Requirements

Pooled transcoding deployments have specific hardware and software requirements. See the Coproduct Support section in the Release Notes for more information.

Implementation Details

To the Access SBC (A-SBC), the Transcoding SBC (T-SBC) is a transcoding agent that offers services that the A-SBC can invoke on-demand when an offer-answer exchange requires

transcoding. The A-SBC uses its public SIP interface and a corresponding realm reserved for communication with a T-SBC, which you configure as a transcoding agent on the A-SBC. You can configure multiple transcoding agents, which can be IP addresses, session agents, and session agent groups (SAG).

In a deployment with multiple transcoding agents, the A-SBC initiates communication in the order in which the transcoding agents were entered on the list. The A-SBC tries to communicate with each transcoding agent listed one-by-one until it:

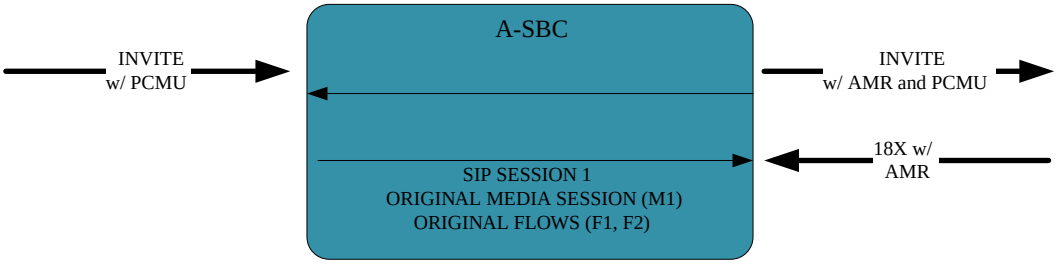
- Receives a 2xx response from a transcoding agent
- The list is exhausted
- The original transaction times out

When a transcoding agent is a session agent with hostnames, the A-SBC uses DNS to resolve the hostname and tries the hosts in order. When transcoding agents are session agent groups (SAG), the A-SBC selects the session agent according to the selection strategy configured for the SAG. When you enable recursing, the A-SBC recurses through all members of the SAG. When session agents and SAGs do not have ports or transport protocols specified, the defaults are 5060 and UDP respectively.

When the A-SBC identifies a transcoding agent, the A-SBC sends an INVITE to which the T-SBC responds with a 2xx message. Configuring the A-SBC as a session agent and disabling dialog transparency (in global SIP configuration) on the T-SBC allows the T-SBC to accept SIP messages from the A-SBC. Then the T-SBC acts as a B2BUA, using the information received in the INVITE from the A-SBC to invoke transcoding and to route SIP signaling back to the A-SBC on egress.

Scenario 1 INVITE with SDP

When the Access Session Border Controller (A-SBC) receives an INVITE with SDP, the A-SBC creates a SIP session and an associated media session with two flows for audio. The A-SBC applies the appropriate codec policy (with **add-on-egress** configured), so that the egress INVITE contains the necessary codec.



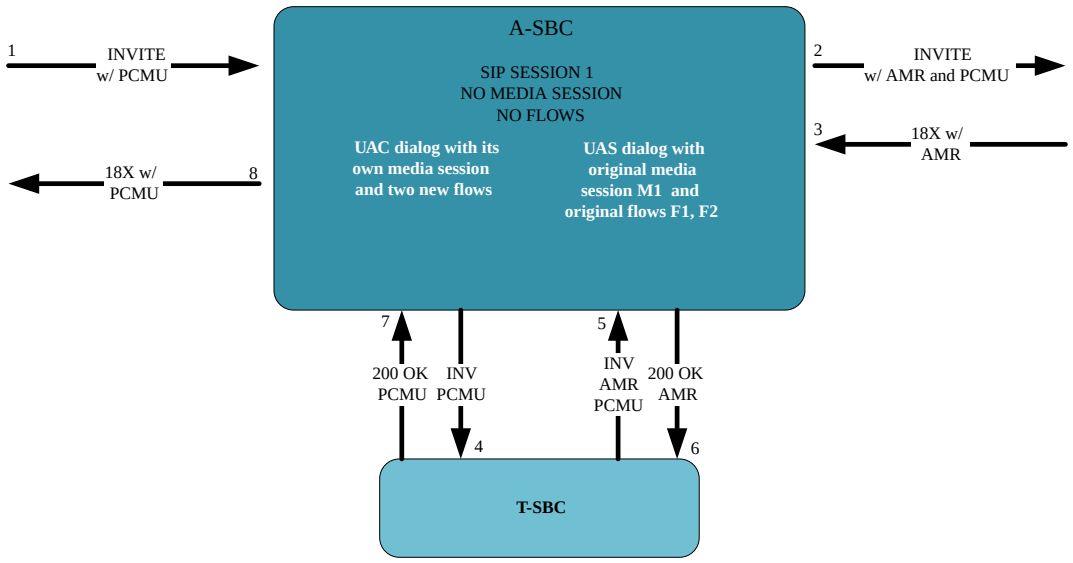
The A-SBC receives an answer to the INVITE, and when the answer contains the added codec the A-SBC invokes the Transcoding Session Border Controller (T-SBC) using an INVITE with the same SDP as the INVITE received on ingress. The communication between the A-SBC and the T-SBC is a separate dialog associated with a new media session.

The following code block shows and example of the INVITE the A-SBC sends to the T-SBC.

```
INVITE sip:192.168.101.78:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.18:5060;branch=z9hG4bK10dspa3058b1io4rk721
Max-Forwards: 70
Call-ID: 9fc71d79b43b4b95de41acefd71a8bc4@192.168.101.78
To: sip:192.168.101.78:5060
Contact: sip:172.16.101.18
From: sip:172.16.101.18;tag=bcf80756721880b7996ccb488b6a1b2f
```

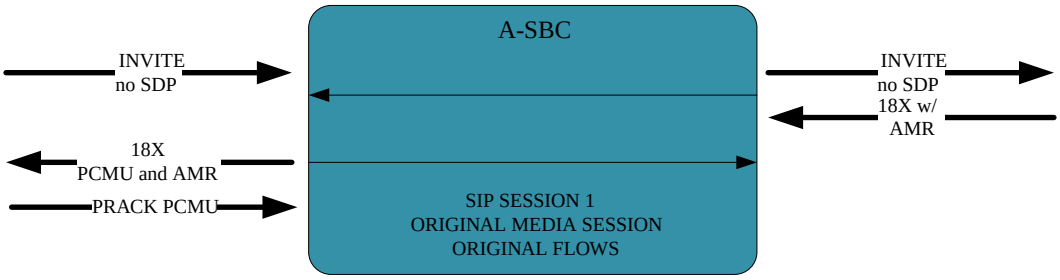
```
CSeq: 1 INVITE
Target-Dialog: 1-16881@172.16.18.5;local-tag=1;remote-
tag=16880SIPpTag011;realm=net192
Acme-Codec-Policy: ;ingress;name="in-2833";allow-codecs="* ";add-codecs-on-
egress="";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Acme-Codec-Policy: ;egress;name="out-2833";allow-codecs="* ";add-codecs-on-
egress="AMR ";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Content-Type: application/sdp
Content-Length: 196^M
P-Visited-Network-ID: open-ims.test
Route: <sip:192.168.101.18:5060;lr;transport=UDP>
v=0
o=user1 53655765 2353687637 IN IP4 192.168.101.18
s=-
c=IN IP4 192.168.101.18
t=0 0
m=audio 20002 RTP/AVP 96 0
a=rtpmap: 96 AMR/8000
a=rtpmap:0 PCMU/8000
```

Note that the Target Dialog and Acme-Codec-Policy headers communicate operational parameters for pooled transcoding. Using such information, the T-SBC applies two codec policies and returns the INVITE to the A-SBC. The A-SBC moves the media session to itself, but the SIP session does not have a media session at this time.



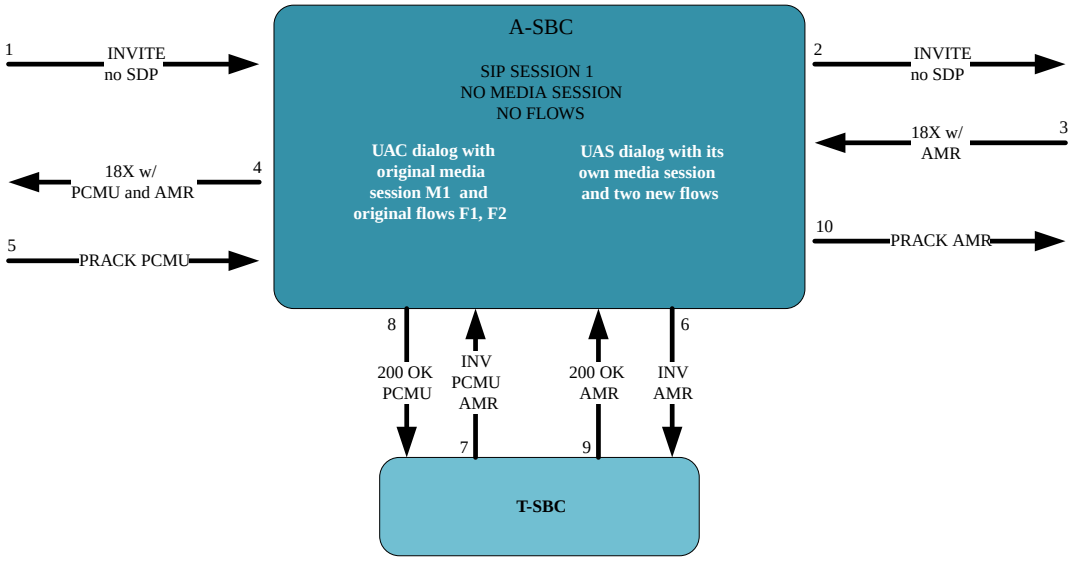
Scenario 2 INVITE without SDP

For an offerless call flow, the system creates a media session when the offer comes in a reliable provisional or final response. The Access Session Border Controller (A-SBC) applies the codec policy and sends the egress offer to the calling UE.



When the A-SBC receives an answer in either a PRACK or ACK message, the A-SBC compares the offer and answer. An answer containing the codec added in the egress offer causes the A-SBC to invoke the Transcoding Session Border Controller (T-SBC) and create a standalone UAC dialog, just as in Scenario 1.

Because the A-SBC advertised its media address in the egress offer to the calling UE, the UAC dialog uses the original media session.



Re-INVITES and Updates with SDP

When a specific session on the Access Session Border Controller (A-SBC) invokes the resources of the Transcoding Session Border Controller (T-SBC), the A-SBC continues to use the same T-SBC for the duration of the session. Anytime when the A-SBC receives a SIP message containing modified SDP, the A-SBC communicates the modification to the T-SBC.

RFC 2833 Considerations

For legacy RFC 2833 inter-working to function properly, the Access Session Border Controller (A-SBC) communicates the RFC 2833 configuration to the Transcoding Session Border Controller (T-SBC) in the UAC dialog.

The following example shows an INVITE with RFC 2833 information sent from the A-SBC to the T-SBC.

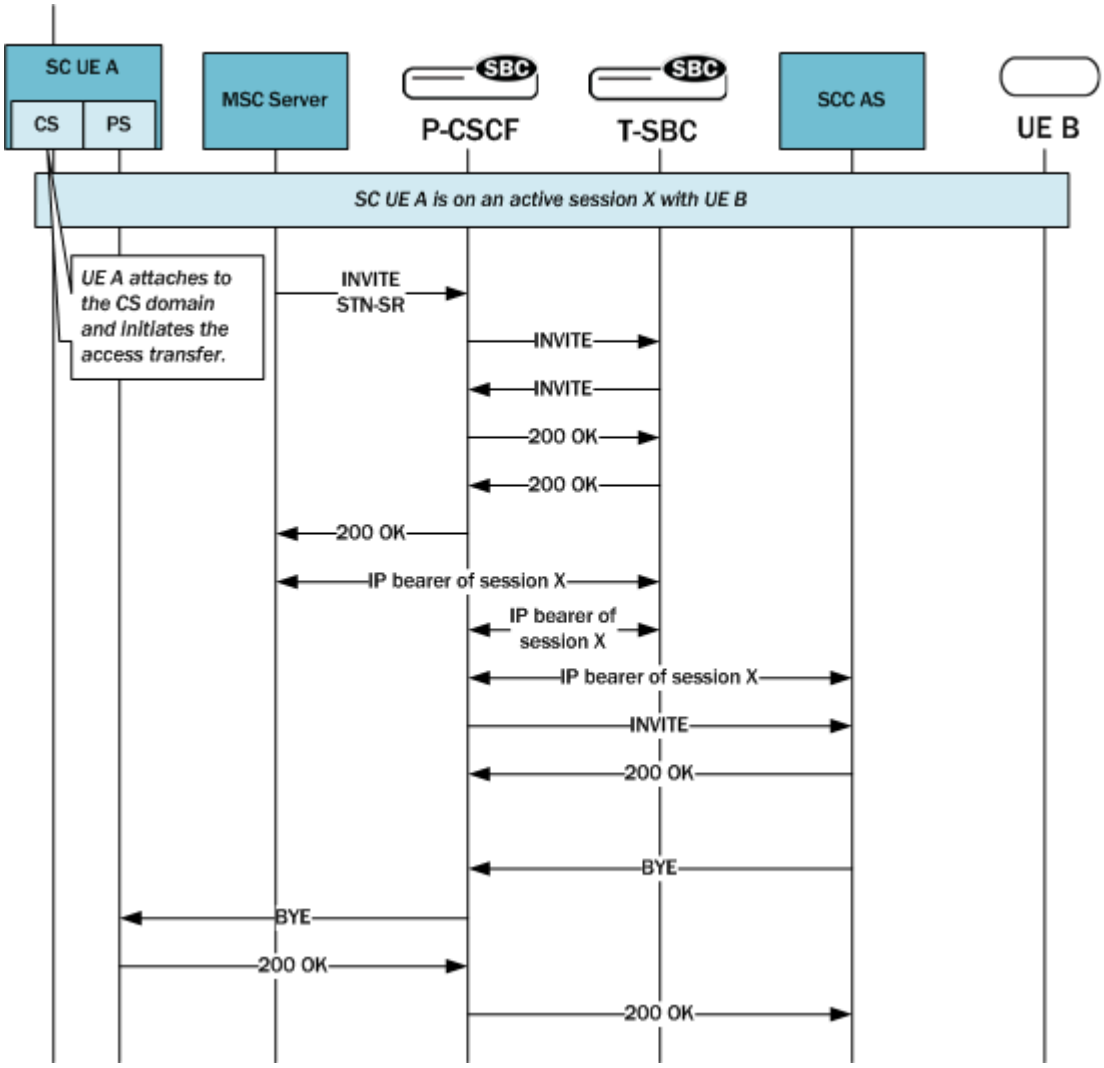
```
INVITE sip:192.168.101.78:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.18:5060;branch=z9hG4bK10dspa3058b1io4rk721
Max-Forwards: 70
Call-ID: 9fc71d79b43b4b95de41acefd71a8bc4@192.168.101.78
```

```
To: sip:192.168.101.78:5060
Contact: sip:172.16.101.18
From: sip:172.16.101.18;tag=bcf80756721880b7996ccb488b6a1b2f
CSeq: 1 INVITE
Target-Dialog: 1-16881@172.16.18.5;local-tag=1;remote-
tag=16880SIPpTag011;realm=net192
Acme-Codec-Policy: ;ingress;name="in-2833";allow-codecs="* ";add-codecs-on-
egress="";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Acme-Codec-Policy: ;egress;name="out-2833";allow-codecs="* ";add-codecs-on-
egress="PCMA ";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Acme-Rfc2833: ;ingress;Digit-
Mode=0;PtTo=101;PtFrom=0;TransNon2833=disabled;TransNonInband=disabled
Acme-Rfc2833: ;egress;Digit-Mode=1;PtTo=101;PtFrom=0
Content-Type: application/sdp
Content-Length: 196
P-Visited-Network-ID: open-ims.test
Route: <sip:192.168.101.18:5060;lr;transport=UDP>
v=0
o=user1 53655765 2353687637 IN IP4 192.168.101.18
s=-
c=IN IP4 192.168.101.18
t=0 0
m=audio 20002 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
```

eSRVCC Support

For enhanced Signal Radio Voice Call Continuity (eSRVCC), the pooled transcoding deployment model supports instances when:

- Original call is not transcoded, but the handover call is
- Original call is transcoded, but the handover call is not
- Original call is transcoded, and so is handover call



Configuration Requirements and Verification

Pooled transcoding requires specific configuration of the Access Session Border Controller (A-SBC) and the Transcoding Session Border Controller (T-SBC). Note that the Oracle Communications Session Border Controller, the A-SBC, does not enforce the requirements. Oracle recommends that you configure your pooled transcoding deployment with care, and verify the configuration to help to identify any potential issues.

A-SBC Configuration Requirements

The Access Session Border Controller (A-SBC) configuration must include the following:

- A public SIP interface the A-SBC can use for communication with the Transcoding Session Border Controller (T-SBC).
- A transcoding realm, which is a separate realm for the public SIP interface used only for communication with the T-SBC.
- Appropriate codec policies, including ones set up to add SDP on the egress leg of the session.

- A global SIP configuration, with **transcoding-realm** set to the name where valid transcoding agents reside, and a **transcoding-agents** list configured with one or any combination of
  - A DNS hostname for a single session agent that can resolve to one or more IP addresses
  - An IPv4 or IPv6 address with or without the port specified (when not specified, the system defaults to port 5060).
  - The name of a session agent group.

T-SBC Requirements

A valid Transcoding Session Border Controller (T-SBC) configuration must include the following:

- A global SIP configuration, with **dialog-transparency** set to **disabled**. You must disable Dialog transparency on the T-SBC, so that the INVITE from the T-SBC contains a different call ID and From tag.
- A public realm with **codec-manip-in-realm** set to **enabled** because the system uses the same realm for transcoding. You must also enable the **mm-in-realm** parameter.

Configuration Verification

You can verify the configuration by using the ACLI **verify-config** command. When verifying the configuration on the Access Session Border Controller (A-SBC), the system displays errors messages when:

- The **transcoding-realm** value configured is not a valid realm.
- Either one of the **transcoding-realm** or **transcoding-agents** parameters is not configured.
- One or more session agent names defined in the **transcoding-agents** list is not a valid session agent.
- The IP address version for a agent in the **transcoding-agents** list is not supported in the transcoding realm you identify. For example, if you list an IPv6 agent in the list and the transcoding realm does not support IPv6.
- The transcoding agent is a hostname and not a valid session agent.

Configure Pooled Transcoding

You must configure a transcoding realm and transcoding agents on your Access Session Border Controller (A-SBC), when used in a pooled transcoding deployment model. Set the parameters as part of the global SIP configuration.

1. Access the **sip-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. If you are adding transcoding support to a pre-existing SIP configuration, you must select the configuration before you can make changes.

3. Enter the name of a configured realm designated as the separate realm for the public SIP interface for exclusive communication with the Transcoding Session Border Controller (T-SBC) in as pooled transcoding deployment.

```
ORACLE(sip-config)# transcoding-realm net157
ORACLE(sip-config)#
```

4. **transcoding-agents**—Enter any IP address, IP address and port combination, session agent hostname, or SAG name to use as a transcoding agent. You can make multiple entries in any combination of these values. For example, you might list an IPv6 address and port, a session agent, and a SAG.
   - To make multiple entries in the list using in one command line, enclose the entire list of value in parentheses (( )), separating each with a Space as in the example below.
   - To add a transcoding agent to an existing list, put a plus sign before the value you want to add, e.g. +154.124.2.8.
   - To remove a transcoding agent from an existing list, put a minus sign before the value you want to remove, e.g. -154.124.2.8.

```
ORACLE(sip-config)# transcoding-agent (sag:sag1 192.168.4.7.3
192.168.2.6:5061)
ORACLE(sip-config)#
```

5. Type **done** to save your configuration.

Monitor Dialogs Between the A-SBC and the T-SBC

You can monitor pooled transcoding communications between the Access Session Border Controller (A-SBC) and the Transcoding Session Border Controller (T-SBC) by using the **show sipd pooled-transcoding** ACLI command. The display shows you information for the client and server User Agents on the A-SBC.

- UAC Dialogs—shows the number of UAC dialogs the A-SBC initiated with the T-SBC. The count is cumulative, and not for any specific session.
- UAS Dialogs—shows the number of UAS dialogs the A-SBC created upon receipt of an INVITE from the T-SBC. The count is cumulative, and not for any specific session.
- XCodeSessions—counts the number of transcoded sessions on the A-SBC.

```
ACMEPACKET# show sipd pooled-transcoding
18:11:28-125
SIP Pooled Transcoding Status
                    -- Period --                 -------- Lifetime --------
                Active    High   Total      Total  PerMax    High
UAC Dialogs            1       1       1          1       1       1
  Early                0       0       0          0       0       0
  Confirmed            1       1       1          1       1       1
  Terminated           0       0       0          0       0       0
UAS Dialogs            1       1       1          1       1       1
  Early                0       0       0          0       0       0
  Confirmed            1       1       1          1       1       1
  Terminated           0       0       0          0       0       0
XCodeSessions          1       1       1          1       1       1
```

Per-Method Statistics

When you set **extra-method-stats** to **disabled** in the global SIP configuration, the system displays the per-method statistics for the Access Session Border Controller (A-SBC) public transcoding-realm.

```
ACMEPACKET# show sipd realms net157 invite
INVITE (18:14:59-36)
                    --------- Server --------   --------- Client --------
Message/Event       Recent      Total  PerMax   Recent      Total  PerMax
                    ------  ---------  ------   ------  ---------  ------
INVITE Requests          0          1       1        0          1       1
Retransmissions          0          0       0        0          0       0
100 Trying               0          1       1        0          1       1
200 OK                   0          1       1        0          1       1
Response Retrans         0          0       0        0          0       0
Transaction Timeouts     -          -       -        0          0       0
Locally Throttled        -          -       -        0          0       0

Avg Latency=0.000 for 0
Max Latency=0.000
BurstRate Incoming=1 Outgoing=0

ACMEPACKET# show sipd realms net157 bye
BYE (18:15:13-50)
                    --------- Server --------   --------- Client --------
Message/Event       Recent      Total  PerMax   Recent      Total  PerMax
                    ------  ---------  ------   ------  ---------  ------
BYE Requests             0          1       1        0          1       1
Retransmissions          0          0       0        0          0       0
200 OK                   0          1       1        0          1       1
Transaction Timeouts     -          -       -        0          0       0
Locally Throttled        -          -       -        0          0       0

Avg Latency=0.000 for 0
Max Latency=0.000
BurstRate Incoming=1 Outgoing=0
```

Pooled Transcoding and MS Teams

The SBC supports pooled transcoding for most call flows, including incoming calls, outgoing calls, call forwarding and call transfers within MS Teams environments.

To support pooled transcoding for MS Teams environments, you augment the SBC elements and the environment described previously with the following configurations on both the A-SBC and T-SBC devices with the following parameters in the realms connecting each SBC:

- **codec-policy**—Create a policy with a **name** and default parameters on the T-SBC and apply it to the realm that connects to the A-SBC. The T-SBC needs a **codec-policy** to support the **rtcp-policy**.
  You may also define a codec policy on the A-SBC based on your transcoding requirements.
- **rtcp-policy**—Create a policy by configuring a **name** and setting the **rtcp-generate** parameter to **all-calls**. Apply this policy on the A-SBC on the realm facing your MS Teams infrastructure. Configure an equivalent policy on your T-SBC and apply it to the realm that connects to the A-SBC.
- **rtcp-mux** —Enable this parameter on the realms of both devices that connect the A-SBC and the T-SBC.

> **Note:**
>
> Standard MS Teams support also requires that you enable **rtcp-mux** on the realm facing the MS Teams environment.

Configure Pooled Transcoding for MS Teams

If you are using Pooled Transcoding within MS Teams environments, you must perform the following configuration steps in addition to the generic pooled transcoding configurations. You perform equivalent procedures below on both the A-SBC and the T-SBC. You create two policies and enable an additional parameter on the realms as listed in the procedure.

Perform the procedures required to configure all pooled transcoding deployments.

1. Access the **rtcp-policy** configuration element from within the **media-manager**. Create this **rtcp-policy** on both the A-SBC and the T-SBC.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# rtcp-policy
ORACLE(rtcp-policy)#
```

> **Note:**
>
> After creating these policies, you will apply them to the realms stated in the steps below.

2. Enter a name for your **rtcp-policy**.

```
ORACLE(rtcp-policy)#name MyRtcpPolicy
```

3. Configure RTCP generation to **all-calls** within this policy.

```
ORACLE(rtcp-policy)#rtcp-generate all-calls
```

4. Type done to save your **rtcp-policy**.

```
ORACLE(rtcp-policy)#done
rtcp-policy
        name                                    MyRtcpPolicy
        rtcp-generate                           all-calls
        hide-cname                              disabled

ORACLE(rtcp-policy)#
```

   You create this **rtcp-policy** on both the A-SBC and the T-SBC.

5. Exit the **rtcp-policy**.

6. Access the **codec-policy** configuration element from within the **media-manager**.

```
ORACLE(media-manager)# codec-policy
ORACLE(codec-policy)#
```

You create a **codec-policy** on both the A-SBC and the T-SBC.

7. Enter a name for your **codec-policy**.

Although the **codec-policy** on your T-SBC only requires a name, you configure the **codec-policy** on your A-SBC as needed to support your requirements for transcoding.

8. Type done to save your **codec-policy**.

```
ORACLE(codec-policy)#done
codec-policy
        name                                    MyCodecPolicy
        allow-codecs
        add-codecs-on-egress
        order-codecs
        packetization-time                      20
        force-ptime                             disabled
        secure-dtmf-cancellation                disabled
        dtmf-in-audio                           disabled
        tone-detect-renegotiate-timer           500
        reverse-fax-tone-detection-reinvite     disabled
        evrc-tty-baudot-transcode               disabled

ORACLE(codec-policy)#
```

9. Exit the **codec-policy**.

The remaining steps in this procedure specify configuration differences between the A-SBC and the T-SBC

10. On the T-SBC:
    a. Apply your **rtcp-policy** to the realm that connects to the A-SBC.
    b. Apply your **codec-policy** on the realm that connects to the A-SBC.
    c. Enable **rtcp-mux** on the realm that connects the A-SBC and the T-SBC.
11. On the A-SBC:
    a. Apply your **rtcp-policy** to the realm that faces your MS-Teams infrastructure.
    b. Enable **rtcp-mux** on the realm that connects the A-SBC and the T-SBC.

       Standard MS Teams support also requires that you enable **rtcp-mux** on the realm facing the MS Teams environment.

    You determine whether you need a **codec-policy** on your A-SBC, based on your transcoding needs on that device. This policy, however, would not be related to this feature.
12. Type done, exit from configuration mode, then Save and Activate your configurations.

Notes on the DIAMETER Rx Interface

DIAMETER Rx support for external bandwidth management is specialized for pooled transcoding.

For pooled transcoding, the identifier appears in this altered format: <policy server name>;<policy server realm>;<dns suffix>. Because there are two dialogs (one for the server UA and one for the client UA), there are two separate media sessions. This situation requires the Oracle Communications Session Border Controller to generate two different DIAMETER session identifiers. At the same time, the Oracle Communications Session Border Controller needs to maintain the AAR and STR associations with the original SIP session. To keep this relationship clear, the Oracle Communications Session Border Controller keeps the DIAMETER session identifier between the two media sessions the same for the two UAs.

The MBCD session identifier associated with the media session for the original SIP session is retained for the DIAMETER session identifier for the duration of the SIP session.

Accounting and Transcoding

An Access Control Record (ACR) record for a typical SIP session looks the same as one for a transcoded session, except that the forward codec (Acme-FlowType_FS1__F) differs from the reverse codec (Acme-FlowType_FS1__R), due to transcoding.

For the RADIUS and Diameter protocols, the ACR record shows only the IP address and port number of the two original endpoints. The record shows no details about the Transcoding Session Border Controller (T-SBC) because the system does not support transcoding for RADIUS and Diameter.

EVRC Family of Codecs

The Acme Packet 6300 supports 2 codecs in the EVRC family. Each codec has has three variants.

- EVRC-A is also commonly referred to as EVRC. The following EVRC-A packet formats are supported:

Header-free packet format = EVRC0

Bundled/interleaved packet format = EVRC

Compact Bundled packet format = EVRC1

- The following EVRC-B packet formats are supported:

Header-free packet format = EVRCB0

Bundled/interleaved packet format = EVRCB

Compact Bundled packet format = EVRCB1

EVRC-A Codec for Transcoding

The Acme Packet 6300 supports the Enhanced Variable Rate codec (EVRC) for transcoding. Three subtypes of the EVRC codec are supported as media-profiles:

- EVRC0—header-free EVRC format. This codec is transcodable.
- EVRC—non-header-free EVRC format. This codec is transcodable.
- EVRC1—fixed rate EVRC format. This codec is transcodable.

> **Note:**
>
> The Acme Packet 6300 handles optional parameters according to RFC 4788 unless otherwise specified.

EVRC0 Supported Options

Required Parameters: none

Optional Parameters:

- silencesupp—If this parameter is not present in a DTX session, the default value 1 MUST be assumed.
- dtxmax—If this parameter is not present in a DTX session, the default value is 32.
- dtxmin—If this parameter is not present in a DTX session, the default value is 12.
- hangover—If this parameter is not present in a DTX session, the default value 1 MUST be assumed.

The payload type is dynamic for this codec. EVRC0 specifies the header-free format of the EVRC codec. Only a single frame (20 ms) is allowed in header-free mode.

EVRC Supported Options

Required Parameters: none

Optional parameters:

- ptime
- maxptime
- maxinterleave—If not signaled, the maxinterleave length is set to 5.
- silencesupp—If this parameter is not present, the default value 1 MUST be assumed.
- dtxmax—See dtxmax in EVRC0 section.
- dtxmin—See dtxmin in EVRC0 section.
- hangover—See hangover in EVRC0 section.

The payload type is dynamic for this codec. The default ptime is 20 ms. The ptimes 20, 40, and 60 ms are supported for transcoding.

EVRC1 Supported Options

Required parameters: none

Optional parameters:

- ptime—See RFC 4566
- maxptime
- fixedrate—Valid values are 0.5 or 1. If this parameter is not present, 0.5 is assumed.
- silencesupp—If this parameter is not present, 1 MUST be assumed.
- dtxmax—See dtxmax in EVRC0 section.

- dtxmin—See dtxmin in EVRC0 section.
- hangover—See hangover in EVRC0 section.

The payload type is dynamic for this codec. Only the 20 ms ptime is supported for transcoding. The media rate will be fixed to either full or half rate depending on the fixedrate parameter (half rate is default).

Default settings for EVRC encoding

- CDMA Rate change for Dim and Burst disabled
- CDMA DTX control enabled

EVRC Configuration

In the ACLI, refer to EVRC codecs exactly as follows:

```
EVRC0
EVRC
EVRC1
```

EVRC-B Codec for Transcoding

The Acme Packet 6300 supports the Enhanced Variable Rate codec, extension B (EVRCB) for transcoding. Three subtypes of the EVRCB codec are supported as media-profiles:

- EVRCB0—This codec is transcodable.
- EVRCB—This codec is transcodable.
- EVRCB1—This codec is transcodable.

> **Note:**
>
> The Acme Packet 6300 handles optional parameters according to RFC 4788 unless otherwise specified.

EVRCB0 Supported Options

EVRBC0 is the header-free format of the EVRCB codec. Only a single frame (20 ms) is allowed in header-free mode.

Required Parameters: none

Optional Parameters:

- silencesupp — If this parameter is not present in a DTX session, the default value 1 MUST be assumed.
- dtxmax — If this parameter is not present in a DTX session, the default value is 32.
- dtxmin — If this parameter is not present in a DTX session, the default value is 12.
- hangover — If this parameter is not present in a DTX session, the default value 1 MUST be assumed.

The payload type is dynamic for this codec. EVRCB0 specifies the header-free format of the EVRCB codec. Only a single frame (20 ms) is allowed in header-free mode.

EVRCB Supported Options

Required Parameters: none

Optional parameters:

- ptime
- maxptime
- maxinterleave—If not signaled, the maxinterleave length is set to 5.
- silencesupp—If this parameter is not present, the default value 1 MUST be assumed.
- dtxmax
- dtxmin
- hangover

The payload type is dynamic for this codec. The default ptime is 20 ms. Only the 20 ms ptime is supported for transcoding.

EVRCB1 Supported Options

Required parameters: none

Optional parameters:

- ptime—See RFC 4566
- maxptime
- fixedrate—Valid values are 0.5 or 1. If this parameter is not present, 0.5 is assumed.
- silencesupp—If this parameter is not present, 1 MUST be assumed.
- dtxmax
- dtxmin
- hangover

The payload type is dynamic for this codec. Only the 20 ms ptime is supported for transcoding. The media rate will be fixed to either full or half rate depending on the fixedrate parameter (half rate is default).

Default fixed settings for EVRCB encoding

- CDMA Rate change for Dim and Burst disabled
- CDMA DTX control enabled

EVRCB Configuration

In the ACLI, refer to EVRCB codecs exactly as follows:

```
EVRCB0
EVRCB
EVRCB1
```

Opus Codec Transcoding Support

Opus is an audio codec developed by the IETF that supports constant and variable bitrate encoding from 6 kbit/s to 510 kbit/s and sampling rates from 8 kHz (with 4 kHz bandwidth) to 48 kHz (with 20 kHz bandwidth, where the entire hearing range of the human auditory system can be reproduced). It incorporates technology from both Skype's speech-oriented SILK codec and Xiph.Org's low-latency CELT codec. This feature adds the Opus codec as well as support for transrating, transcoding, and pooled transcoding.

Opus can be adjusted seamlessly between high and low bit rates, and transitions internally between linear predictive coding at lower bit rates and transform coding at higher bit rates (as well as a hybrid for a short overlap). Opus has a very low algorithmic delay (26.5 ms by default), which is a necessity for use as part of a low audio latency communication link, which can permit natural conversation, networked music performances, or lip sync at live events. Opus permits trading-off quality or bit rate to achieve an even smaller algorithmic delay, down to 5 ms. Its delay is very low compared to well over 100 ms for popular music formats such as MP3, Ogg Vorbis, and HE-AAC; yet Opus performs very competitively with these formats in terms of quality across bit rates.

Opus Supported Options

Required SDP Parameters:

rate — Specifies the sampling frequency. This parameter is mapped to the RTP clock rate in "a=rtpmap". The range is limited to and must be 48000 Hz.

Optional SDP Parameters:

- **maxplaybackrate** — Specifies the maximum output sampling rate in Hz that the receiver is capable of rendering. The range is 8 kHz to 48 kHz; common values are 8, 12, 16, 24, and 48 kHz.
- **sprop-maxcapturerate** — Specifies the maximum input sampling rate in Hz that the sender is likely to produce. The Vocallo OCT2224 DSP currently supports only 8000 and 16000 Hz for transcoding. The range is 8 kHz to 48 kHz; common values are 8, 12, 16, 24, and 48 kHz.
- **ptime** — Specifies the packetization interval in milliseconds. The DSP supports packetization intervals of 10, 20, 40, 60, 80, and 100 ms. This parameter is mapped to "a=ptime" in the SDP. Possible values are 3, 5, 10, 20, 40, 60, or an arbitrary multiple of Opus frame sizes rounded up to the next full integer value up to a maximum value of 120. The default is 20 ms.
- **maxptime** — Specifies the maximum packetization interval allowed. The default is 100 ms.
- **minptime** — Specifies the minimum packetization interval allowed. The default is 20 ms.
- **maxaveragebitrate** — Specifies the maximum average rate of bits received for a session in bits per second. Although the range is 6000 to 51000, only bit rates of 6000 to 30000 bps are transcodable by the DSP. A media profile configured with a value for **maxaveragebitrate** greater than 30000 is not transcodable and cannot be added on egress in the **codec-policy** element.
- **stereo** — Specifies whether the decoder receives stereo or mono signals. The possible values are 0 (mono) and 1 (stereo). The default is 0.
- **sprop-stereo** — Specifies whether the sender is likely to produce stereo audio. The possible values are 0 (mono) and 1 (stereo). The default is 0.

- **cbr** — Specifies whether the decoder uses a constant or a variable bit rate. The possible values are 0 (variable bit rate) and 1 (constant bit rate). The default is 0.
- **useinbandfec** — Specifies whether the Opus decoder supports Forward Error Correction (FEC). The possible values are 0 (no) and 1 (yes). The default is 1.
- **usedtx** — Specifies whether the Opus decoder utilizes Discontinuous Transmission (DTX). The possible values are 0 (no) and 1 (yes). The default is 0.

The payload type is dynamic for this codec.

Sample media-profile configuration for adding Opus

Parameter	Value
name	opus
subname	WB
media-type	audio
payload-type	104
transport	RTP/AVP
clock-rate	48000
req-bandwidth	0
frames-per-packet	0
parameters	maxplaybackrate=16000 sprop-maxcapturerate=16000 usedtx=0
average-rate-limit	5000
peak-rate-limit	0
max-burst-size	0
sdp-rate-limit-headroom	0
sdp-bandwidth	enabled
police-rate	0
standard-pkt-rate	0

Monitoring and Debugging

CLI commands:

The **show sipd codecs** command is modified to add **opus Count**.

SNMP:

- New SNMP OID **apSysXCodeOPUSCapacity** is added to transcoding utilization statistics as reported in the **apSysMgmtGroupTrap**. When utilization falls below 80%, the **apSysMgmtGroupClearTrap** is sent.
- Opus realm statistic **apCodecRealmCountOPUS** is added to **apCodecRealmStatsEntry**.

Alarms:

Opus Transcoding Capacity Threshold Alarm — A warning level alarm that doesn't affect health is triggered when the Opus transcoding utilization exceeds 95% of capacity. The alarm is cleared when the Opus transcoding utilization falls below 80% of capacity.

SILK Codec Transcoding Support

SILK is an audio codec developed by Skype Limited that supports bit rates from 6 kbit/s to 40 kbit/s and sampling rates of 8, 12, 16, or 24 kHz. It can also use a low algorithmic delay of 25 ms (20 ms frame size + 5 ms look-ahead). This feature adds the SILK codec as well as support for transrating, transcoding, and pooled transcoding.

SILK Supported Options

Required SDP Parameters:

rate — Specifies the sampling frequency. SILK supports four different audio bandwidths – narrowband at 8 kHz, mediumband at 12 kHz, wideband at 16 kHz, and super wideband at 24 kHz. This parameter is mapped to the RTP clock rate in “a=rtpmap”. The DSP only supports audio sampling rates of 8 kHz and 16 kHz for transcoding; the 12 kHz and 24 kHz bandwidths are not transcodable.

Optional SDP Parameters:

- **ptime** — Specifies the packetization interval in milliseconds. The DSP supports packetization intervals of 20, 40, 60, 80, and 100 ms. This parameter is mapped to “a=ptime” in the SDP. The default is 20 ms.
- **maxptime** — Specifies the maximum packetization interval in milliseconds. The default is 100 ms.
- **minptime** — Specifies the minimum packetization interval in milliseconds. The default is 20 ms.
- **maxaveragebitrate** — Specifies the maximum average rate of bits received for a session in bits per second. Bit rates of 5000 to 30000 bps are transcodable by the DSP.
- **usedtx** — Specifies whether the SILK decoder utilizes Discontinuous Transmission (DTX). The possible values are 0 (no) and 1 (yes). The default is 0.

The payload type is dynamic for this codec.

Sample media-profile configuration for adding SILK

Parameter	Value
name	SILK
subname	wideband
media-type	audio
payload-type	103
transport	RTP/AVP
clock-rate	16000
req-bandwidth	0
frames-per-packet	0
parameters	N/A
average-rate-limit	5000
peak-rate-limit	0
max-burst-size	0
sdp-rate-limit-headroom	0
sdp-bandwidth	enabled
police-rate	0

Parameter	Value
standard-pkt-rate	0

Monitoring and Debugging

CLI commands:

The **show sipd codecs** command is modified to add **SILK Count**.

SNMP:

- New SNMP OID **apSysXCodeSILKWBCapacity** is added to transcoding utilization statistics as reported in the **apSysMgmtGroupTrap**. When utilization falls below 80%, the **apSysMgmtGroupClearTrap** is sent.
- SILK realm statistic **apCodecRealmCountSILK** is added to the **apCodecRealmStatsEntry** table located in the `ap-tc.mib`.

Alarms:

SILK Transcoding Capacity Threshold Alarm — A warning level alarm that doesn't affect health is triggered when the SILK transcoding utilization exceeds 95% of capacity. The alarm is cleared when the SILK transcoding utilization falls below 80% of capacity.

Comfort Noise Transcoding

The Comfort Noise (CN) codec provides a means of encoding periods of silence that occur in an audio stream into a low-level sound that confirms to the listener that the call remains active. In a scenario where the endpoint does not support CN packets during periods of silence, the listener might think that the phone call disconnected. To avoid such a scenario, you can enable the Oracle Communications Session Border Controller (SBC) to transcode the CN packet into in-band RTP packets of the voice codec resulting in low-level sound during silences in the audio stream.

Without CN transcoding enabled, the SBC forwards all of the CN packets through to the endpoint on ingress and back out again on egress. Also, without CN transcoding enabled, the SBC passes through all of the thousands of in-band RTP packets that make up the silences when the endpoint does not support CN. Passing thousands of RTP packets can use a large amount of bandwidth. To save bandwidth in scenarios where one endpoint supports CN and the other endpoint does not, you can set the codec policy to transcode a CN packet into in-band audio RTP packets containing silence. Such transcoding results in a lower transmission rate of RTP packets because the system sends only one CN packet on egress rather than the thousands of RTP packets that would otherwise pass through between ingress and egress.

To configure the SBC to transcode the CN codec, you must add "CN" to the **add-codecs-on-egress** list in the codec-policy configuration.

For transcoding comfort noise:

- The side of the call that does not support CN, must use an audio codec that the SBC supports for transcoding. See "Transcodable Codecs."
- The side of the call that supports CN, must use an audio codec that the SBC supports as interoperable with CN. Interoperable codecs: (non-signalling codecs) PCMA, PCMU, and G.726.

Be aware that enabling CN transcoding can generate periods when no RTP packets flow on the side of the call that sends and receives CN packets. Depending on the values set in the following guard timer parameters, the system might detect no flow and drop the call.

- flow-time-limit
- initial-guard-timer
- subsq-guard-timer
- tcp-flow-time-limit
- tcp-initial-guard-timer
- tcp-subsq-guard-timer

Setting the value for a guard timer parameter to zero disables the parameter. When you set the parameters to zero, the system does not terminate any calls due to lack of RTP packets. Set the guard timers in the **media-manager-config** object.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

Supported and Unsupported Platforms and Behavior

Comfort Noise transcoding supports:

- Any platform that supports transcoding.
- simultaneous transcoding of comfort noise and telephone event (RFC 2833) to and from in-band DTMF.
- using either the comfort-noise-generate SPL or comfort noise transcoding. The system cannot support both methods at the same time.
- High Availability (HA) as currently implemented, regarding the configuration and media flows.

Troubleshooting

- Inspect the SDP m= and a= lines for correct modifications as a result of the codec policy.
- When you enable SIP debug, the sipmsg.log shows "CN-XCODE-GEN-Enabled" for the flow that you enabled to generate CN.

System Behavior Without Comfort Noise Transcoding

The following scenarios describe how the Oracle Communications Session Border Controller (SBC) handles Comfort Noise (CN) packets without transcoding enabled.

Both sides offer and answer support for CN

CN packets pass through the SBC from the Offerer to the Answerer and from the Answerer to the Offerer.

Neither side offers or answers support for CN

Any silence in the audio stream passes through the SBC in the RTP audio packets.

One side supports CN, and the other side does not

The Offerer does not send CN packets because the Answerer advertises no support for CN. Any silence in the audio stream passes through the SBC in the RTP audio packets.

System Behavior With Comfort Noise Transcoding

The following scenarios show how the Oracle Communications Session Border Controller (SBC) handles Comfort Noise (CN) packets with transcoding enabled.

General Behavior in the Scenarios

When the SDP produced by the ingress codec-policy O[1] contains an audio codec that interoperates with CN (PCMA, PCMU, and G.726), and the egress codec-policy allows the interoperable codec and is configured to add CN, the SBC adds the default payload type of 13 to the SDP m= line.

When the SDP produced by the ingress codec-policy O[1] does not contain a codec that interoperates with CN, and the egress codec-policy is configured to add an interoperable audio codec plus the CN codec, the SBC adds the default payload type of 13 to the SDP m= line.

When the SBC adds the CN codec, the system adds the following default a= line to the SDP:

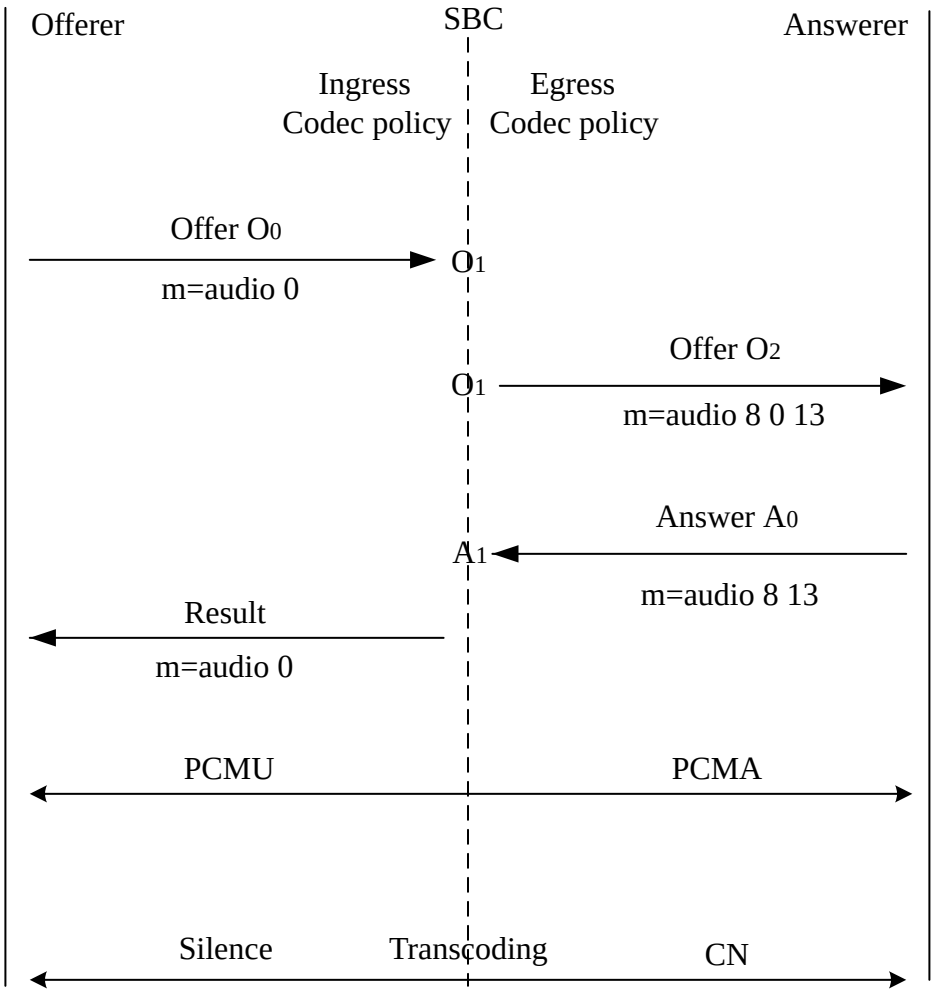
a=rtpmap:13 CN/8000

For more information about the concepts and terms used in the scenarios, see "Transcoding Processing Overview."

Comfort Noise Transcoded Scenario

In a typical scenario with CN transcoding enabled, the SBC transcodes the ingress audio codec for the media stream to PCMA. The DSP detects silence in the media stream from the Offerer, translates the silence into a CN packet, and sends the packet to the Answerer. The Answerer sends a CN packet to the DSP, which translates the packet into silence in the media stream and sends the packet to the Offerer. The result is silence on egress.

Ingress codec policy	Setting	Egress codec policy	Setting
allow codecs	*	allow codecs	*
add codecs on egress	N/A	add codecs on egress	PCMA CN
order codecs	N/A	order codecs	N/A



The side of the call that does not support CN, must use an audio codec that the SBC supports as transcodable, represented by "x" in the call flow.

The side of the call that supports CN, must use an audio codec that the SBC supports as interoperable with CN, represented by PCMA in the call flow.

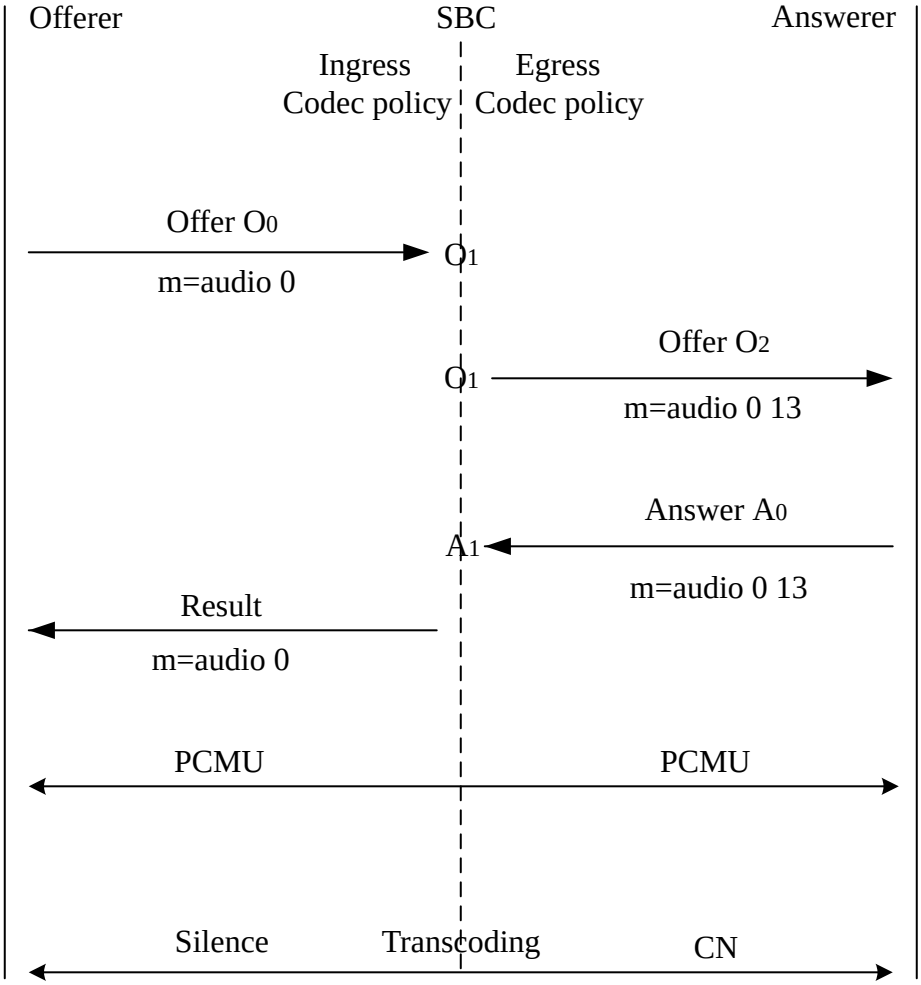
Audio Codec Not Transcoded - Comfort Noise Codec Transcoded Scenario

In the following scenario, the ingress codec policy and the egress codec policy both allow all offered codecs. The setting for add-codecs-on-egress is CN.

1. The Offerer sends an offer O° to the SBC with an SDP m-line for an audio codec PCMU, but with no CN signaling codec
2. The codec-policy for the ingress realm allows the audio codec, PCMU, and outputs O^1 which contains PCMU.
3. The SBC egress codec-policy takes O^1 as input and adds CN after checking that at least one of the audio codecs in O^1 interoperates with CN, resulting in PCMU and CN in output O^2.
4. The Answerer responds with the answer in A^o containing the PCMU audio codec and the signaling codec CN, indicating the Answerer supports comfort noise. The egress codec-policy produces answer A^1 with no changes.
5. The SBC removes CN from A^1 because it was not offered by the Offerer. The result contains only the audio codec PCMU.

Even with CN transcoding enabled, the SBC does not transcode the audio codec (PCMU). The SBC transcodes silence detection and generation, as well as CN packet detection and generation.

Ingress codec policy	Setting	Egress codec policy	Setting
allow codecs	*	allow codecs	*
add codecs on egress	N/A	add codecs on egress	CN
order codecs	N/A	order codecs	N/A



Comfort Noise and Telephone Event Codecs Both Transcoded Scenario

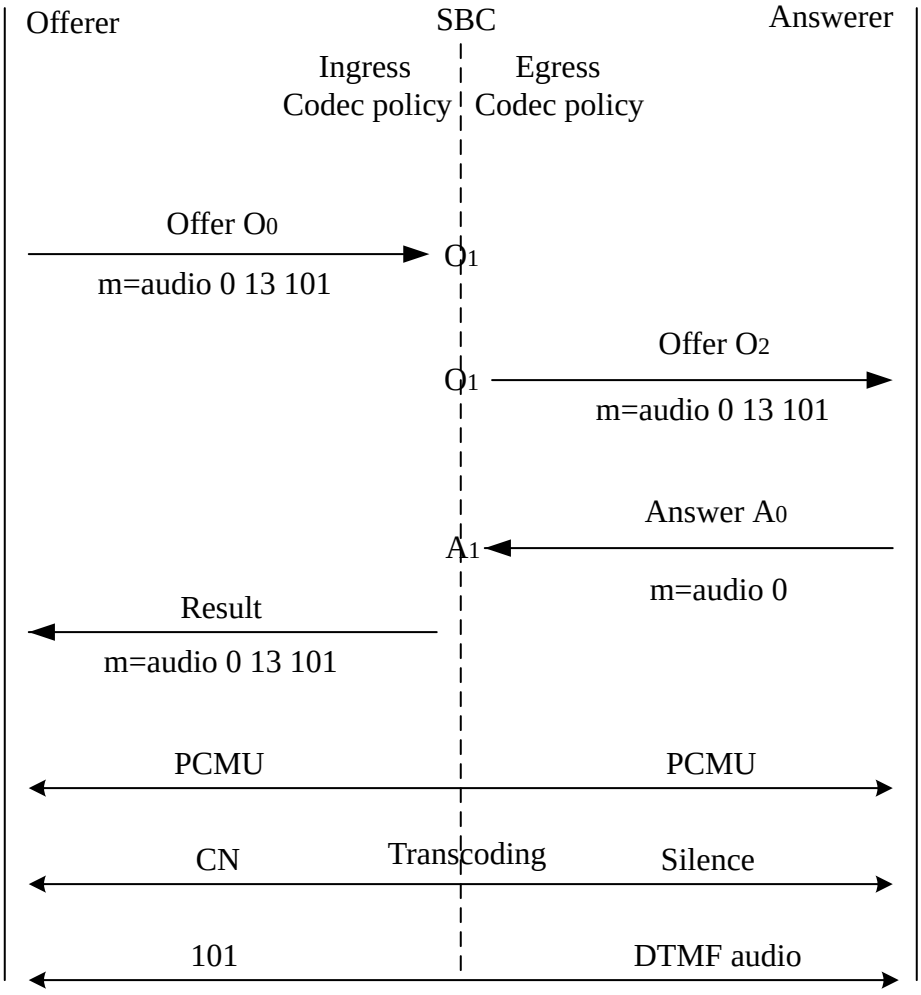
In the following scenario, the SBC transcodes both Comfort Noise (CN) and telephone event codecs.

1. The Offerer sends an offer (O°) with an SDP m-line for an audio codec PCMU to the SBC, and includes both CN and telephone-event signaling codecs.
2. The codec-policy for the ingress realm allows all codecs and outputs O^1. The SBC egress codec-policy takes O^1 as input, allows all codecs, and produces output O^2.
3. The Answerer responds with the answer in A° containing the PCMU audio codec. The egress codec-policy produces answer A^1 with no changes.

4. The SBC applies the add-codecs-on-egress parameter from the ingress codec-policy to add CN and telephone-event to A[1]. The result contains the audio codec PCMU and the CN and telephone-event signaling codecs because the SBC excepts signalling codecs from the rule that the system can apply the add-codecs-on-egress parameter only by the egress codec policy.

Even with CN transcoding enabled, the SBC does not transcode the audio codec (PCMU). The SBC transcodes silence detection and generation, as well as CN packet detection and generation. The SBC transcodes telephone-events to in-band DTMF audio.

Ingress codec policy	Settings	Egress codec policy	Settings
allow codecs	*	allow codecs	*
add codecs on egress	CN telephone event	add codecs on egress	N/A
order codecs	N/A	order codecs	N/A
dtmf-in-audio	preferred	dtmf-in-audio	preferred

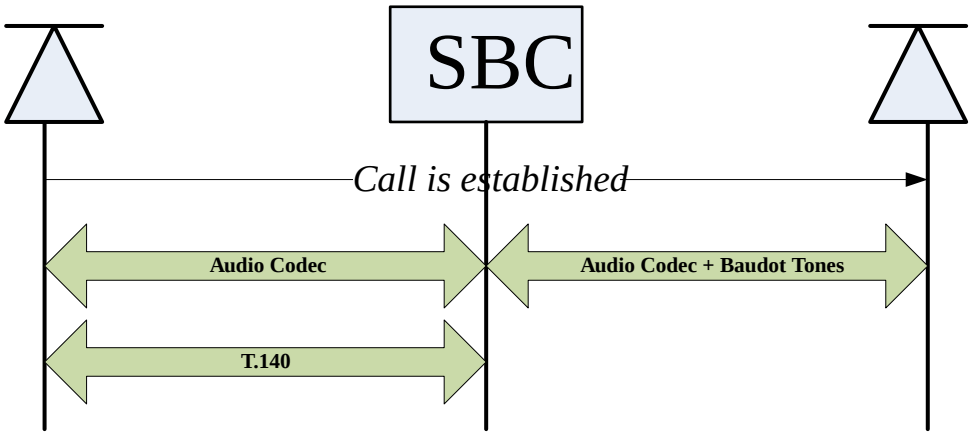


T.140 to Baudot Relay

The T.140 to Baudot Relay feature uses the SBC's transcoding resources to relay T.140 text messages to Baudot tones and vice versa. The T.140 Protocol is used for multimedia text

conversation over IP and is designed as a replacement for TDD devices. Baudot tones are a common protocol in the US in Telecommunications Device for the Deaf (TDD). Details of the protocol implementation are available in TIA/EIA-825-A. The T.140-Baudot relay is a regulatory requirement, and is specified for both emergency and non-emergency traffic. This feature is not available on virtual platforms; it is only available on Acme Packet hardware platforms.

T.140 to Baudot transcoding entails that one call leg is provisioned with a Baudot-capable codec, and the other call leg accepts T.140 in the SDP. Additionally, the T.140 side includes an audio stream. Once this scenario is established, the call may begin as audio-to-audio. At any point forward, T.140 may be received on one call leg or Baudot tones may be received on the other call leg. Each text indication will be transcoded to its compliment on the other side of the call. When T.140 <-> Baudot tone transcoding is active, the existing audio stream is preempted.



The system's transcoding hardware detects baudot tones in the incoming audio stream and generates T.140 packets on an outbound text stream. In the reverse direction T.140 packets will be detected on the text stream and Baudot tones will be generated on the appropriate outgoing audio stream.

T.140 to Baudot relay is invoked when an outbound **codec-policy** removes any text "m=" line from the egressing offer. The processing also removes all non-Baudot-tone capable codecs from the egress SDP offer. If at this point no Baudot-capable codecs remain in the SDP, the call is torn down.

Baudot Tones capable codecs:

- PCMA
- PCMU
- EVRC
- EVRC0
- EVRC1
- EVRCB
- EVRCB0
- EVRCB1

Codec Policy Configuration for T.140 to Baudot Relay

To support T.140 to Baudot relay, configure the **allow-codecs** parameter in the **codec-policy** with text:no. This value causes the SBC to strip any "m=text" occurrence in the outbound

INVITE and enable T.140 to Baudot transcoding. When SDP passes through the SBC and a text "m=" line is removed on the egress side of the call, then T.140-baudot relay/transcoding will be invoked for that call.

Unlike other transcodable codecs, T.140 is not valid in **add-codecs-on-egress** parameter.

Limitations

The following scenarios are not supported:

- Configuration of T.140 in **add-codes-on-egress**
- Hairpin calls (T.140 - Baudot Relay - T.140)
- Lawful Intercept
- SRTP for T.140

T.140 to Baudot Relay Examples

This section includes two examples; access-side call initiation and core-side call initiation. The following codec-policy is applied to the Core-side realm.

```
ACMEPACKET (codec-policy)#
name T140-to-tone
allow text: no *
add-codecs-on-egress PCMU
```

Figure 20-1 Call Initiated from the Access-side

This diagram shows the baseline implementation of T.140 to Baudot tones. The UE sends an INVITE into the core, via the SBC. Codec-policy indicates SBCto remove all text codecs (including T.140), it also indicates to add PCMU. All non-Baudot codecs are removed from the egress invite - AMR in this case. The core confirms PCMU in a 200 OK. The SBC forwards the 200 OK to the UE confirming the initially-offered codecs (AMR and T.140).

The call is set up so that the SBC will transcode audio between AMR and PCMU, and seamlessly relay T.140 to and from Baudot tones.

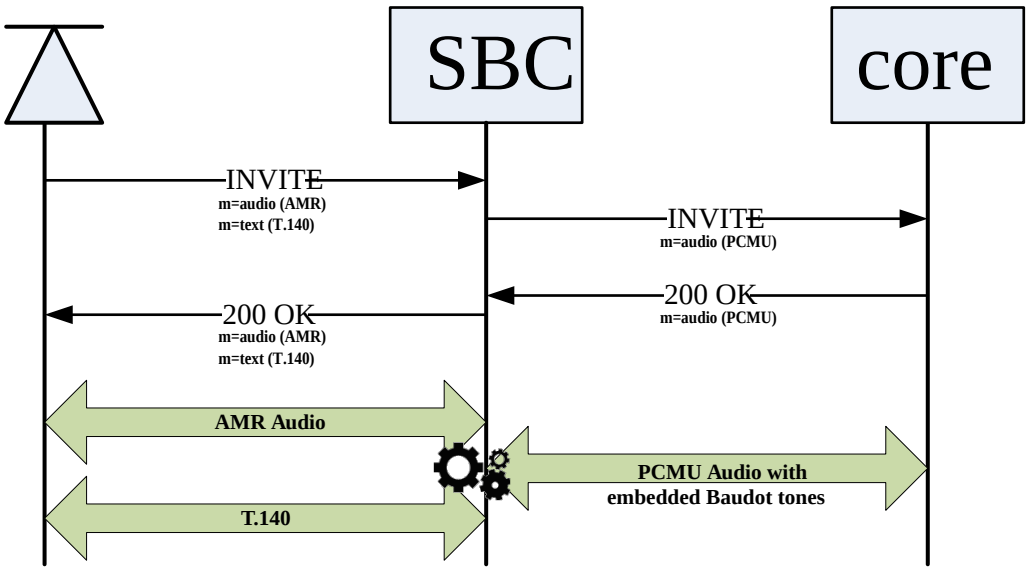
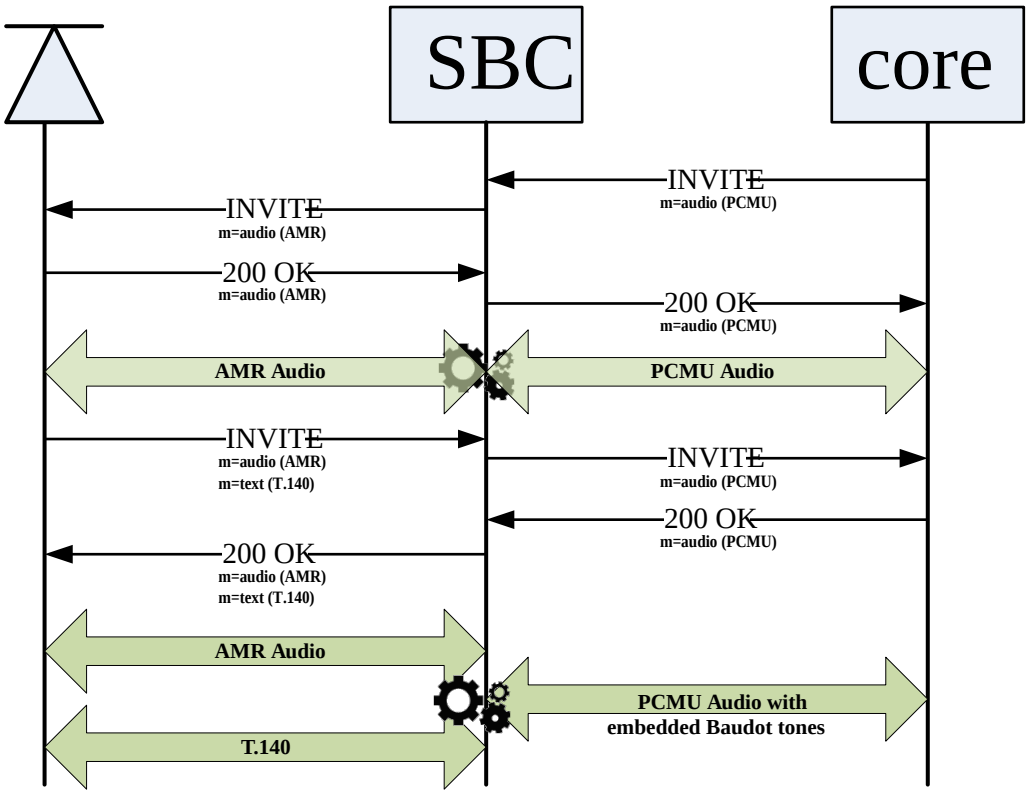


Figure 20-2 Call Initiated from the Core with T.140 reINVITE

In this example, the call is initiated from the core. The SBC transcodes between PCMU and AMR via expected means. Then, in the same dialog, the access side then sends a reINVITE adding T.140 and AMR. Codec policy dictates to accept T.140 on the UE-side and strip the text codec and all non-Baudot capable codecs from the core side. Additionally, the SBC is ready for T.140 / Baudot relay (transcoding). Thus when the UE begins sending T.140, they are transcoded into tones within the PCMU stream.



AMR-NB and AMR-WB Specifications

The Oracle Communications Session Border Controller supports Adaptive Multi-Rate Narrow Band & Wide Band codecs. All configurations of this codec, as indicated by SDP, are transcodable except when the following SDP parameters are enabled:

- robust-sorting
- interleaving
- CRC

When AMR is configured in a codec policy's add-codecs-on-egress parameter, it is forwarded from the Oracle Communications Session Border Controller with the following default settings:

- 12.2 kbps (AMR-NB)
- 23.85 kbps (AMR-WB)
- RTP/IF1 format
- No redundant packets

- bandwidth efficient default payload
- No CRC frame
- 20ms default ptime

AMR AMR-WB Payload Type Mapping

When the Oracle Communications Session Border Controller connects endpoints that choose AMR with different dynamic payload types but equivalent AMR parameters, payload type remapping can be handled by the local network processor rather than using transcoding resources. In prior releases, the system transcoded all AMR-to-AMR calls or all AMR-WB-to-AMR-WB calls when the Offer/Answer differed only in the assignment of dynamic payload types, or in the values assigned to certain AMR/AMR-WB parameters.

The decision to transcode is based entirely upon a comparison of the SDP Offer presented by the session initiator and the SDP Answer returned by the session responder. Transcoding is NOT REQUIRED when all of the following conditions are met.

- The offer/answer codecs are identical.
- The offer/answer payload formats (specified by the AMR/AMR-WB octet-align parameter) are identical.
- The offer/answer mode-sets (specified by the AMR/AMR-WB mode-set parameter) are identical or intersect.

To use this feature you must create appropriate media profiles (examples given below) and codec policies. You must also enable the audio-payload-type-mapping option in the media-manager-config.

> **Note:**
>
> On a per-call basis, if AMR/AMR-WB payload mapping is invoked, the system can not also perform RFC 2833 to SIP INFO/H.323 UII payload translation. If a call starts with RFC 2833 translation and per a call change AMR/AMR-WB payload mapping, the system prioritizes the AMR/AMR-WB action and ceases RFC 2833 translation.

AMR AMR-WB octet-align Parameter

AMR/AMR-WB sessions can be established in either bandwidth-efficient or octet-aligned format. Octet-aligned format increases processing efficiency, and is required for AMR's robust sorting, interleaving and frame CRC capabilities. With octet-aligned format, all fields in the AMR/AMR-WB header are individually aligned to octet boundaries by the addition of padding bits. With bandwidth-efficient format, only the full payload is octet aligned resulting in the addition of fewer padding bits.

Format usage is specified by the optional AMR/AMR-WB octet-align parameter, which assumes one of two values: 0 (the default), indicating bandwidth-efficient format, or 1 indicating octet-aligned format. In the absence of the octet-align parameter, bandwidth-efficient format is employed.

A mismatch of offer/answer octet-align values requires transcoding as shown in the following table, where N/P indicates the absence of the octet-align parameter.

```
octet-align
  Values
```

Offer	Answer	Format	Transcoding Required
0	0	bandwidth-efficient	No
1	1	octet-aligned	No
0	1	mismatch	Yes
1	0	mismatch	Yes
N/P	N/P	bandwidth-efficient	No
N/P	0	bandwidth-efficient	No
N/P	1	mismatch	Yes
0	N/P	bandwidth-efficient	No
1	N/P	mismatch	Yes

AMR AMR-WB mode-set Parameter

AMR/AMR-WB codecs support 9 standard encoding modes as shown in the following table.

Frame Content	Mode Indicator	Frame Content
AMR 4.75 kbit/sec	0	AMR-WB 6.60 kbit/sec
AMR 5.15 kbit/sec	1	AMR-WB 8.85 kbit/sec
AMR 5.90 kbit/sec	2	AMR-WB 12.65 kbit/sec
AMR 6.70 kbit/sec	3	AMR-WB 14.25 kbit/sec
AMR 7.40 kbit/sec	4	AMR-WB 15.85 kbit/sec
AMR 7.95 kbit/sec	5	AMR-WB 18.25 kbit/sec
AMR 10.2 kbit/sec	6	AMR-WB 19.85 kbit/sec
AMR 12.2 kbit/sec	7	AMR-WB 23.05 kbit/sec
NA	8	AMR-WB 23.85 kbit/sec

The optional AMR/AMR-WB mode-set parameter can be used to restrict the session mode set to a sub-set of the 9 standard modes. If a mode-set is specified, it must be honored, and frames encoded with modes not specified within the sub-set must not be sent in any RTP payload or used in codec mode requests. In the absence of a mode-set parameter, the inclusive mode-set (0,1,2,3,4,5,6,7,8) provides the default value.

Other AMR AMR-WB Parameters

AMR/AMR-WB support the following additional optional parameters. These parameters are irrelevant in the transcoding decision process. However, they can change the SDP in the answer returned to the session initiator.

```
channels                      mode-change-neighbor
crc                         mode-change-period
interleaving                mode-set
max-red                     octet-align
maxtime                     ptime
mode-change-capability      robust-sorting
```

Examples and Explanations

The following sections illustrate scenarios of various levels of complexity.

Basic Scenarios illustrate default-based offer/answer exchanges. In the absence of explicitly configured AMR/AMR-WB octet-align parameter in the original offer or answer, both the initiator and the responder opt for bandwidth efficient payload format. In a similar fashion, the absence of an AMR/AMR-WB mode-set parameter indicates mutual acceptance of the

inclusive mode-set (0,1,2,3,4,5,6,7). This section also includes examples of intersecting mode-sets.

Advanced Scenarios illustrate offer/answer exchanges that contain AMR or AMR-WB mode-set parameters.

SDP examples contain rtpmap and fmtp attributes whose syntax is described in the following two sections.

rtpmap Attribute

The rtpmap attribute takes the form:

```
a=rtpmap:<payloadType> <encodingName>/<clockRate>[/audioChannels]
```

payloadType: contains a dynamically-assigned RTP payload type as specified in RFC 3551, RTP Profile for Audio and Video Conferences with Minimal Control. The Internet Assigned Numbers Authority (IANA) has designated RTP payload types 96 through 127 as available for dynamic assignment.

encodingName: contains the codec identifier (AMR or AMR-WB in the following tables).

clockRate: contains the sampling rate.

audioChannels: (used only for audio streams, and optional) contains the number of audio channels — not used in transcoding decisions.

fmtp Attribute

The fmtp attribute shown in the following tables takes the form:

```
a=fmtp:<amrParameter> <parameterValues>
```

amrParameter contains one of the AMR/AMR-WB RTP-specific parameters listed below.

```
channels
crc
interleaving
max-red
maxtime
mode-change-capability
mode-change-neighbor
mode-change-period
mode-set
octet-align
ptime
robust-sorting
```

parameterValue: contains the value of the specified parameter.

Basic Scenarios

Examples provided in this section are supported by the following codec policy and media profiles.

```
codec-policy
        name    net192
        allow-codecs    *
        add-codecs-on-egress    AMR::PT96 AMR-WB::PT97 AMR::PT98
    AMR-WB::PT99
        order-codecs
        force-ptime    disabled
        packetization-time    30
        dtmf-in-audio    disabled


media-profile
        name                        AMR
        subname                     PT96
        media-type                  audio
        payload-type                96
        transport                   RTP/AVP
        req-bandwidth               0
        frames-per-packet           0
        parameters
        average-rate-limit          0
        peak-rate-limit             0
        max-burst-size              0
        sdp-rate-limit-headroom     0
        sdp-bandwidth               disabled
        police-rate                 0
        standard-pkt-rate           0


media-profile
        name                        AMR-WB
        subname                     PT97
        media-type                  audio
        payload-type                97
        transport                   RTP/AVP
        req-bandwidth               0
        frames-per-packet           0
        parameters
        average-rate-limit          0
        peak-rate-limit             0
        max-burst-size              0
        sdp-rate-limit-headroom     0
        sdp-bandwidth               disabled
        police-rate                 0
        standard-pkt-rate           0


media-profile
        name                        AMR
        subname                     PT98
```

```
        media-type                audio
        payload-type              98
        transport                 RTP/AVP
        req-bandwidth             0
        frames-per-packet         0
        parameters                mode-set="0,1,2,3"
        average-rate-limit        0
        peak-rate-limit           0
        max-burst-size            0
        sdp-rate-limit-headroom   0
        sdp-bandwidth             disabled
        police-rate               0
        standard-pkt-rate         0


media-profile
        name                      AMR-WB
        subname                   PT99
        media-type                audio
        payload-type              99
        transport                 RTP/AVP
        req-bandwidth             0
        frames-per-packet         0
        parameters                mode-set="0,1,2,3"
        average-rate-limit        0
        peak-rate-limit           0
        max-burst-size            0
        sdp-rate-limit-headroom   0
        sdp-bandwidth             disabled
        police-rate               0
        standard-pkt-rate         0
```

The following SDP reflects the case where the session responder answers with the same codec and payload type as presented by the session initiator.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 97 98 99 96
from AMR-WB/PT97 media profile	• a=rtpmap:97 AMR-WB/16000/1
from AMR-WB/PT98 media profile	• a=rtpmap:98 AMR/8000 • a=fmtp:98 mode-set=0,1,2,3
from AMR-WB/PT99 media profile	• a=rtpmap:99 AMR-WB/16000/1 • a=fmtp:99 mode-set=0,1,2,3
initiator offer	• a=rtpmap:96 AMR/8000
Answer received from responder	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000
Answer sent to initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000

Codecs	Match	Values
codec	match	AMR/AMR
payload formats	match	bandwidth-efficient (default value)
mode-sets	match	inclusive mode-set (0,1,2,3,4,5,6,7)
payload mapping	match	96/96

Here, given the identical offer and answer, neither transcoding nor payload type mapping is required.

The following SDP reflects the case where the session responder answers with the codec offered by the session initiator, but with a different payload type.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 97 98 99 96
from AMR-WB/PT97 media profile	• a=rtpmap:97 AMR-WB/16000/1
from AMR-WB/PT98 media profile	• a=rtpmap:98 AMR/8000 • a=fmtp:98 mode-set=0,1,2,3
from AMR-WB/PT99 media profile	• a=rtpmap:99 AMR-WB/16000/1 • a=fmtp:99 mode-set=0,1,2,3
initiator offer	• a=rtpmap:96 AMR/8000
Answer received from responder	• m=audio xx RTP/AVP 97 • a=rtpmap:97 AMR/8000
Answer sent to initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000

Codecs	Match	Values
codecs	match	AMR/AMR
payload formats	match	bandwidth-efficient (default value)
mode-sets	match	inclusive mode-set (0,1,2,3,4,5,6,7)
payload mapping	no match	96/97

Here, given the identical offer/answer except for the payload type, transcoding is not required; payload type mapping is used.

The following SDP reflects the case where the session responder answers with a codec not offered by the session initiator.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 97 98 99 96
from AMR-WB/PT97 media profile	• a=rtpmap:97 AMR-WB/16000/1

SDP	Contents
from AMR-WB/PT98 media profile	• a=rtpmap:98 AMR/8000 • a=fmtp:98 mode-set=0,1,2,3
from AMR-WB/PT99 media profile	• a=rtpmap:99 AMR-WB/16000/1 • a=fmtp:99 mode-set=0,1,2,3
initiator offer	• a=rtpmap:96 AMR/8000
Answer received from responder	• m=audio xx RTP/AVP 97 • a=rtpmap:97 AMR/16000
Answer sent to initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000

Codecs	Match	Values
codecs	no match	AMR/AMR-WB
payload formats	match	bandwidth-efficient (default value)
mode-sets	match	inclusive mode-set (0,1,2,3,4,5,6,7)
payload mapping	no match	96/97

Here, given that the offer and answer codecs (AMR and AMR-WB) are not identical, transcoding is required.

The following SDP reflects the case where the offer specifies the inclusive mode-set, and the answer specifies a non-inclusive mode-set.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 97 98 99 96
from AMR-WB/PT97 media profile	• a=rtpmap:97 AMR-WB/16000/1
from AMR-WB/PT98 media profile	• a=rtpmap:98 AMR/8000 • a=fmtp:98 mode-set=0,1,2,3
from AMR-WB/PT99 media profile	• a=rtpmap:99 AMR-WB/16000/1 • a=fmtp:98 mode-set=0,1,2,3
initiator offer	• a=rtpmap:96 AMR/8000
Answer received from responder	• m=audio xx RTP/AVP 98 • a=rtpmap:96 AMR/8000 • a=fmtp: 98 mode-set=0,1,2,3
Answer sent to initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000

Codecs	Match	Values
codecs	match	AMR/AMR-WB
payload formats	match	bandwidth-efficient (default value)
mode-sets	intersect	(0,1,2,3,4,5,6,7) / (0,1,2,3)
payload mapping	no match	96/98

Here, given that the answer mode-set (0,1,2,3) intersects with the offered mode-set (0,1,2,3,4,5,6,7), transcoding is not required; payload type mapping is used.

The following SDP reflects the case where the offer specifies a non-inclusive mode-set, and the answer specifies the inclusive mode-set.

SDO	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 98 • a=rtpmap:98 AMR/8000 • a=fmtp:98 mode-set=0,1,2,3 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 96 97 99 98
from AMR/PT96 media profile	• a=rtpmap:96 AMR/8000
from AMR-WB/PT97 media profile	• a=rtpmap:97 AMR-WB/16000/1
from AMR-WB/PT99 media profile	• a=rtpmap:99 AMR-WB/16000/1 • a=fmtp:99 mode-set=0,1,2,3
initiator offer	• a=rtpmap:98 AMR/8000 • a=fmtp:98 mode-set=0,1,2,3
Answer received from responder	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000
Answer sent to initiator	• m=audio xx RTP/AVP 98 • a=rtpmap:98 AMR/8000 • a=fmtp:98 mode-set=0,1,2,3

Codecs	Match	Values
codecs	match	AMR/AMR
payload formats	match	bandwidth-efficient (default value)
mode-sets	intersect	(0,1,2,3) / (0,1,2,3,4,5,6,7)
payload mapping	no match	98/96

Here, given that the offer mode-set (0,1,2,3) intersects with the answer mode-set (0,1,2,3,4,5,6,7), transcoding is not required; payload type mapping is used.

Advanced Scenarios

Examples provided in this first section are supported by the following codec policy and media profile.

```
codec-policy
        name                        net182
        allow-codecs                *
        add-codecs-on-egress        AMR::PT97-5-6-7
        order-codecs
        force-ptime                 disabled
        packetization-time          20
        dtmf-in-audio               disabled


media-profile
        name                        AMR
        subname                     PT97-5-6-7
```

```
media-type                audio
payload-type              97
transport                 RTP/AVP
req-bandwidth             0
frames-per-packet         0
parameters                mode-set="5,6,7"
average-rate-limit        0
peak-rate-limit           0
max-burst-size            0
sdp-rate-limit-headroom   0
sdp-bandwidth             disabled
police-rate               0
standard-pkt-rate         0
```

The following SDP reflects the case where the offer and answer specify non-matching mode-sets.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • a=fmtp:96 mode-set=0,1,2,3,4 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 97 96
from AMR/PT97-5-6-7 media profile	• a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=5,6,7
initiator offer	• a=rtpmap:96 AMR/8000 • a=fmtp:96 mode-set=1,2,3,4
Answer received from responder	• m=audio xx RTP/AVP 97 • a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=5,6,7
Answer sent to initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • a=fmtp:96 mode-set=0,1,2,3,4

Codecs	Match	Values
codecs	match	AMR/AMR
payload formats	match	bandwidth-efficient (default value)
mode-sets	no match	(1,2,3,6) / (5,6,7)
payload mapping	no match	96/97

Here, given that the offer mode-set (1,2,3,4) does not intersects with the answer mode-set (5,6,7), transcoding is required.

The following SDP reflects the case where the offer and answer specify intersecting mode-sets.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • a=fmtp:96 mode-set=0,1,2,3,4 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 97 96
from AMR/PT97-5-6-7 media profile	• a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=5,6,7
initiator offer	• a=rtpmap:96 AMR/8000 • a=fmtp:96 mode-set=1,2,3,4
Answer received from responder	• m=audio xx RTP/AVP 97 • a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=4,5,6,7
Answer sent to initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • a=fmtp:96 mode-set=0,1,2,3,4

Codecs	Match	Values
codecs	match	AMR/AMR
payload formats	match	bandwidth-efficient (default value)
mode-sets	no match	(1,2,3,4) / (4,5,6,7)
payload mapping	no match	96/97

Here, given that the offer mode-set (1,2,3,4) intersects with the answer mode-set (4,5,6,7), transcoding is not required; payload type mapping is used.

Examples provided in this next section are supported by the following codec policy and media profiles.

```
codec-policy
        name                    net192
        allow-codecs            *
        add-codecs-on-egress    AMR::PT96-MAXRED AMR::PT97-BE-0257
                                AMR::PT98-OA AMR::PT99-OA-0257
        order-codecs
        force-ptime             disabled
        packetization-time      20
        dtmf-in-audio           disabled


media-profile
        name                    AMR
        subname                 PT96-MAXRED
        media-type              audio
        payload-type            96
        transport               RTP/AVP
        req-bandwidth           0
        frames-per-packet       0
        parameters              max-red=220
        average-rate-limit      0
        peak-rate-limit         0
```

```
        max-burst-size            0
        sdp-rate-limit-headroom   0
        sdp-bandwidth             disabled
        police-rate               0
        standard-pkt-rate         0


media-profile
        name                      AMR
        subname                   PT97-BE-0257
        media-type                audio
        payload-type              97
        transport                 RTP/AVP
        req-bandwidth             0
        frames-per-packet         0
        parameters                mode-set="0,2,5,7"
        average-rate-limit        0
        peak-rate-limit           0
        max-burst-size            0
        sdp-rate-limit-headroom   0
        sdp-bandwidth             disabled
        police-rate               0
        standard-pkt-rate         0


        media-profile
        name                      AMR
        subname                   PT98-OA
        media-type                audio
        payload-type              98
        transport                 RTP/AVP
        req-bandwidth             0
        frames-per-packet         0
        parameters                octet-align=1
        average-rate-limit        0
        peak-rate-limit           0
        max-burst-size            0
        sdp-rate-limit-headroom   0
        sdp-bandwidth             disabled
        police-rate               0
        standard-pkt-rate         0


media-profile
        name                      AMR
        subname                   PT99-OA-0257
        media-type                audio
        payload-type              99
        transport                 RTP/AVP
        req-bandwidth             0
        frames-per-packet         0
        parameters                mode-set="0,2,5,7",octet-align=1
        average-rate-limit        0
        peak-rate-limit           0
        max-burst-size            0
        sdp-rate-limit-headroom   0
```

```
sdp-bandwidth              disabled
police-rate                0
standard-pkt-rate          0
```

The following SDP reflects the case where the offer and answer differ only in the inclusion of an AMR/AMR-WB parameter in the answer.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 97 98 99 96
from AMR/PT97-BE-0257 media profile	• a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=0,2,5,7
from AMR/PT98-OA media profile	• a=rtpmap:98 AMR/8000 • a=fmtp:98 octet-align=1
from AMR/PT99-OA-0257 media profile	• a=rtpmap:99 AMR/8000 • a=fmtp:99 mode-set=0,2,5,7 octet-align=1
initiator offer	• a=rtpmap:96 AMR/8000
Answer received from responder	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • a=fmtp:96 max-red=220
Answer sent to initiator	• m=audio xx RTP/AVP 96 • a=rtpmap:96 AMR/8000 • a=fmtp:96 max-red=220

Codecs	Match	Values
codecs	match	AMR/AMR
payload formats	match	bandwidth-efficient (default value)
mode-sets	match	inclusive mode-set (0,1,2,3,4,5,6,7)
payload mapping	match	96/96

Here, given that the offer/answer are identical except for the AMR max-red parameter, neither transcoding nor payload type matching is required.

The following SDP reflects the case where the offer and answer differ only in the inclusion of an AMR/AMR-WB parameter in the offer.

SDP	Contents
Offer received from session initiator	• m=audio xx RTP/AVP 97 • a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=0,2,5,7 • a=fmtp:97 max-red=220 • INVITE SDP (partial, codec information only)
Offer sent to responder (per the net192 codec policy)	• m=audio xx RTP/AVP 96 98 99 97
from AMR/PT96-MAXRED media profile	• a=rtpmap:96 AMR/8000 • a=fmtp:96 max-red=220

SDP	Contents
from AMR/PT98-OA media profile	• a=rtpmap:98 AMR/8000 • a=fmtp:98 octet-align=1
from AMR/PT99-OA-0257 media profile	• a=rtpmap:99 AMR/8000 • a=fmtp:99 mode-set=0,2,5,7 octet-align=1
initiator offer	• a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=0,2,5,7 max-red=220
Answer received from responder	• m=audio xx RTP/AVP 97 • a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=0,2,5,7
Answer sent to initiator	• m=audio xx RTP/AVP 97 • a=rtpmap:97 AMR/8000 • a=fmtp:97 mode-set=0,2,5,7

Codecs	Match	Values
codecs	match	AMR/AMR
payload formats	match	bandwidth-efficient (default value)
mode-sets	match	(0,2,5,7) / (0,2,5,7)
payload mapping	match	97/97

Here, given that the offer/answer are identical except for the AMR max-red parameter, neither transcoding nor payload type matching is required.

ACLI Configuration

To enable AMR Mode-set payload type SDP rewriting:

1. Navigate to the media-manager-config element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# media-manager-config
ACMEPACKET(media-manager-config)#
```

2. options—Set the options parameter by typing options, a Space, the option-name audio-payload-type-mapping=yes with a “plus” sign in front of it, and then press Enter.

```
ACMEPACKET(media-manager-config)# options +audio-payload-type-mapping=yes
```

   If you type the option without the “plus” sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration’s options list, you must prepend the new option with a “plus” sign as shown in the previous example.

3. Save and activate your configuration.

EVS Codec Transcoding Support

Enhanced Voice Services (EVS) is a super-wideband speech audio codec developed by 3GPP and documented in TS 26.441. EVS supports source-controlled variable bit rate, sampling rates of 8, 16, 32, or 48 kHz, dynamic payload type, and an interoperability mode for AMR-WB. The Oracle Communications Session Border Controller (SBC) supports typical transcoding

features. EVS can also analyze traffic signaling, allowing it to change to the correct core EVS codec when necessary. These changes can occur at every 20ms frame boundary. The SBC also supports transcoding EVS to and from all supported transcodable codecs unless the EVS mode is using super-wideband or fullband EVS bandwidths. Note that, by configuration, you can set the SBC to recognize EVS-WB as transcodable, which is useful for scenarios such as supporting SRVCC handovers.

Before configuring the SBC to transcode EVS, you must enable it with the **setup entitlements** command. EVS codec feature support includes:

- transrating
- transcoding
- pooled transcoding
- RTCP generation
- AMR-WB interoperability and payload-type mapping

> **Note:**
>
> See the *Release Notes* for any platform-based EVS transcoding limitations.

Bitrate support per bandwidth includes:

- Narrowband (NB) — 5.9, 7.2, 8, 9.6, 13.2, 16.4, 24.4
- Wideband (WB) — 5.9, 7.2, 8, 9.6, 13.2, 13.2 channel-aware, 16.4, 24.4, 32, 48, 64, 96, 128 (6.6 ~ 23.85 for AMR-WB IO)
- Super-wideband (SWB) — 9.6, 13.2, 13.2 channel-aware, 16.4, 24.4, 32, 48, 64, 96, 128
- Fullband (FB) — 16.4, 24.4, 32, 48, 64, 96, 128

EVS data is always octet-aligned in both Primary and AMR-WB interoperability mode.

Interoperation with AMR-WB

EVS' AMR-WB interoperable (IO) mode provides backwards compatibility with endpoints that support AMR-WB, but don't support EVS. Based on user configuration and SDP offers, AMR-WB IO mode allows the SBC to deliver media between such endpoints without using trancoding resources.

EVS Supported Options

There are no required SDP Parameters for EVS. Some EVS parameters may have values that the SBC's DSP does not support. Supported values must be verified before the SBC makes transcoding decisions. if any of these parameter checks fail, the SBC marks the codec as non-transcodable

Unless noted otherwise, see 3GPP TS 26.445 and related specifications for complete parameter documentation. Optional SDP parameters include:

- **ptime**—The length of time in milliseconds represented by the media in a packet. See RFC 4566 for more details.
  For both EVS Primary mode and EVS AMR-WB IO mode, the supported ptimes are 20, 40, and 60 ms.
- **maxptime**—The maximum amount of media that can be encapsulated in each packet, expressed as time in milliseconds. See RFC 4566 for more details.

For both EVS Primary mode and EVS AMR-WB IO mode, the supported maxptimes are 20, 40, and 60 ms.

- **evs-mode-switch**—Specifies whether Primary mode or EVS AMR-WB IO mode should be used at the start or update of the session. The default of 0 specifies the use of primary mode.
- **hf-only**—Specifies whether to limit the session to header-full format. The default of 0 allows both compact and header-full format in both directions.
- **dtx**—Specifies whether or not to support discontinuous transmission. The default of 1 specifies that DTX is enabled.
- **dtx-recv**—This parameter enables (dtx-recv=1) or disables (dtx-recv=0) the receiver's DTX functionality towards the sender.
- **max-red**—Specifies the maximum number of milliseconds allowed between the first transmission of a frame, a redundant transmission and the corresponding redundant transmission. See RFC 4867 for more details.
- **channels**—Specifies the number of audio channels, with a default of 1. The SBC supports only 1 channel for transcoding.
- **cmr**—Specifies whether codec mode request (CMR) is supported for the session. The default of 0 enables all CMR values.

The following parameters apply only to **EVS Primary** mode:

- **br**—Specifies, in kilobits per second, the range of source codec bit-rate for EVS Primary mode in the session for both send and receive directions.
  Source codec bit-rates for the EVS codec

Source codec bit-rate (kbit/s)	Audio bandwidths supported	Source Controlled Operation Available
5.9 (SC-VBR)	NB, WB	Yes (Always On)
7.2	NB, WB	Yes
8.0	NB, WB	Yes
9.6	NB, WB, SWB	Yes
13.2	NB, WB, SWB	Yes
13.2 (channel aware)	WB, SWB	Yes
16.4	NB, WB, SWB, FB	Yes
24.4	NB, WB, SWB, FB	Yes
32	WB, SWB, FB	Yes
48	WB, SWB, FB	Yes
64	WB, SWB, FB	Yes
96	WB, SWB, FB	Yes
128	WB, SWB, FB	Yes

  If the given br value conflicts with the given bw value, the SBC DSP marks the codec as non-transcodable.
- **br-send**—Specifies, in kilobits per second, the range of source codec bit-rate for EVS Primary mode in the session for the send direction.
  If the given br-send value conflicts with the given bw value, the SBC DSP marks the codec as non-transcodable.

- **br-recv**—Specifies, in kilobits per second, the range of source codec bit-rate for EVS Primary mode in the session for the receive direction.
  The SBC can independently decode an EVS packet with any bit rate that is received. Thus, br-recv is not used in determining whether the codec is transcodable or not.
- **bw**—Specifies the audio bandwidth for EVS Primary mode to be used in the session for the send and the receive directions.
  For transcoding, the SBC DSP only supports nb, wb, and nb-wb.
- **bw-send**—Specifies the bandwidth to be used in the session for the send direction.
  For transcoding, the SBC DSP only supports nb, wb, and nb-wb.
- **bw-recv**—Specifies the bandwidth to be used in the session for the receive direction.
  For transcoding, the SBC DSP only supports nb, wb, and nb-wb.
- **ch-send**—Specifies the number of audio channels for the send direction. The default is 1.
- **ch-recv**—Specifies the number of audio channels for the receive direction. The default is 1.
- **ch-aw-recv**—Enumerated setting for channel-aware mode. The default of 0 specifies that partial redundancy mode is not used for the receive direction at the start of the session.

The following parameters apply only to **EVS AMR-WB IO** mode. Optional parameters of AMR-WB not defined below may not be used in the EVS AMR-WB IO mode.

- **mode-set**—Restricts the active codec mode set to a subset of all modes when the EVS codec operates in AMR-WB IO.
  Source codec bit-rates for the AMR-WB Interoperable Modes of the EVS codec

Mode Indicator	Source codec bit-rate (kbit/s)
0	6.6
1	8.85
2	12.65
3	14.25
4	15.85
5	18.25
6	19.85
7	23.05
8	23.85

> **Note:**
>
> The SBC supports only mode-sets 0-7 for both AMR and AMR-WB.

- **mode-change-period**—Specifies a number of frame-blocks, N (1 or 2). This is the frame-block period at which codec mode changes are allowed for the sender. See RFC 4867 for more details.
- **mode-change-capability**—Specifies if the client is capable of transmitting with a restricted mode change period. See RFC 4867. The default, and only allowed value is 2 in EVS AMR-WB IO.
- **mode-change-neighbor**—Permissible values are 0 and 1. If 1, the sender SHOULD only perform mode changes to the neighboring modes in the active codec mode set. See RFC 4867 for more details.

Default EVS Media Profile

By default, the SBC uses the following parameters for EVS, referenced in the configuration as **EVS**.

```
media-profile
        name                                    EVS
        subname
        media-type                              audio
        payload-type
        transport                               RTP/AVP
        clock-rate                              16000
        req-bandwidth                           0
        frames-per-packet                       0
        parameters
        average-rate-limit                      6000
        peak-rate-limit                         0
        max-burst-size                          0
        sdp-rate-limit-headroom                 0
        sdp-bandwidth                           disabled
        police-rate                             0
        standard-pkt-rate                       0
        as-bandwidth                            0
```

CLI Commands

CLI command changes made to support EVS include:

- The **show sipd codecs** command includes **EVS Count**.
- The **show sipd transcode** command includes **EVS**.
- The **show xcode codecs** command includes **EVS-AMR-WB** sessions.

SNMP

This section presents SNMP OID detail the SBC uses to support EVS.

ap-codec.mib

Object Name/OID	Description
apCodecRealmCountEVS 1.3.6.1.4.1.9148.3.7.1.1.1.33	The count of SDP media streams received in the realm that negotiated to the EVS codec.

The EVS realm statistic **apCodecRealmCountEVS** is available in the **apCodecRealmStatsEntry**.

ap-smgmt.mib

Object Name/OID	Description
apSysXCodeEVSCapacity 1.3.6.1.4.1.9148.3.2.1.1.49	The percentage of licensed EVS transcoding utilization (non pollable).
apSysMgmtXCodeEVSUtilGroup 1.3.6.1.4.1.9148.3.2.4.2.35	Object to monitor licensed EVS transcoding utilization.

New Traps—The new SNMP OID **apSysXCodeEVSCapacity** is added to transcoding utilization statistics, as reported in the **apSysMgmtGroupTrap**. When utilization falls below 80%, the system sends the **apSysMgmtGroupClearTrap**.

Trap Name (and Clear Trap Name)	Description
apSysMgmtCPULoadAvgTrap (apSysMgmtCPULoadAvgClearTrap)	The system generates the trap when the CPU Load Average Alarm exceeds its minor alarm threshold. The system sends the clear trap when the CPU load average recedes to the minor alarm level.

Capability MIB IODs

Object Name/OID	MIB file
apSmgmtXCodeEVSUtilCap 1.3.6.1.4.1.9148.2.1.8.59	ap-smgmt.mib
apCodecRealmCodecCap9 1.3.6.1.4.1.9148.2.1.13.11	ap-codec.mib

Alarms

The Licensed EVS Transcoding Capacity Threshold Alarm is a warning triggered when the EVS transcoding utilization exceeds 95% of licensed capacity. This alarm does not affect the system's health score. The system clears this alarm when the EVS transcoding utilization falls below 80% of licensed capacity.

RADIUS

The Acme-FlowType_FS{1,2}_{F,R} AVPs reflect the use of the EVS codec.

EVS Configuration Detail

This section discusses aspects of SBC configuration that you must consider in addition to typical transcoding configuration for EVS.

Payload Type Mapping

The user enables EVS AMR-WB IO payload type mapping using the **media-manager-config** option **audio-payload-type-mapping=yes**. If the option is not present, EVS AMR-WB IO payload type mapping is disabled.

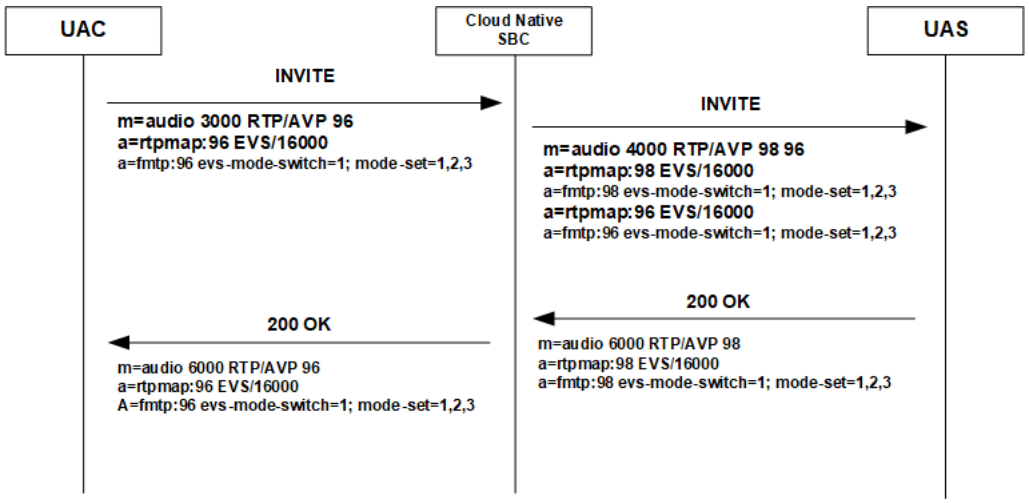
```
media-manager
 …
 options
audio-payload-type-mapping=yes
```

Furthermore, the **payload-type** is not critical as long as the **audio-payload-type-mapping=yes** option is configured. Normally the PT used in the network is configured in the **payload-type** field. But the SBC matches against codec name/string instead of **payload-type**.

EVS Call Flow Example

The flow depicted below provides an example that uses the **audio-payload-type-mapping=yes** to enable payload type mapping. In addition, the flow uses **add-codecs-on-egress** configuration in the egress codec policy configured as **EVS::PT98**. The offer contains

EVS in AMR-WB IO mode and the SBC adds a separate EVS AMR-WB IO with a different payload type. The answerer selects PT 98 and since audio-payload-type-mapping=yes, so transcoding is not required, and payload type mapping is used. The offer and answer are both octet-aligned and they have intersecting mode-sets, so transcoding is not required and payload type mapping is used.



AMR-WB to EVS AMR-WB IO Transparent Call Example

This example explains how you can configure the SBC and applicable resources to support calls between EVS and AMR-WB endpoints without requiring transcoding, and therefore not consuming SBC transcoding resources. This use case, established by logic within the SBC, can accommodate calls in either direction.

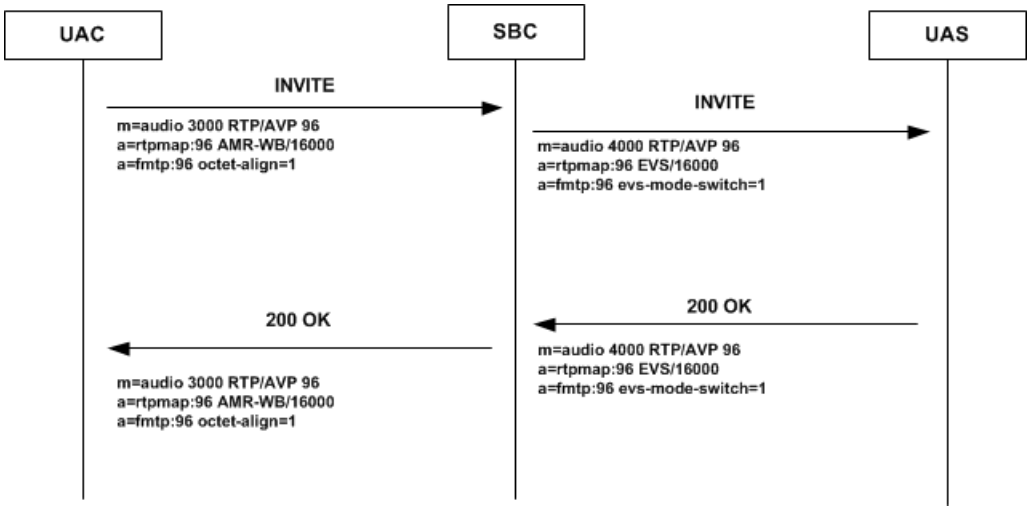
In this call flow a codec policy is set on the egress realm to not allow AMR-WB and to add EVS AMR-WB IO on egress. The offer contains AMR-WB. The SBC recognizes that the incoming AMR-WB codec matches the EVS codec to be added:

- They have the same octet-align
- EVS codec is in interoperable mode (evs-mode-switch=1)
- You have a media-profile configured that matches the parameters presented in the AMR-WB offer

The SBC strips AMR-WB and adds EVS with the same payload type as AMR-WB even though EVS is not configured with that payload type.

```
codec-policy
name Transparent-AMR-WB-IO
allow-codecs * AMR-WB:no
add-codecs-on-egress EVS::WBIO
force-ptime disabled
```

The call proceeds transparently between AMR-WB and EVS.



EVS and SRVCC

The SBC includes support and requirements that you must consider when handling call flows that include the EVS codec and Single Radio Voice Call Continuity (SRVCC) handovers. This support includes basic functionality as well as transcoding considerations when EVS is in super-wideband mode.

Media Management

Enable **mm-in-realm** in the core realm to support transcoding EVS within environments that include SRVCC. Oracle also recommends that you enable **codec-manip-in-realm** for these deployments.

Transcoding Free Operation

You can configure the SBC to handle scenarios that cannot use transcoding, but still support super-wideband end stations. Examples of such scenarios include SRVCC handovers wherein super-wideband is established before the handover. The **srvcc-trfo** parameter in the **realm-config** allows you to establish this functionality on a per-realm basis for the EVS codec. This behavior is compliant with TS24-237 (proxy mode) for EVS calls.

For example, assume a call established from the IMS core towards a VoLTE leg is using EVS-SWB and the codec needs to change because of an SRVCC event. With this configuration in place, the SBC forces codec renegotiation with the far end in the event of a SRVCC handover from a packet to a circuit switching environment. The codec change on the VoLTE leg needs to match what the Mobile Switching Center (MSC) server offers to establish transcoder free operation. Assume the MSC offers AMR-WB to follow the process.

1. The SBC recognizes AMR-WB being presented to the VoLTE UE and refrains from sending the 200 OK as an answer to the INVITE (proxy mode).
2. Instead, the SBC sends it own INVITE towards the SRVCC UE, replacing EVS SWB in the SDP to force a renegotiation. Instead of EVS SWB, this INVITE contains the SDP from the SRVCC UE (AMR-WB).
3. The IMS core receives the SBC INVITE and propagates it towards the SRVCC UE.
4. Ultimately the SRVCC UE accepts AMR-WB as codec and replies towards the SBC with a 200 OK.
5. The IMS core sends this 200 OK to the SBC (ATU-STI).
6. The SBC replies with its own 200 OK to the MSC (STN-SR).

7. RTP restarts using AMR-WB between the VoLTE UE and the SRVCC UE.

This codec renegotiation happens end-to-end with the ATGW in the media path.

Navigate to the core **realm-config** and configure this parameter using the syntax below.

```
ORACLE(realm-config)# srvcc-trfo evs
```

This 'manual' trigger by the SBC to force SDP re-negotiation prevents the call from requiring transcoding to EVS-SWB after the handover to the circuit switching environment.

Circumventing EVS SWB Transcoding Scenarios

You can configure SBC to maintain a call that may otherwise fail when an SRVCC event puts the SBC in the position of having to transcode from EVS SWB. When configured, the SBC can recognize the requirement to transcode EVS SWB, and present EVS WB to the SRVCC UE. The SBC supports transcoding EVS WB so there is no need to drop the call. This configuration also allows the SBC to continue to use EVS SWB for the opposing UE.

You can configure the SBC to support these scenarions using the **realm-config** option, **srvcc-evs-swb-to-wb**.

For example, assume a call has been established with EVS SWB when an SRVCC handover occurs. EVS SWB is not available within an MSC/3G network. By default, the call would fail. But when configured with the **srvcc-evs-swb-to-wb** option, the SBC internally changes the presented codec from EVS SWB to EVS WB on ingress, while continuing to present EVS SWB back to the egress. This allows the SBC to present this transcodable codec and support the SRVCC handover, while continuing to use EVS SWB on the other side.

Configure this option using the syntax below.

```
ORACLE(realm-config)# options +srvcc-evs-swb-to-wb
```

If you type options and then the option value without the plus sign, you overwrite any previously configured options. To add a new option to an options list, pre-pend the new option with a plus sign as shown in the previous example.

In addition, this function requires that you have an EVS media profile name as an allowed codec on both sides of the call. The default EVS profile can serve this purpose.

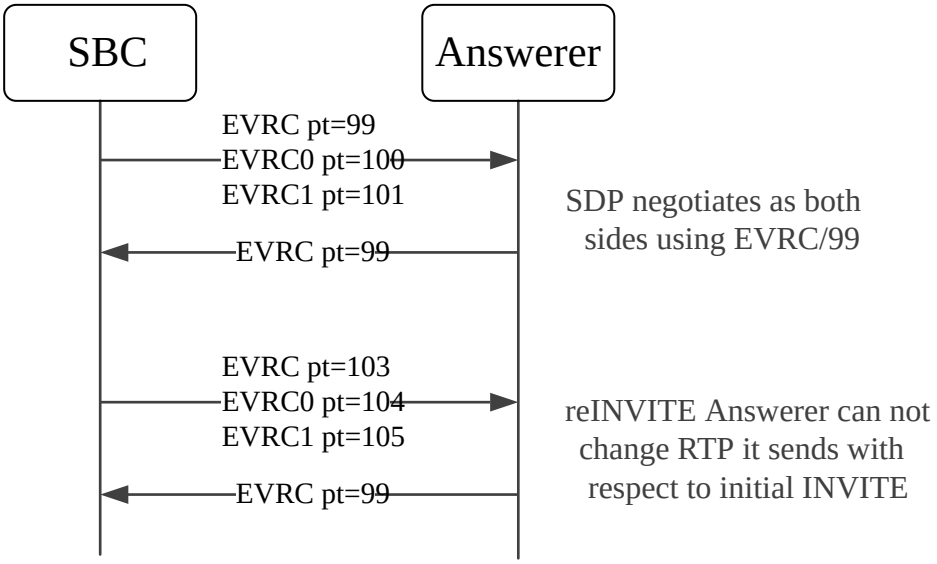
Asymmetric Dynamic Payload Types Enablement

Transcoding Support for Asymmetric Dynamic Payload Types supports calls with asymmetric payload types such that a codec offered with one payload type and answered with another payload type is acceptable to the SBC. This behavior requires transcoding resources.

The SBC's default behavior when an endpoint answers an SDP offer with a different payload type than offered is to use what the endpoint replied with as if that were what the SBC included in its SDP offer. The SBC would then expect to receive the same payload type that the endpoint offered. This is referred to as symmetric payload type mapping, whereby the payload type number is the same for both media flows.

The SBC needs to support the asymmetric case when codecs with dynamic payload types are used. For example, a call may be set up with the dynamic codec using one payload type value, and a reINVITE may use a different payload type value for the same codec. Because of the range of remote UE equipment's behavior in a reINVITE case, the SBC can support this via its Asymmetric Dynamic Payload Type feature. That is, some devices do not necessarily use the

currently negotiated payload type, they may use previously negotiated payload types. As such devices interact with the default SBC behavior, one-way audio may result.



For transcoded calls, enabling this feature places no additional load on the system. But for calls that are not transcoded, this feature consumes transcoding resources.

Configurations for non-Standard PT Cases

This section explains two configuration options you can use to support rare call flow issues associated with payload types. The first scenario includes non-transcoded call flow wherein a UAC presents an unexpected payload type (PT) within the context of a re-INVITE. This flow requires additional logic for the SBC to handle the re-INVITE. The second scenario includes the SBC handling payload types within the context of matching a **codec-policy** with traffic even though the **sub-names** are not the same. Both of these scenarios become supported when you enable their respective configuration options.

Asymmetric PT for Calls without Transcoding

In rare instances, the SBC may need to handle a dialog that uses three different PTs for the same codec within a call that does not require transcoding. The SBC supports payload type mapping without transcoding when all of the following conditions are met:

- The offer/answer codecs are identical.
- The offer/answer payload formats (specified by the AMR/AMR-WB octet-align parameter) are identical.
- The offer/answer mode-sets (specified by the AMR/AMR-WB mode-set parameter) are identical or intersect.
- You have set the **audio-payload-type-mapping=yes** option in the **media-manager**.

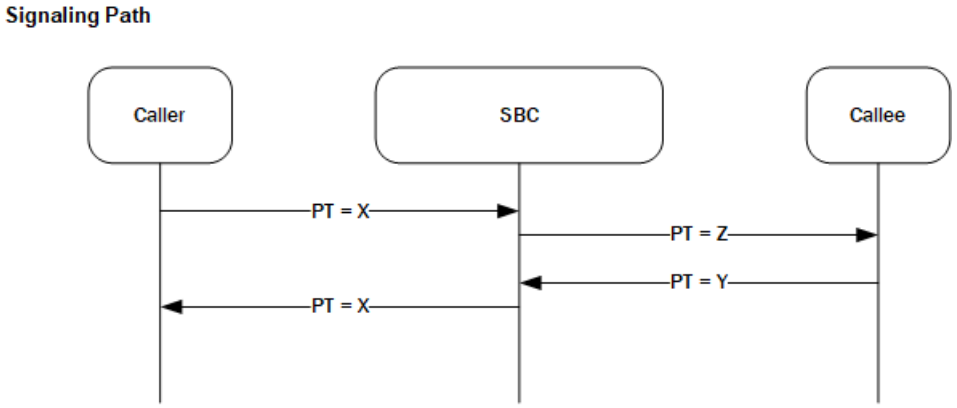
All of these conditions may be met even though the caller and callee are presenting different PT values. The SBC, by default, behaves in compliance with the RFC 3264 (8.3.2 Changing the Set of Media Formats), which recommends against scenarios like the ones this feature accommodates. The SBC, however, provides you with configuration options to overcome these scenarios.

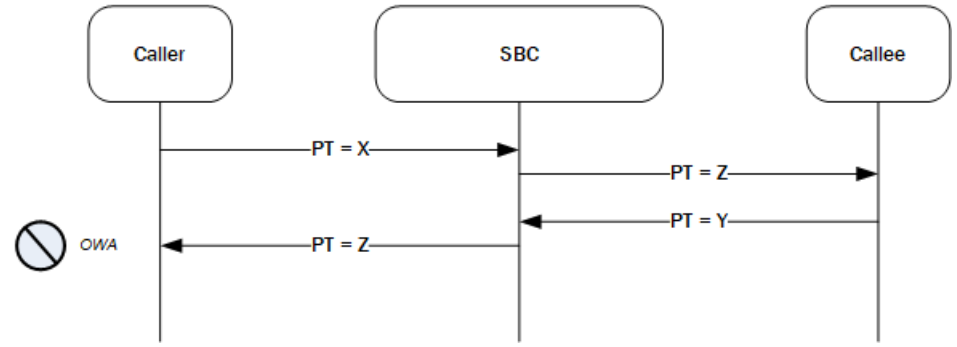
> **Note:**
>
> The SBC performs PT mapping on both the signaling plane (SDP offer/answer) and the media plane (RTP PT header).

The diagram below presents a case wherein the re-INVITE issue addressed by this feature occurs. This call flow uses a new codec and a previously used PT during a re-INVITE sequence. Ultimately, the UAS issues a previously used PT in its Re-INVITE response causing the SBC to be unable to perform payload mapping on the calling (west) flow.

This call flow proceeds as follows:

1. A caller presents an INVITE during the signaling phase using payload type Z (PT 116 for codec AMR-WB), X (PT 96 for codec AMR).
2. The callee responds to the INVITE , with payload type Z (PT 116 for codec AMR-WB).
3. This results in a non-transcoded call established on AMR-WB (116).
4. Later, the caller issues a Re-INVITE with a new offer that includes PT X (associating now 96 for AMR-WB). Recall that X was initially associated with AMR in the initial offer. Here, the caller endpoint is not compliant with RFC 3264.
5. To comply with RFC3264, the SBC modifies the already used PT X (96) to PT Y (100) in the outgoing INVITE.
6. The callee responds to the INVITE, but presents the previously used payload type Z (PT 116 for AMR-WB), establishing asymmetric payload type mapping at the egress.
7. Since the **audio-payload-type-mapping** feature is enabled, the SBC modifies the PT Z (116) to payload X (96) and sends it in the SIP response (200 OK) towards the caller.
8. At the media level, without this feature enablement, the audio_pt mapping is done as follows.
   - Called (east) flow: PT X maps to PT Z (96 to 116)
   - Calling (west) Flow: PT Z maps to PT X (116 to 96)
9. But the callee is an asymmetric endpoint and can accept and send different payload types. Therefore, the callee starts sending RTP on PT Y (100) and expecting RTP on PT Z (116).
10. At the media level, the SBC cannot map the PT Y (100) to the identifier "X" (96) on the calling (west) flow, so the call degrades to one-way audio (OWA).
11. The RTP on the called (east) side of the call flows correctly.

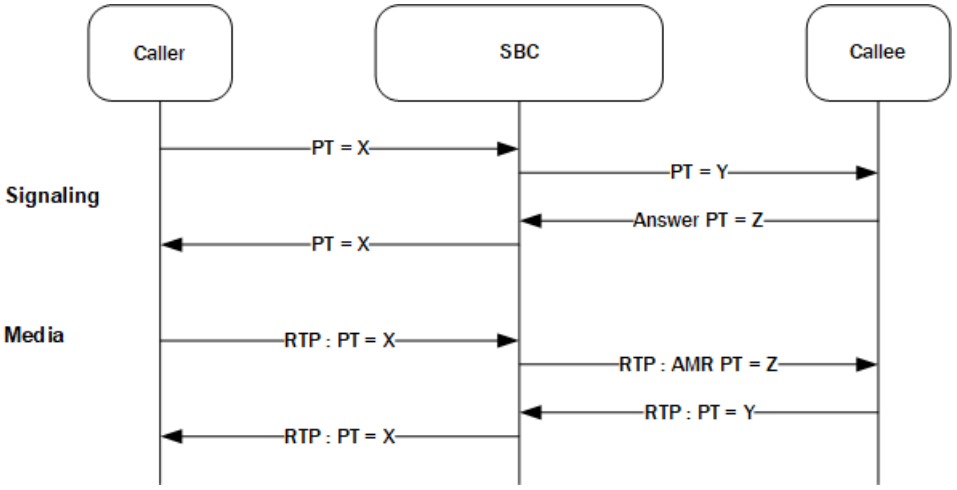




To support this call flow, you can set the **asymmetric-pt** option on **realm-config** that must support this asymmetric PT function.

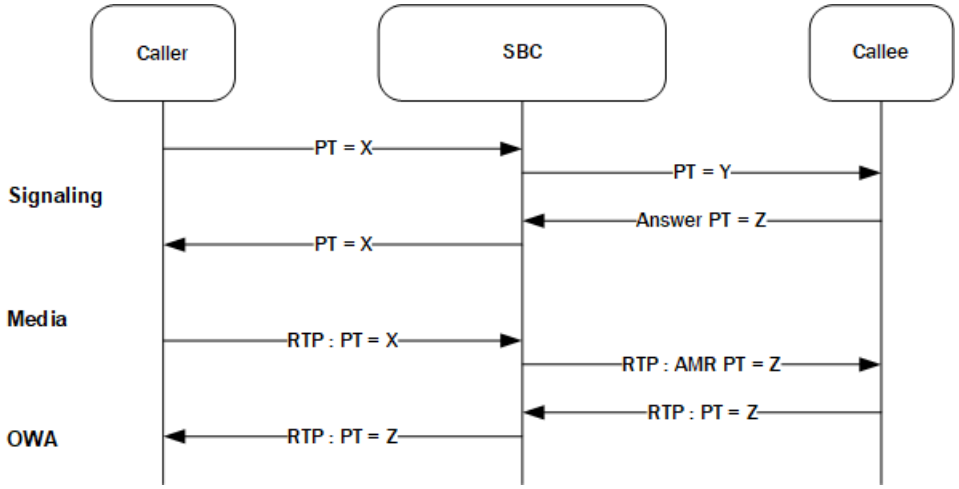
```
ORACLEORACLE(realm-config)#options +asymmetric-pt
```

This option is applicable only for calls that start as non-transcoded calls, but are faced with a new codec and a previously used PT during a re-INVITE sequence. If you enable the **asymmetric-pt** option, the use case works properly. In this case, the EP honors the PT sent in the offer and sends the RTP on the same PT. This case results in asymmetric PT mapping on the egress leg.



You must exercise caution when considering how the **asymmetric-pt** option would work in your environment. For example, when you enable it, the use-case below does not work.

Note that an endpoint can be either asymmetric or symmetric. You enable the **asymmetric-pt** option only when you know that the you need to handle asymmetric endpoints, and therefore need to send and receive RTP on different PTs. The result again is one-way audio (OWA). If the caller endpoint below is symmetric, meaning it sends RTP on the same PT specified in the 200 OK response, you should not enable this option.



Loose Matching on Subnames

Call flows can use different media profile subnames titled within **media-profile** objects even though those profiles are essentially the same. Similar to the issue above, you can also configure the SBC with the **loose-subname-match** option on **realm-config** to support calls when the only variance between codec negotiation are the subnames used. This issue can cause call issues for both transcoded and non-transcoded calls.

Consider the case wherein multiple **media-profile** objects are applicable to a given call because they are all used with AMR-WB. Note that they all have different subnames.

```
media-profile
name AMR-WB
subname default31
parameters mode-set="0,1,2"
octet-align=1
```

> **Note:**
>
> This first profile will not be matched because it has octet-align=1.

```
media-profile
name AMR-WB
subname default3G
parameters mode-set="0,1,2"
octet-align=0
```

> **Note:**
>
> This profile will be used, supported by this feature.

```
media-profile
name AMR-WB
subname defaultA1
parameters mode-set="0,1,2,3,4,5,6,7,"
octet-align=1
```

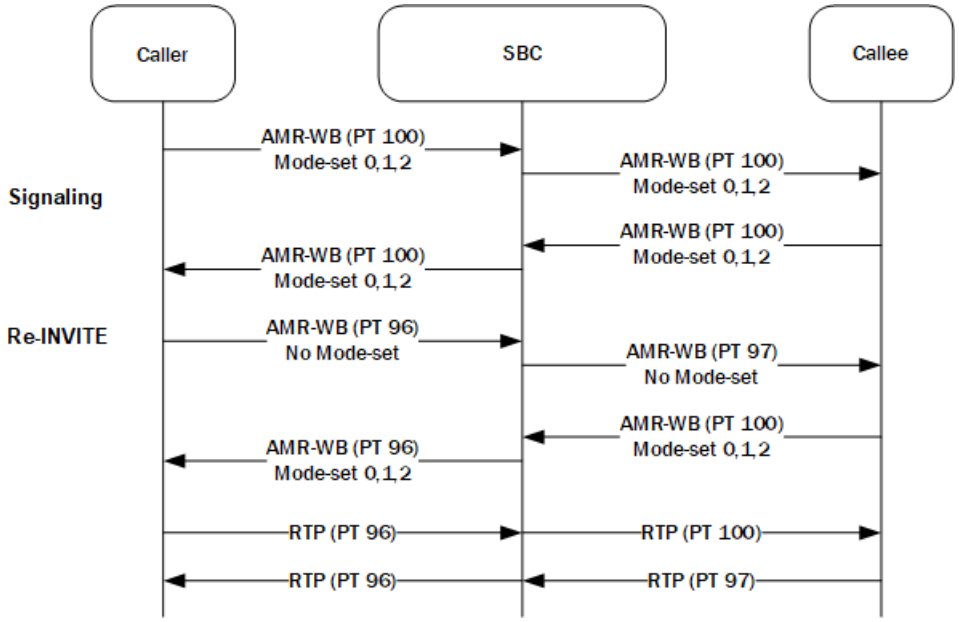
> **Note:**
>
> This third profile will not be matched because it has octet-align=1.

```
media-profile
name AMR-WB
subname defaultAll
parameters mode-set="0,1,2,3,4,5,6,7,"
octet-align=0
```

> **Note:**
>
> This profile will be used, supported by this feature.

Consider the following sequence, wherein this feature applies:

1. An initial call is connected with AMR-WB, mode-set = 0,1,2. This occurs during the signaling cycle.
2. In the Re-INVITE, the caller sends an offer with all mode-set and the callee's 200 OK includes the subset (mode-set = 0,1,2).
   Here the outgoing offer matches the "defaultAll" media profile and the response matches default3G. Since the codec, per the profile, does not match, the system does not initiate audio-pt mapping.
3. Here the outgoing PT 97 matches the profile with subname "defaultAll" media profile and the response to PT 100 matches the profile with subname "default3G".



In this flow, the SBC converts the the RPT of PT 96 from UAC to RTP PT 100. The SBC then also converts RTP 97 from the UAS to RTP 96 towards the UAC.

To support these call flows, you set the **loose-subname-match** option on the egress **realm-config**.

```
ORACLEORACLE(realm-config)#options +loose-subname-match
```

To have the system perform asymmetric PT mapping and audio PT mapping, along with loose media-profile matching, you enable all three options covered above, including **audio-payload-type-mapping=yes**, **asymmetric-pt**, and **loose-subname-match**.

> **Note:**
>
> If the **loose-subname-match** is not enabled, then the audio PT mapping does not work and the system forwards PT 100 in re-invite to the Ingress and does not convert the RTP audio PT from Egress to RTP PT 96 on west flow.

Configure Transcoding for Asymmetric Dynamic Payload Types

Transcoding support for asymmetric dynamic payload types enables the Oracle Communications Session Border Controller to perform transcoding when the Real-time Transport Protocol (RTP) is offered with one payload type and is answered with another payload type. Enable transcoding for asymmetric dynamic payload types from the command line.

Additional applicable features include supporting specific disparate PT mapping and **media-policy** subname matching.

Before You Begin

- Confirm that you are in Superuser mode.

Procedure

1. Access the **media-manager-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

2. Type select to begin editing.

```
ORACLE(media-manager-config)# select

ORACLE(media-manager-config)#
```

3. Use the **options +audio-allow-asymmetric-pt** command to enable support for asymmetric payload types.

```
ACMEPACKET#(media-manager) options +audio-allow-asymmetric-pt
ACMEPACKET#(media-manager)
```

4. Use the **audio-payload-type-mapping** option to enable support for AMR-WB payload mapping without using transcoding resources.

```
ACMEPACKET#(media-manager) options +audio-allow-asymmetric-pt
ACMEPACKET#(media-manager)
```

5. Type **done** to save your configuration.
6. To proceed with both PT mapping and subname matching, navigate to the realm-config on which you need those features (call egress).

```
ACMEPACKET#(media-manager) exit
ACMEPACKET#(media-manager)realm-config
ACMEPACKET#(realm-config)select
```

7. To configure PT mapping when a re-INVITE introduces an unrecognized PT value during a non-transcoded call, set the **asymmetric-pt** option.

```
ACMEPACKET#(realm-config)options +asymmetric-pt
```

8. To configure loose subname matching for applying equivalent **codec-policy** objects to a call in case of PT value mismatches, set the **loose-subname-match** option

```
ACMEPACKET#(realm-config)options +loose-subname-match
```

Asymmetric Payload Type Support for RFC2833 Interworking

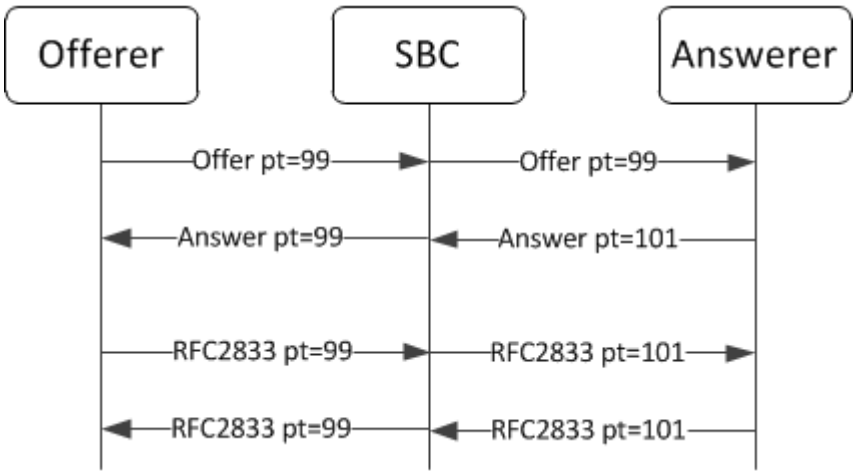
RFC3264 describes "asymmetric" behavior when each participant of a call leg sends can expect a different payload-type value for the RTP stream. Conversely, when the two participants in a call leg must send and receive an RTP stream using the same Payload Type value, the behavior is symmetric. The Oracle Communications Session Border Controller can

be configured to behave in an RFC-compliant manner, which is to support the asymmetric case. Symmetric-only cases are enforced by default.

Default Symmetric RFC2833 Interworking

The Oracle Communications Session Border Controller's default behavior when an endpoint answers an SDP offer with a different payload type value for RFC 2833 telephone-events than what the SBC sent, is to mimic what the Answerer replied with as the payload type value it (the SBC) will expect. This is referred to as symmetric payload type mapping, whereby the SBC sends RTP with the payload type value that the answerer replied with in the signaling phase of the call, and also expects that same payload type value when RTP is sent to it (despite having sent a different value in the initial SDP offer).

The following example shows the default behavior. When the Answerer returns PT 101, the Oracle Communications Session Border Controller is prepared to enforce symmetric-only behavior on the egress realm by expecting RFC2833 packets with Payload Type 101, even though it offered Payload Type 99.

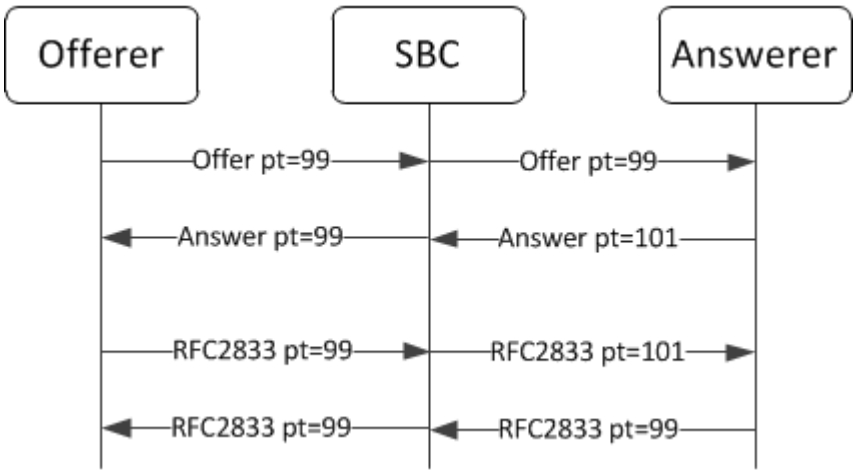


Since ingress-side behavior is to reply with the payload type offered, it expects payload type 99 and sends payload type 99. Thus to facilitate RFC2833 interworking, the payload types will be remapped from 99 -> 101 in the eastbound direction and 101 -> 99 in the westbound direction.

Asymmetric RFC2833 Interworking

The Oracle Communications Session Border Controller supports the asymmetric, RFC-compliant case by configuration. In this case, as the signaling is set up on the egress side of the call, the Oracle Communications Session Border Controller indicates it expects to receive RFC2833 packets with Payload Type 99, while the answerer indicates it expects to receive RFC 2833 packets with Payload Type 101. This valid call state can be accommodated by the

SBC when the **rfc2833-allow-asymmetric-pt** option is configured.



Therefore, the Oracle Communications Session Border Controller being able to support asymmetric payload types for RFC2833 packets, will remap from 99 ->101 in the eastbound direction and not have to remap the payload type value in the westbound direction.

The **rfc2833-allow-asymmetric-pt** option can be configured on a **sip-interface** or **session-agent** where this behavior is desired.

RFC2833 Interworking With and Without Transcoding Resources

In addition to the above behavior, it is important to note that the Oracle Communications Session Border Controller acts differently for forwarding media streams depending on if the call is transcoded or not. If the call is not transcoded, any RTP stream, regardless of the payload type being different from what was negotiated in the signaling phase will be forwarded through the system as is. In either of the above examples, if the Answerer sends RFC2833 packets with payload type 105, that RTP stream will be forwarded to the ingress leg of the call, untouched by payload mapping. This behavior could potentially mask the fact that the Oracle Communications Session Border Controller's egress interface is expecting a different payload type value than what is sent since the traffic is still forwarded to the ingress realm.

If the call is transcoded via **codec-policy**, any RTP stream that uses an unexpected payload type (as different from what was negotiated in the signaling phase) will be dropped by the system. In the above example, if the Answerer sends RFC2833 packets with payload type 105, that RTP stream will be dropped.

Separate Clock Rates for Audio and Telephone Events

RFC 4733 recommends that telephone events within an audio stream that use the same synchronization source (SSRC) should use the same timestamp clock rate as the audio channel. As an example, if SILK/16000 is being used as the audio stream then the flow should use telephone-event TE/16000. By default, the SBC complies with this behavior. You can configure the SBC, however, to support flows when using different clock rates for audio and telephone events. This allows the SBC to adapt to environments that do not follow the recommendation.

To perform this function, you configure the **allow-diff2833-clock-rate-mode** parameter on the **sip-interface**. The applicable interface can point to the caller or callee to accommodate offers with different clock rates. The most common scenario to use this features is when a caller sends an INVITE to the SBC with SDP that uses different codec and tel-event clock rates. Furthermore, a Re-INVITE (or UPDATE) coming from, for example, a callee, results in the SBC applying this function based on this parameter's setting on the callee's **sip-interface**. This is

also true for a Re-INVITE (or UPDATE) coming from a caller, with the SBC referring to the caller's ingress **sip-interface**.

> **Note:**
>
> If you enable this feature, and the SDP presents multiple telephone events, the SBC selects the clock rate of the top-most telephone-event in the SDP.

Mixed clock rate offers appear in the SDP media-level rtpmap attributes with different clock rates.

```
a=rtpmap:104 SILK/16000
a=rtpmap:100 telephone-event/8000
```

When presented with mixed clock rates and set to **use-2833-clock-rate**, the SBC accepts these mixed rate lines and generates RFC2833 packets using the signaled telephone-event clock rate to the corresponding interface. When set to **use-codec-clock-rate**, it generates rfc2833 packets using the audio codec's clock rate.

This configuration does not change any other signaling behavior at the egress. For example, if the SBC adds a codec as a result of your **add-codecs-on-egress** configuration, it also adds the corresponding telephone-event with the same clock rate. Nevertheless, it is still required that you consider the results of your codec processing and manipulation to achieve the desired effect of your **allow-diff2833-clock-rate-mode** setting. In addition, this feature does not change how the SBC handles transcoding processes, but it can affect how the SBC treats or generates DTMF digits, whether they are RFC2833 packets, SIP-INFO messages or inband DTMF tones.

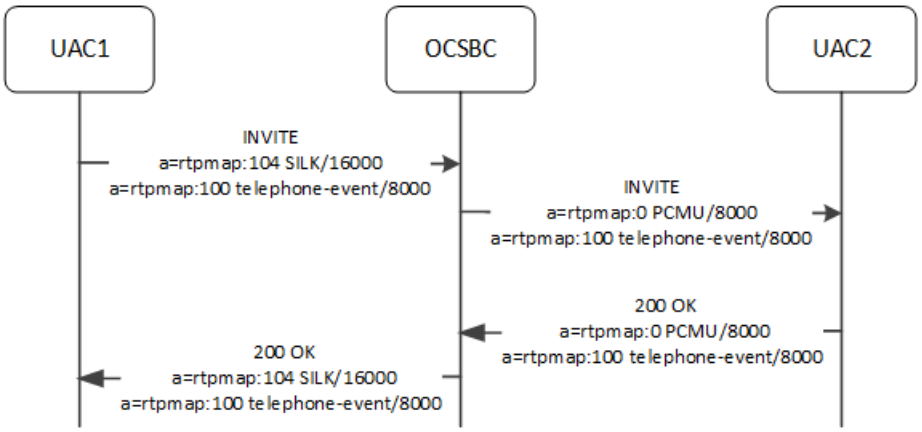
The existing behavior for inband DTMF signal generation/detection and generation of SIP-INFO messages is not affected by this feature:

- The SBC generates inband DTMF signal and encodes it into RTP packets according to the audio codec bandwidth.
  Note that the DTMF generator can only operate in 8000 Hz or 16000 Hz mode. This means the SBC cannot reliably detect inband DTMF signal that is encoded in RTP media using other bandwidths, such as OPUS FB, SILK SWB and EVS FB.
- For non-transcoded call, the system generates outgoing RFC2833 or SIP-INFO using the duration field in the received SIP-INFO or RFC2833 as is, regardless of the clock rate signaled for telephone-event.

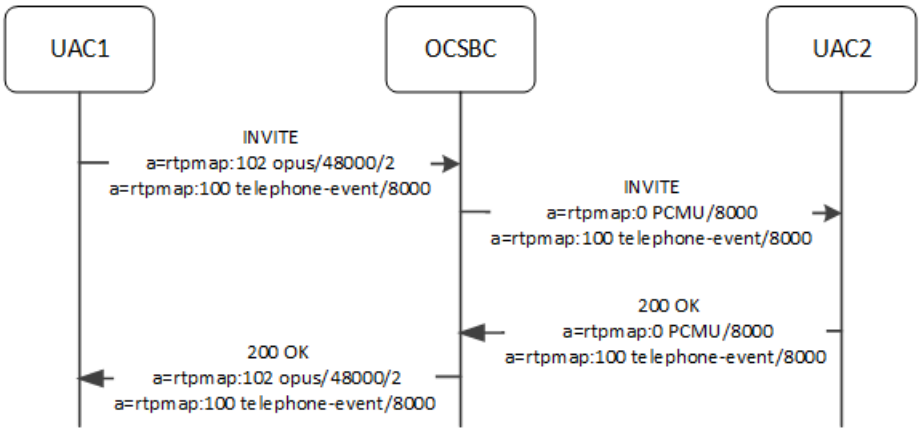
Call Flows for Differing Tel-event and Codec Clock Rates

This section presents examples of the application of the **allow-diff2833-clock-rate-mode** parameter within call flows.

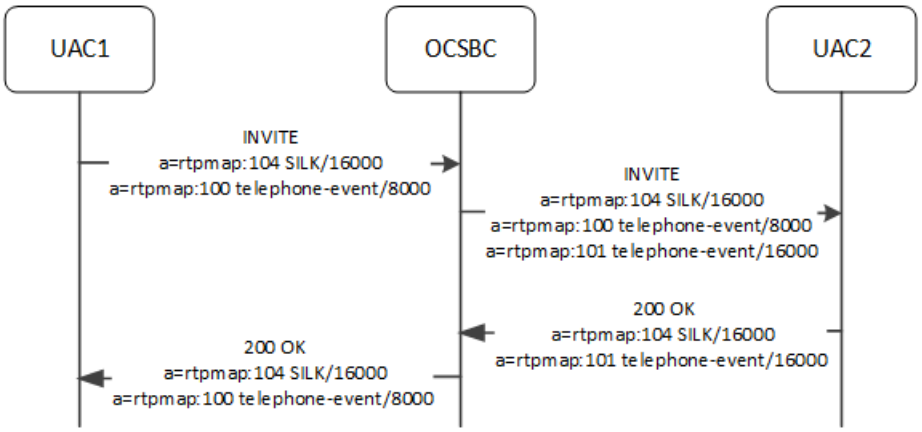
In this first example, you set the **allow-diff2833-clock-rate-mode** parameter on the ingress interface to **use-2833-clock-rate**. The diagram below shows the SBC accepting a different clock rate for the SILK codec and the tel-event from UAC1. This interface is ingress to UAC1. The SBC presents PCMU, and telephone-event with a clock rate of 8000. The SBC maintains the original offer and presents the 200OK to UAC1 using the clock rates it offered.



In this next example call flow, the diagram shows the SBC using the same **allow-diff2833-clock-rate-mode** configuration as the example above, allowing it to accept a different clock rate for the opus codec and the telephone-event from UAC1. Again, the SBC presents PCMU, and telephone-event with a clock rate of 8000. The SBC maintains the original offer and presents the 200OK to UAC1 using the clock rates it offered.



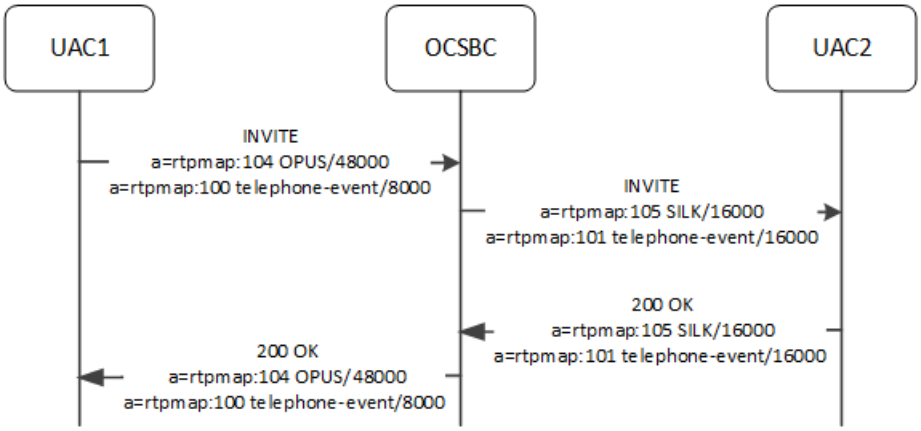
In this next example call flow, the diagram presents the SBC handling differing telephone-event and codec clock rates without transcoding. Notice that the "top" codec in A0 and O1 are the same. Without enabling features such as dtmf-in-audio detection or RTCP generation, which you could expect would force a transcoded call, the system treats the call as passthrough, even though the telephone-event clock rate differs on ingress and egress. This is because the audio codecs negotiated between ingress and egress are the same. Nevertheless, the SBC maintains the differing telephone-event clock rates and supports the flow.



In this next example call flow, your configuration includes **allow-diff2833-clock-rate-mode** on both the ingress (UAC1) and egress (UAC2) interfaces set to different values. In addition, the example is complicated by codec policies:

> **Note:**
>
> The designation of ingress and egress above is relative to UAC1. From the UAC2 perspective, the **sip-interface** pointing to UAC2 is the ingress interface.

- The **sip-interface** pointing to UAC2 is set to **use-2833-clock-rate**
- The **sip-interface** pointing to UAC1 is set to **use-codec-clock-rate**
- The **codec-policy** towards UAC1 allows OPUS and telephone-event
- The **codec-policy** towards UAC2 does not allow OPUS, but adds SILK-wideband



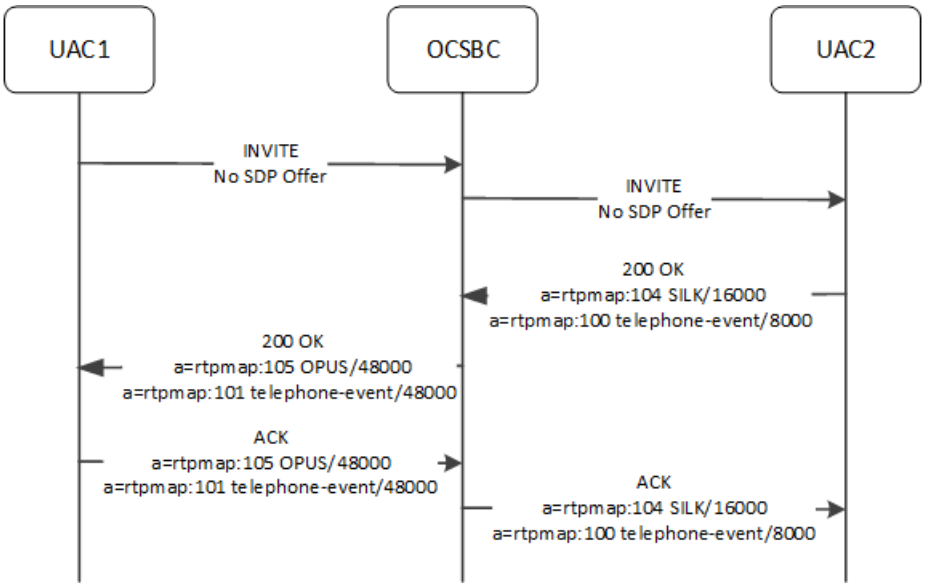
Using this configuration, and receiving an offer (O1) that contains OPUS/48000 and telephone-event/8000, SBC behavior includes:

1. Per ingress codec-policy, accepts OPUS/48000 and the telephone-event/8000.
2. Per egress codec-policy, removes OPUS, which is not supported.
3. Offers SILK/16000 and telephone-event/16000 to the egress.
4. Sends out the offer to UAC2.
5. Receives the 200 OK answer, which confirms the SDP, from UAC2.
6. Sends an answer towards UAC1 containing a telephone-event using the same clock rate originally offered (telephone-event/8000).
7. Towards ingress (UAC1), generates rfc2833 packets using the audio codec's clock rate (48000).
8. Towards egress (UAC2), generates rfc2833 packets using the telephone-event's clock rate (16000).

Note that the **allow-diff2833-clock-rate-mode** setting on the interface facing UAC2 has no effect on this flow because the audio codec and telephone-event both have the same clock rate. At egress, the SBC continues to match the clock rate of the telephone-event with a codec introduced by the **add-codecs-on-egress** parameter.

In this next example call flow, you have configured the SBC with a **codec-policy** towards UAC2 that accepts SILK and telephone-event; and **allow-diff2833-clock-rate-mode** is

use-2833-clock-rate. In addition, **codec-policy** towards UAC1 does not accept SILK; and adds OPUS at egress. **allow-diff2833-clock-rate-mode** is **use-2833-clock-rate**.



Using this configuration, and receiving an offerless INVITE, SBC behavior includes:

1. UAC1 sends an offerless INVITE. Offer is received from UAC2 in 200-OK. Therefore, UAC2 becomes ingress and UAC1 becomes egress.
2. Ingress codec policy (UAC2's **codec-policy**) does not remove anything from the offer since it accepts SILK and telephone-event both.
3. Egress codec policy (UAC1's **codec-policy**) removes SILK and a telephone-event mismatching clock rate of audio codec. It adds OPUS and a telephone-event matching clock rate of OPUS.
4. Answer towards UAC2 contains telephone-event of same clock rate as was received in offer.
5. Rfc2833 packets towards ingress and egress will be generated based on telephone event clock rate i.e. 8000 towards UAC2 (ingress) and 48000 towards UAC1 (egress).

Call Flows for Differing Egress Clock Rates

This feature operates on a call's ingress and/or egress interface. This section presents example call flows wherein the application of the **allow-diff2833-clock-rate-mode** parameter is depicted on a call's egress interface.

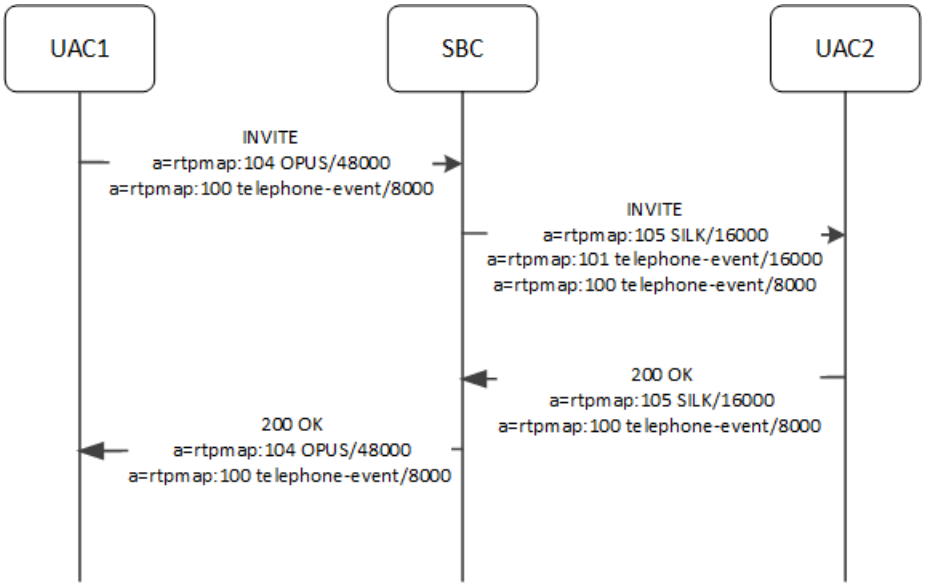
SDP negotiation follows the same procedures and rules for establishing media flows regardless of whether that negotiation is with the caller or callee. The previous section presented call flows using this feature during negotiation with the caller, or from the SBC perspective, on the call's ingress interface. This section presents call negotiation using this feature with both the caller and callee over the ingress and egress interfaces. The feature does not behave differently over either interface.

When you configure media-profiles, you include payload types, which include clock rates. You can choose to present a specific payload type for both your codec and for telephone events. With this feature, the clock rates for these payload types do not have to be the same. One of the target environments for this feature includes using 8k for telephone event clock rates regardless of the media codec. Although the working SBC configuration and the SDP negotiation ultimately determine the actual payload types used for each call, you can use this feature to successfully bypass the RFC 4733 recommendation for clock rates being the same.

Example 1

In this example, you have configured **allow-diff2833-clock-rate-mode** to the value **use-2833-clock-rate** on both the ingress and egress **sip-interface** elements.

- **allow-diff2833-clock-rate-mode** to **use-2833-clock-rate** on the ingress and the egress **sip-interface**
- Ingress **codec-policy** allows OPUS and telephone-event
- Egress **codec-policy** does not accept OPUS, but adds SILK wide-band, and telephone-event::8k



The call flow proceeds as follows:

1. The ingress **codec-policy** allows telephone-event having a different clock rate than audio codec. So, the first offer contains OPUS/48000 and telephone-event/8000.
2. The egress **codec-policy** removes OPUS, which is not supported, but does not remove telephone-event/8000. It adds SILK/16000 and the corresponding telephone-event/16000, and sends out the offer to UAC2.

> **Note:**
>
> The egress side allows a telephone-event clock rate that is different from the audio clock rate. Therefore, the system does not remove telephone-event/8000 from the offer. Because telephone-event/8000 is already present in the offer, the system does not add the configured telephone-event::8k (based on codec policy) again.

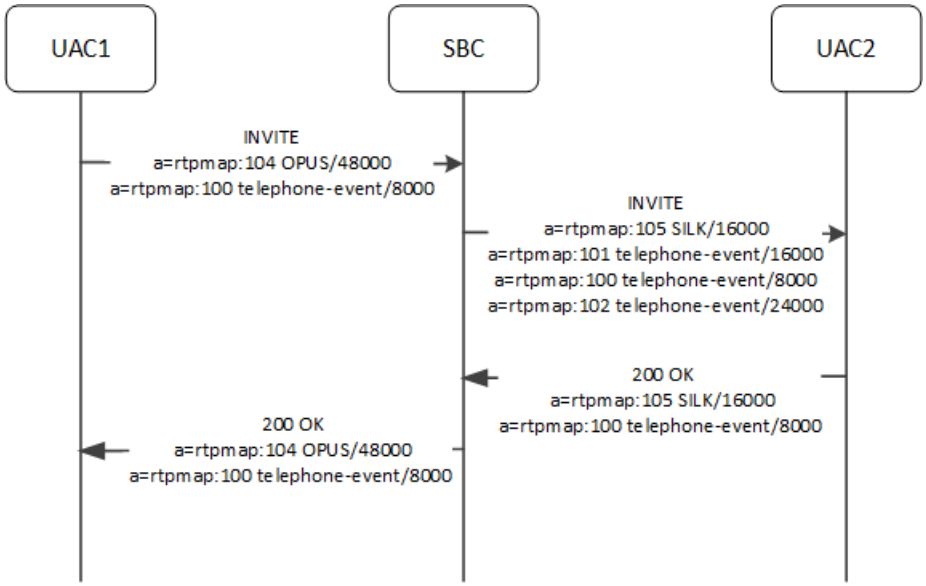
3. The egress side generates an answer and sends it towards ingress containing a telephone-event that is the same clock rate as was originally offered, in this case (telephone-event/8000).
4. The system generates RFC 2833 packets towards the ingress and egress using telephone-event clock rate, which is 8000.

Results of this call flow include the ingress **codec-policy** allowing OPUS and telephone-event. The egress **codec-policy** does not allow OPUS, but adds SILK-wideband, telephone-event::8k.

Example 2

In this example, you have configured **allow-diff2833-clock-rate-mode** to the value **use-codec-clock-rate** on both the ingress and egress **sip-interface** elements.

- **allow-diff2833-clock-rate-mode** to **use-codec-clock-rate** on the ingress and the egress **sip-interface**
- Ingress **codec-policy** allows OPUS and telephone-event
- Egress **codec-policy** does not accept OPUS, but adds SILK wide-band, telephone-event::8k and telephone-event::24k



The call flow proceeds as follows:

1. The ingress **codec-policy** allows telephone-event having a different clock rate than the audio codec. Therefore, the first offer contains OPUS/48000 and telephone-event/8000.
2. The egress codec-policy removes OPUS, which is not supported, and doesn't remove telephone-event/8000. It adds SILK/16000, the corresponding telephone-event/16000, and telephone-event/24000 and sends out the offer to UAC2.

> **Note:**
>
> The egress side allows different clock rates for telephone-event and audio, Therefore, the system does not remove the telephone-event/8000 from the offer. Since telephone-event/8000 is already present in the offer, the system does not add the configured telephone-event::8k (based on codec policy).

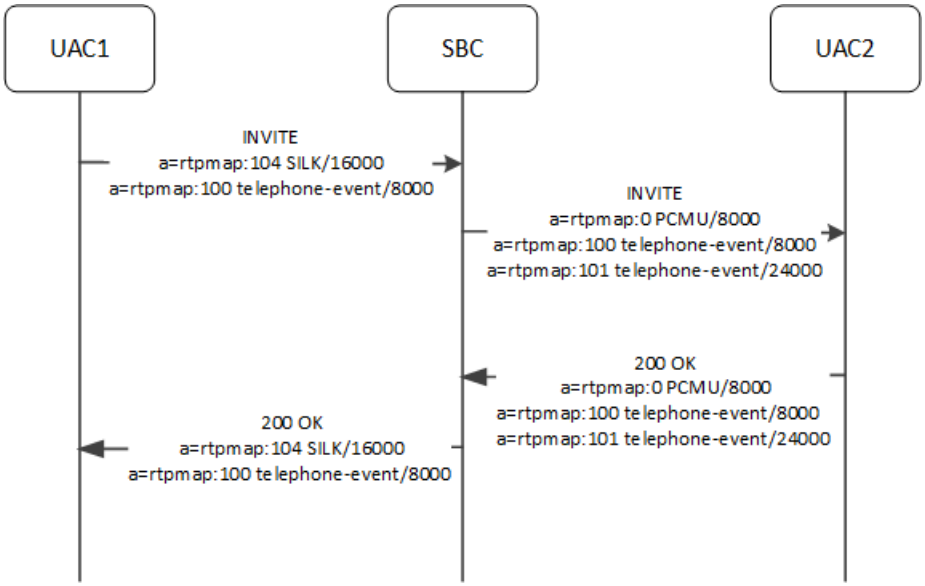
3. The egress side generates an answer and sends it towards the ingress containing a telephone-event with the same clock rate that was originally offered (telephone-event/8000).
4. The system generates RFC 2833 packets towards the ingress and egress, based on the audio codec's clock rate, which is 48000 towards ingress and 16000 towards egress.

Example 3

In this example, you have configured:

- **allow-diff2833-clock-rate-mode** to **use-codec-clock-rate** on the ingress and the egress **sip-interface**
- Ingress **codec-policy** allows SILK and telephone-event
- Egress **codec-policy** does not accept SILK, but adds PCMU, telephone-event::8k and telephone-event::24k

In addition, the SBC sends and receives multiple telephone-events in the answer.



The call flow proceeds as follows:

1. The ingress **codec-policy** allows the telephone-event to have a different clock rate than the audio codec. Therefore, the first offer contains SILK/16000 and telephone-event/8000.
2. The egress **codec-policy** removes SILK, which is not supported. This policy adds PCMU/8000 and telephone-event/24000 and sends out the offer to UAC2.

> **Note:**
>
> The egress side allows the audio and telephone-event to use different clock rates. Therefore, the system does not remove telephone-event/8000 from the offer. Since telephone-event/8000 is already present in offer, the system does not add the configured telephone-event::8k (based on codec policy).

3. The egress side generates an answer, selects the topmost telephone-event (8000) and sends the answer towards the ingress containing a telephone-event using the same clock rate that was originally offered (telephone-event/8000).
4. The system generates RFC 2833 packets towards the ingress and egress, based on the audio codec's clock rate, which is 16000 towards the ingress and 8000 towards the egress.

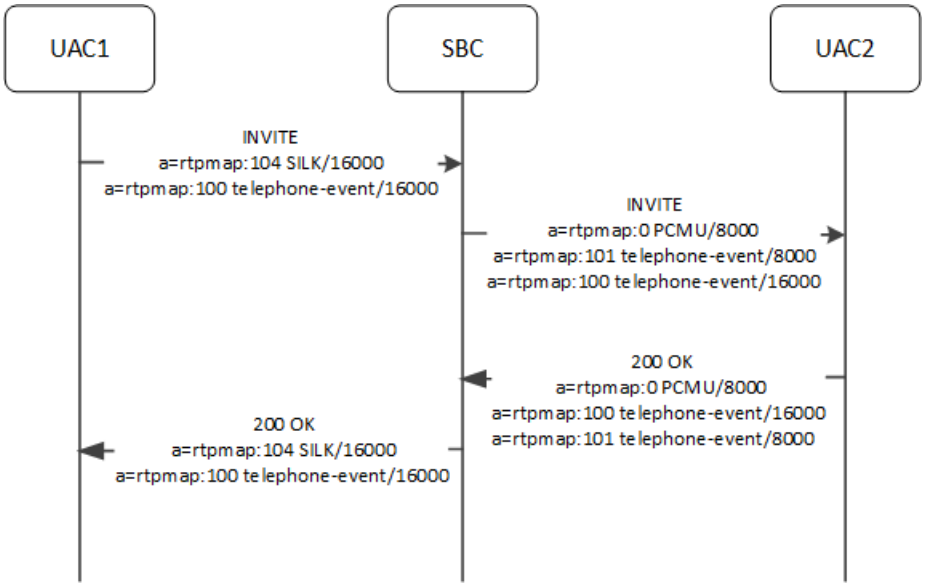
Results of this call flow include the ingress **codec-policy** allowing SILK and telephone-event. The egress **codec-policy** does not allow SILK, but adds PCMU, telephone-event::8k and telephone-event::24k;.

Example 4

In this example, you have configured:

- **allow-diff2833-clock-rate-mode** to **use-codec-clock-rate** on the ingress and egress **sip-interface**
- Ingress **codec-policy** allows SILK and telephone-event
- Egress **codec-policy** does not accept SILK, but adds PCMU, telephone-event::8k and telephone-event::16k

In addition, you have received multiple telephone-events in the answer.



The call flow proceeds as follows:

1. The ingress **codec-policy** allows the telephone-event to have a different clock rate than the audio codec. The first offer contains SILK/16000 and telephone-event/16000.
2. The egress **codec-policy** removes SILK, which is not supported. This policy also adds PCMU/8000 and the corresponding telephone-event/8000 and sends out the offer to UAC2.

> **Note:**
>
> The egress side allows the audio and telephone-event to use different clock rates. Therefore, the system does not remove telephone-event/16000 from the offer. Since telephone-event/16000 is already present in the offer, the system does not add hence telephone-event::16k (based on codec policy) again.

3. The egress side generates an answer, selects the topmost telephone-event, which is 16000, and sends the answer towards ingress containing a telephone-event that uses the same clock rate originally offered (telephone-event/16000).

The results of this call flow has the system generating ingress rfc2833 packets based on audio codec's clock rate (16000), and generating rfc2833 packets using the telephone-event clock rate (16000) towards the egress.

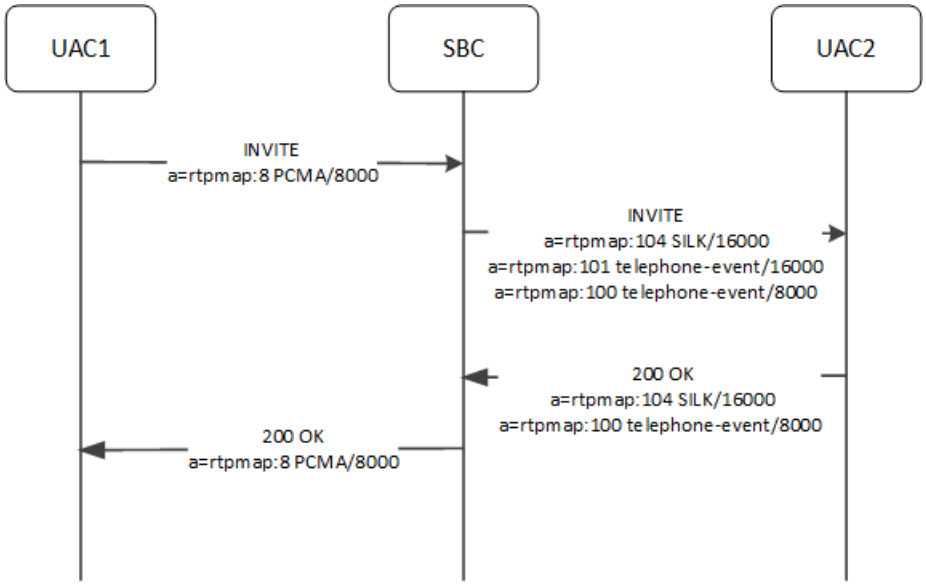
Example 5

In this example, you have configured:

- **allow-diff2833-clock-rate-mode** to **use-codec-clock-rate** on the ingress and egress **sip-interface**
- Ingress codec policy allows PCMA and telephone-event
- Egress codec-policy does not allow PCMA, but adds SILK-wideband, telephone-event::8k

In this example, there is no telephone event and, therefore, no telephone-event clock rate from the caller. The SBC, however, does need to negotiate telephone-events with UAC2. This example depicts the egress using your clock rate mode configuration on the egress to support different clock rates between itself and UAC2 independent of, and not triggered by the caller's signaling.

Here, your configuration uses the **allow-diff2833-clock-rate-mode** setting to support different RTP and telephone event clock rates on the egress. In addition, the egress **codec-policy** does not allow PCMA, generating transcoding on the call.



The call flow proceeds as follows:

1. The Ingress offer contains PCMA/8000, but does not contain a telephone-event.
2. The SBC egress **codec-policy** removes PCMA, which is not supported. It adds SILK/16000, a corresponding telephone-event/16000 and a telephone-event/8000 (configured in codec policy), then sends out the offer to UAC2.
3. The egress side generates an answer and sends the answer towards the ingress using PCMA/8000.

> **Note:**
>
> This call flow depicts an inband/Rfc2833 or SIP-INFO/Rfc2833 DTMF interworking case. If you set **rfc2833-mode** to **preferred** at the egress and **transparent** at the ingress, the SBC generates SIP-INFO packets towards the ingress side.

This configuration and flow ultimately includes the egress **codec-policy** replacing PCMA with SILK-wideband at 16k, and supporting telephone-events at 8k, thereby supporting different clock rates on the egress only.

Supporting Different Codec and Telephone-Event Rates in the SDP

This procedure shows you how to configure the SBC to use different clock for codec and telephone-events when the SDP offer presents rates that are not the same. If **allow-diff2833-clock-rate-mode** is disabled, the SBC does not support differing clock rates. When enabled, the SBC can use different rates.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access those configuration elements.

   ```
   ORACLE(configure)#session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **allow-diff2833-clock-rate-mode**—Enable this value to specify whether or not the SBC can present SDP that contains a telephone-event clock rate that is not the same as the audio codec clock rate. When this parameter is disabled, the SBC does not send telephone-event with a different clock rate than audio codec as an answer towards the ingress.

   Values include

   - **use-2833-clock-rate**—Allow the use of different clock rates and generate RFC2833 packets using the telephone-event clock rate.
   - **use-codec-clock-rate**—Allow the use of different clock rates and generate RFC2833 packets using the codec clock rate.

   The example below sets the value to **use-codec-clock-rate**.

   ```
   ORACLE(sip-interface)#allow-diff2833-clock-rate-mode use-codec-clock-rate
   ```

5. Type **done** to complete the configuration.
6. Save and activate your configuration.

Simultaneous Payload Type Mapping for Audio and DTMF

In addition to enabling audio payload type mapping for AMR and AMR-WB and enabling EVS AMR-WB IO payload type mapping, the **audio-payload-type-mapping** option, within the **media-manager**, configures the Oracle Communications Session Border Controller (SBC) to support simultaneous payload type mapping for audio and DTMF RFC-2833 for AMR, AMR - WB, and EVS in AMR wideband IO mode. Payload type mapping requires fully compatible SDP, with the exception of the payload type number. Simultaneous audio and DTMF RFC 2833

payload type mapping also requires that the payload type numbers for audio and DTMF be different.

ACLI Configuration

To enable simultaneous payload type mapping of audio and DTMF:

1. Navigate to the media-manager-config element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# media-manager-config
ACMEPACKET(media-manager-config)#
```

2. options—Set the options parameter by typing options, a Space, the option-name audio-payload-type-mapping=yes with a “plus” sign in front of it, and then press Enter.

```
ACMEPACKET(media-manager-config)# options +audio-payload-type-mapping=yes
```

   If you type the option without the “plus” sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration’s options list, you must prepend the new option with a “plus” sign as shown in the previous example.

3. Save and activate your configuration.

DTMF Indication over HD Audio Codecs

When performing DTMF transcoding while HD Audio codecs are present, Oracle Communications Session Border Controller accounts for `telephone-event` tone indication at clock rates that match those of the HD audio codecs.

The `telephone-event` tone indication's clock rate, when sent alongside an HD codec, must match the audio codec's clock rate. While many non-HD codecs use 8000 Hz clock rates, HD codecs can use clock rates of 16000 Hz. Transcoding processing takes these two `telephone-event` clock rates into account when performing SDP manipulation. If the wrong clock rate is used, RFC2833 `telephone-event` indications may be relayed incorrectly.

On the ingress side of the call, if a `telephone-event` is received, and no audio codec with a matching clock rate is received, the ingress-side SDP response will have the unmatched `telephone-event` removed.

If the **allow-codec** parameter uses the `:no` tag to remove the last of an audio codec that matches a `telephone-event` (8000 or 16000) from SDP, then the `telephone-event` of that clock rate (if present) will be removed from the SDP too.

When codec policy dictates to add AMR-WB, and the received SDP contains PCMU/PCMA and `telephone-event` (8000), the SDP offer sent from the egress interface will include `telephone-event` (16000)

If **rfc2833-mode** is set to **preferred**, the Oracle Communications Session Border Controller adds `telephone-event` 16000 to outbound SDP if AMR-WB is present in the outbound SDP.

On the egress side of the call if the SDP contains a `telephone-event` without an audio codec with a matching clock rate, an appropriate audio codec will be added. If codec policy adds a `telephone-event`, then the SBC analyzes the audio codecs in the outbound SDP and ensures that matching telephone events (8000 and/or 16000) are present.

Once the Oracle Communications Session Border Controller determines the SDP to forward, the order of `telephone-event` clock rates will be modified to match the order of audio codec rates. Thus if AMR-WB is the top codec followed by codecs with 8000 Hz clock rates, the telephone-event with a clock rate of 16000 Hz will be listed above telephone-events with 8000 Hz clock rates.

FAX Detection

In some deployments, an originator sends inband fax messages through the Oracle Communications Session Border Controller (SBC) to terminating endpoints that do not support uncompressed codecs. Thus the terminating call leg must communicate FAXes either through out of band T.38 or in-band G.711 codecs. In some cases the terminating endpoint can determine that it is being sent a FAX and send a reINVITE to request that it be sent T.38 FAX instead of inband FAX, thereby switching from an audio call to a FAX call. If the SBC does not receive this reINVITE, it will send its own reINVITE toward the terminating endpoint to establish the FAX session with a codec the endpoint can support.

The SBC can detect FAX tones based on the receipt of the DIS Preamble V.21 flag or the CNG/CED tones. These tones are embedded in a faxable codec in the RTP media stream. You must set the **tone-detection** parameter in the codec policy to **fax-cng** or **fax-v21** as necessary.

> **Note:**
>
> Enabling tone-detection steers all faxable calls through the DSPs thus requiring DSP resources.

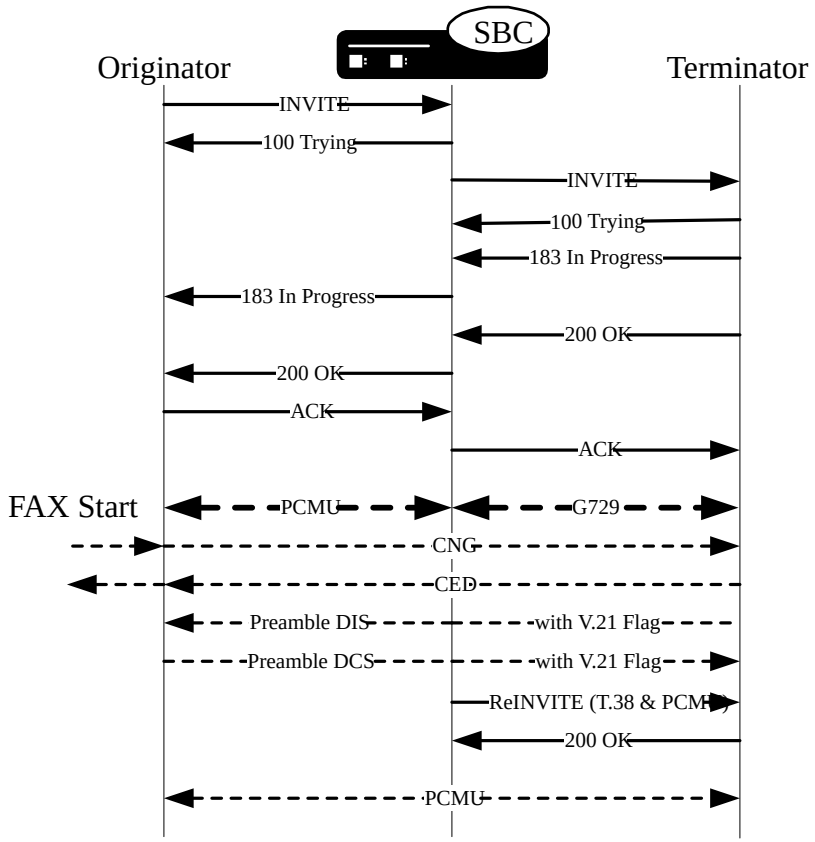
Renegotiate Timer

Upon detection of the DIS Preamble V.21 flag or CNG tone, the SBC starts a configurable tone Detect Renegotiate Timer (**tone-detect-renegotiate-timer**). The tone detect renegotiate timer parameter is configurable per codec policy and defaults to 500ms. If the SBC receives a ReINVITE from the originating or terminating endpoint, the tone Detect Renegotiate Timer will be reset and no ReINVITE will be sent from the SBC toward the terminating endpoint.

If the SBC does not receive a ReINVITE from the called endpoint before this timer expires, it creates and sends an INVITE to the terminating endpoint.

ReINVITE Codecs

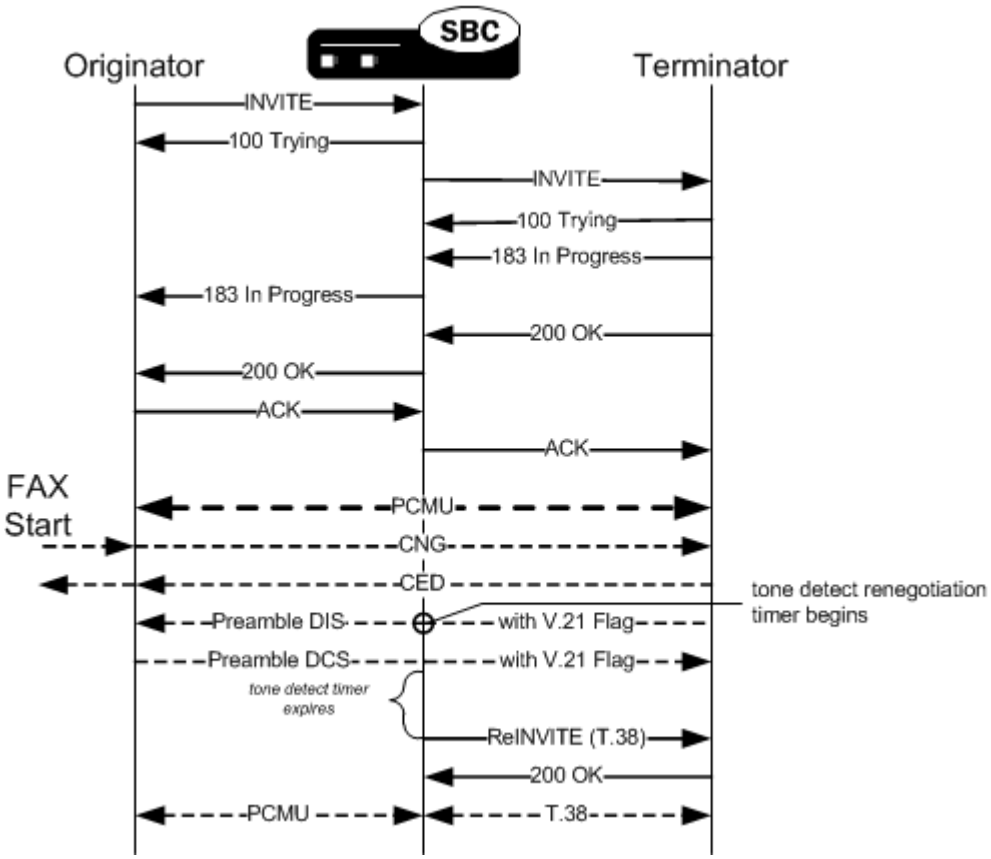
There are two codecs that can be inserted into the reINVITE message: T.38 & G711 (defaulting to PCMU). You configure **T.38OFD** and/or **G711OFD** in the add codecs on egress parameter in the codec policy configuration element. These codec names are only used for reINVITEs in this feature. Each one is inserted into the reINVITE's message body on its own m= line. The SBC sends 2 m-lines so the endpoint can pick its preferred codec from the two.



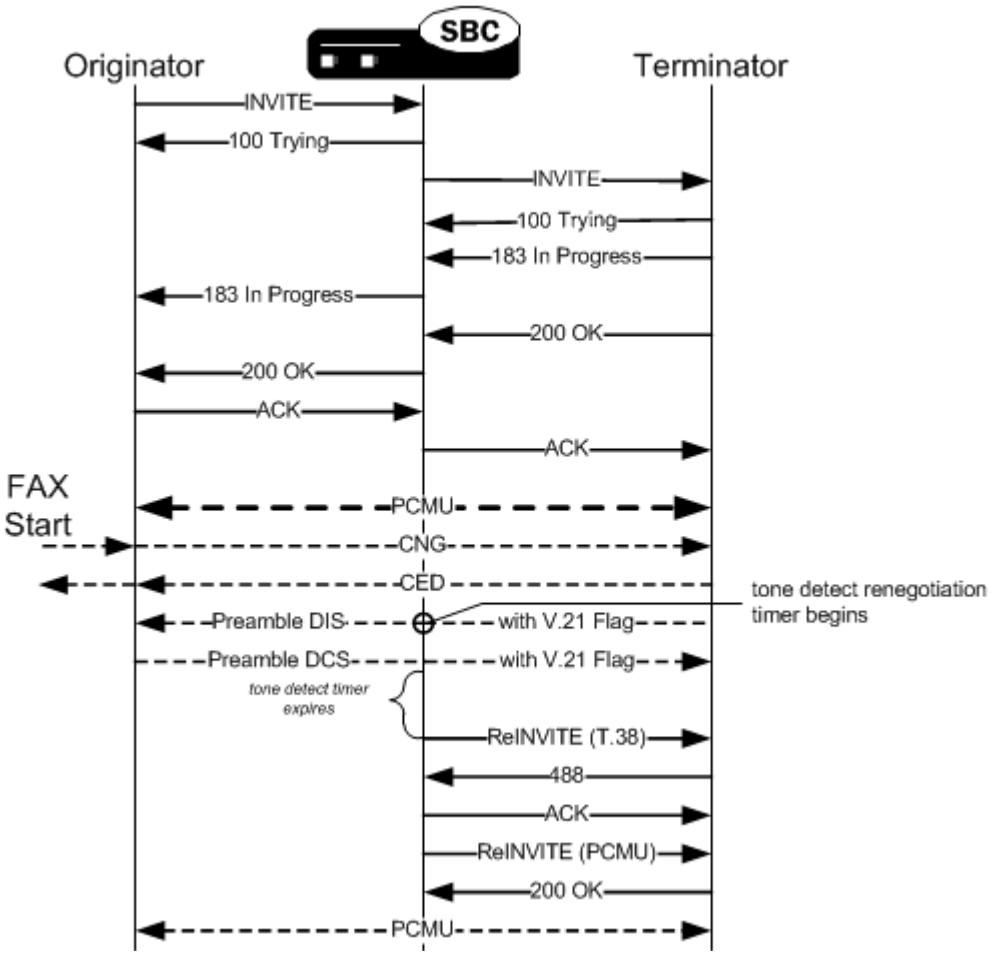
Call Completion

In the typical case, the SBC sends a reINVITE toward the terminating endpoint with the specified codecs. The terminating endpoint replies with a 200OK indicating it can use the specified codec. When this call leg is set up, FAX media will flow through the SBC and be transcoded between the originator's codec and the terminator's codec.

The following call flow shows the typical example:



When the terminating endpoint rejects the T.38 reINVITE, and the OFDFB is configured as an add-on-egress codec, the SBC sends a G711 offer toward the call terminator.



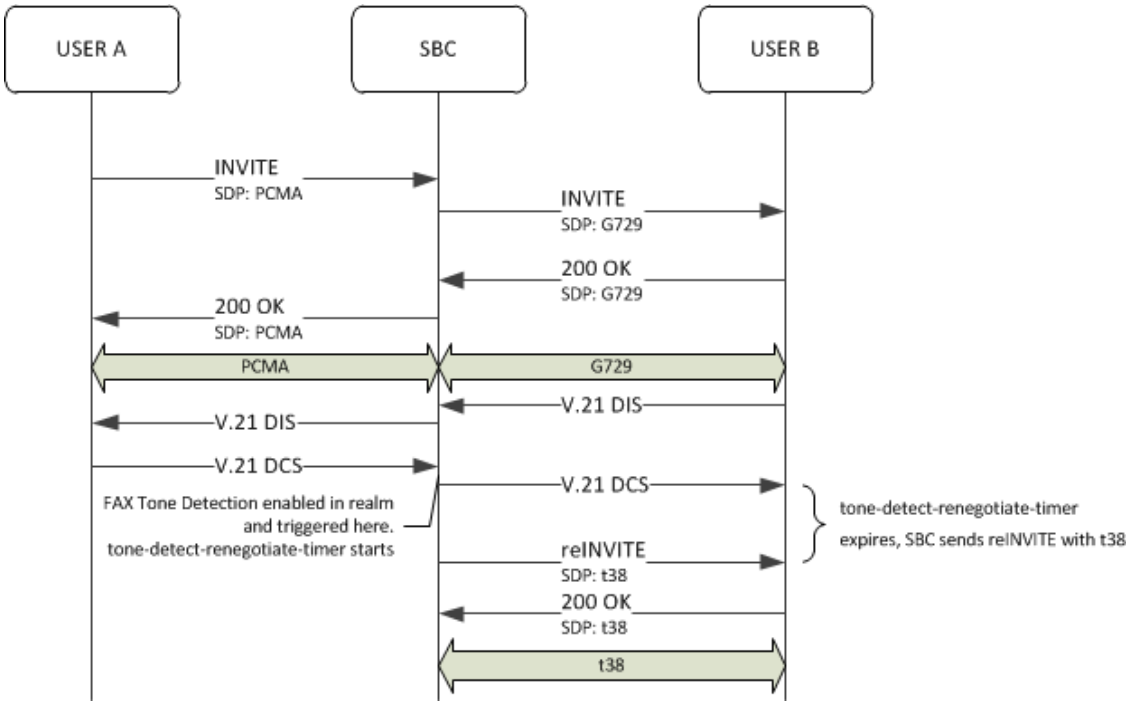
Glare Condition

After not receiving a ReINVITE from either of the endpoints after the configured amount of time, the SBC sends a reINVITE toward the terminating endpoint. If the SBC receives a ReINVITE from either endpoints while waiting for the response to its own reINVITE, it will reject the endpoint-sourced reINVITE with a 491 Request Pending error message.

reINVITE Toward Caller Using Compressed Codec

In some scenarios, the network operator configures tone detection in one realm where uncompressed codecs are used by UAs. The other realm, where compressed codecs are used does not have tone detection enabled. After the call is set up, the compressed side initiates a FAX call. The SBC receives user A's reply which includes V.21 DCS in the realm where tone detection is enabled. The SBC forwards this tone to user B. Upon no response from user B, the SBC creates and sends it a reINVITE that includes T.38 in the SDP. This action prompts User B to accept and use T.38 so that theSBC can transcode from User A's FAX via PCMA to User B's T.38 codec.

The pertinent aspect of this scenario is that the reINVITE is created and sent into the realm where tone-detection parameter is not configured. This behavior is enabled by enabling the **reverse-fax-tone-detection-reinvite** parameter. For example:



Supporting FAX to UAs that Do Not Support Multiple SDP M-Lines

The Oracle Communications Session Border Controller (SBC) sometimes supports FAX transcoding scenarios using a Re-INVITE that includes two m-lines in the SDP. Some end stations, however, do not support multiple m-lines, causing the FAX setup to fail. You can configure the SBC to resolve this problem on a per realm basis via transcoding policy.

There are two scenarios within which the SBC supports FAX setup by issuing a ReINVITE to a caller (or callee):

- The caller (or callee) issues a ReINVITE indicating it wants to change the call to FAX.
- The caller (or callee) embeds FAX tones in the RTP stream.

For both scenarios, the two options for providing the FAX media type include audio or image. In both scenarios, the SBC may send a Re-INVITE that offers both media options in the SDP. This allows the caller (or callee) to negotiate media type with the SBC. In the former scenario, the caller's ReINVITE either offers a single media option in the SDP m-line, in which case the SBC adds the other, or the caller offers both options. Note that the SBC must forward a final response to the caller (or callee) in this scenario. In the latter scenario, the SBC simply recognizes the attempt to start in-line FAX and issues a ReINVITE to support that setup.

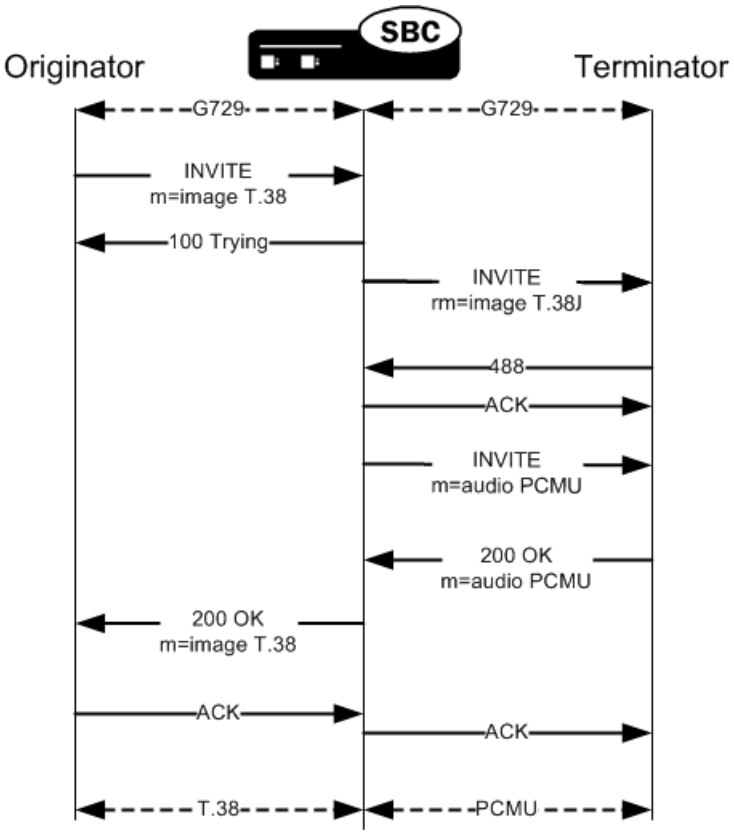
You can configure the SBC to remedy the lack of support for multiple m-lines in some stations by offering one media type at a time using the **codec-policy** element's **fax-single-m-line** parameter. When you configure this parameter, the SBC offers either audio or image media type in the ReINVITE. Should the callee reject the offer, with a **488 - Not Acceptable Here** message for example, the SBC sends another ReINVITE with the other media type choice. This negotiation may or may not result in a transcoded media stream. That is, if both end stations negotiate to the same codec, no transcoding is needed.

Example 1 - Offer Image First

This example depicts a transcoding scenario that changes a call to FAX. The initial Re-INVITE arrives offering image as the media type. The SBC issues its Re-INVITE with **fax-single-m-**

line set to **image-first**. The terminating station declines image media type, so the SBC retries, offering audio media type. The terminating station accepts this media type. Note that the SBC sends its response to the originating station, accepting image media type. Subsequently, the first leg operates with image media type and the second with audio, requiring transcoding.

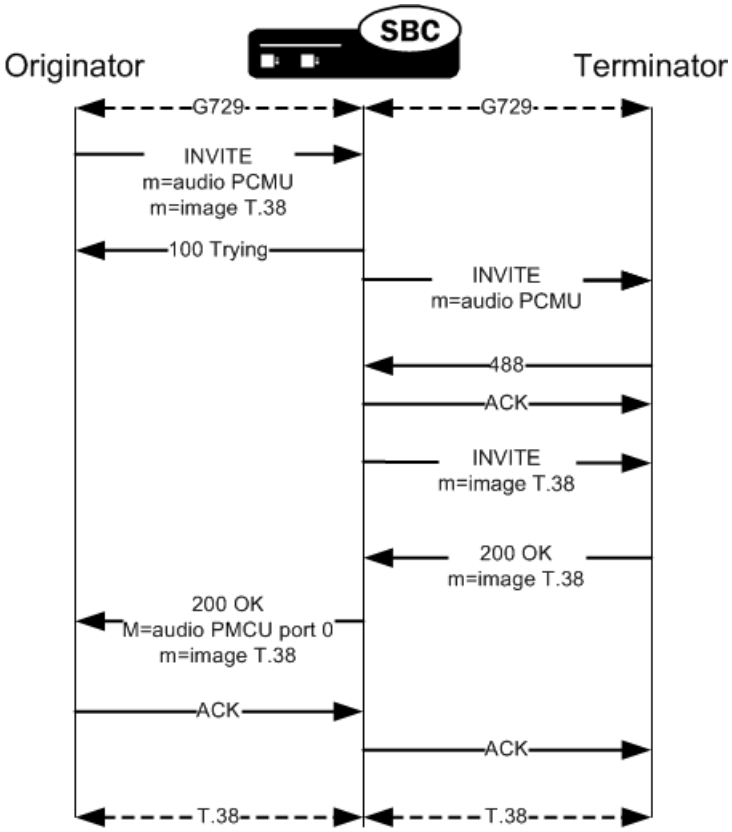
Figure 20-3 Offer Image First



Example 2 - Offer Audio First

This example depicts a transcoding scenario and configuration that changes a call to FAX. The initial Re-INVITE arrives with 2 m-lines. The SBC issues its Re-INVITE with **fax-single-m-line** set to **audio-first**. The terminating station declines audio media type, so the SBC retries, offering image media type. The terminating station accepts this media type. Note that the SBC sets its response to the originating station with audio media type set to port 0. Subsequently, both legs operate with image media type, requiring no transcoding.

Figure 20-4 Offer Audio First



FAX Detection and Redirect

You can configure the SBC to detect fax signaling within a SIP call and redirect those calls directly to a group of one or more fax servers. By default, the SBC sends a reINVITE either to a caller or calling party, based on your setting for the **reverse-fax-tone-detection-reinvite** parameter, when it detects a fax tone from the media stream. There are some call flows, however, that need redirection to the FAX endpoint without using this reINVITE. You can configure this support by setting the **fax-servers** parameter with the name of an applicable **session-agent-group** on the applicable **session-agent**. When enabled, the Fax Redirect feature takes precedence over the above mentioned legacy fax functionality.

> **Note:**
>
> This feature is dependent on specific DSP resources and, therefore, is not available when deployed on platforms that do not support transcoding. Refer to your version's release notes to determine which platforms apply.

Within the context of fax call flows that can use this feature, incoming audio/fax calls are often initially answered by an auto attendant or interactive voice response (IVR) service before going to a target fax server. Without configuring the **fax-servers** parameter, the SBC cannot send the fax transaction to this new endpoint. Instead, the SBC sends a reINVITE to the calling/caller side based on your **reverse-fax-tone-detection-reinvite** configuration.

To support fax re-direct, you configure:

- A **session-agent** object for each target fax server. Each **session-agent** should include:
  - A **realm-id** to ensure proper routing. The SBC does not consult **local-policy** to route the request, instead using the fax server's information on the IVR's **session-agent**.
  - A **codec-policy** that includes **add-codecs-on-egress** configured with G711OFD, T.38OFD or both.
- A **session-group** that includes:
  - The **dest** parameter configured with the names of each target fax server's **session-agent**.
  - **sag-recursion** enabled.
  - Your selected **strategy**.
- A **session-agent** for the IVR. This agent needs the name of the applicable **session-group** configured on the **fax-servers** parameter.
- A **codec-policy** on each **realm** that sends applicable fax traffic. These should include:
  - **add-codecs-on-egress** configured with G711OFD, T.38OFD or both.
  - **tone-detection** configured with CNG/CED.

  Use at least one of G711OFD/T.38OFD codecs in this policy to send an offer with faxable codecs to the fax server.

  > **Note:**
  >
  > If the policy has no fax codecs, this feature still sends egress offers to the fax server, but the faxes fail.

- A **codec-policy** on each applicable ingress realm that includes:
  - **tone-detection** configured with CNG/CED.
  - **add-codecs-on-egress** configured with G711OFD, T.38OFD or both.

If you enable **reverse-fax-tone-detection-reinvite** on the ingress **codec-policy**, you must have at least one tone detection codec (G711OFD/T.38OFD) configured on the egress **codec-policy** associated with the egress peer. If not, the SBC cannot detect fax tone.

For example, if you enable **reverse-fax-tone-detection-reinvite** on the **codec-policy** in the caller's realm, you must also configure the IVR session agent's **codec-policy** to add G711OFD, T.38OFD or both using the **add-codecs-on-egress** parameter .

Each **codec-policy** discussed above includes additional configuration that you can use to refine your media management of these flows. Consider additional **codec-policy** configuration, such as adding or removing specific codecs, to enhance your individual deployments.

With this feature configured and triggered by fax tone detection, the SBC:

1. Sends a SIP BYE message to terminate the session established with the IVR and waits for the 200 OK.
2. When it receives a 200 OK to the BYE, the SBC deletes both the east and west side flows to release the media resources consumed by the audio call. If there is no reply from IVR before the BYE transaction times out, the SBC clears the call resources.
3. The SBC does not send a BYE to the caller after terminating a call with an IVR.
4. The SBC selects a fax server from your **session-group** based on your configured selection criteria.

5. The SBC sends an INVITE with an SDP offer based on the codec policy configured on the egress realm of selected fax server.
6. When it receives a 200 OK response for the INVITE request sent to the fax server, the SBC creates a new client dialog. It then adds this client dialog to the call session, modifies media flows, and sends an ACK to the fax server.
7. The SBC sends a reINVITE towards the caller.

This reINVITE provides the caller with the new media IP/port allocated from the steering pool for the new media flows. Although the SBC uses the SDP originally negotiated with caller side, it updates the media IP/port and session version number in this SDP without consulting the egress **codec-policy** configured on caller's realm.

Ensuing SBC behavior is based the response from the caller for the reINVITE. If it receives:

- A 200 OK, the SBC replies with an ACK and begins fax media processing.
- No response, the SBC sends two BYEs, towards the fax server and the caller after the reINVITE transaction times out.
- A reINVITE/UPDATE from the caller while it is creating a session with a fax server, the SBC sends a 491 - Request Pending to the caller.
- If it receives a failure response (4xx/5xx/6xx), the SBC sends the ACK and then two BYEs, towards the caller and the fax server to clear both the call legs.

Regarding the selection of a target fax server, enabling **sag-recursion** on your fax group ensures the SBC recurses through your entire session-group. Otherwise, the SBC attempts to reach the first fax server only. When enabled, the SBC:

- Tries to reach the next fax server in your group if it receives 4xx or 5xx responses from its target.
- Terminates recursion through your group if it receives a 6xx response from any fax server in your group.
- Terminates recursion through your group if it receives any response configured in the **stop-sag-recurse** parameter.

Additional functionality includes:

- If it receives a 200 OK with a different SDP, the SBC sends an ACK, and then a BYE to the fax server. Furthermore, the SBC does not try any other server, instead sending a BYE to the caller with a reason header containing the message **Exhausted fax servers**.
- If it does not receive any response from a target, the SBC waits for the transaction timeout, then tries the next target in your **session-group**.
- If it receives a re-direct (3xx) response from the fax server, the SBC sequentially tries to INVITE the new address(es). If it does not receive a success response from any of these targets, the SBC resumes recursion through your **session-group**.
- If it receives no response, including no 100 Trying, from a specific fax server, the system retransmits this request. Once the transaction times out, the system sends an INVITE request to the next fax server in the list.
- If it receives no successful response from any target in your **session-group**, the SBC clears the transaction by sending a BYE with the reason header **Exhausted fax servers** to the caller.

Feature Interaction

This feature interacts with additional SBC configuration that affects its behavior, including:

- Transcoding—This feature uses transcoding to support G.711 to T.38 conversion and for detecting fax tones.
- PRACK-interworking—You can use this to handle/respond to any reliable provisional responses received from the fax-server.
- UPDATE-interworking—You can use this to convert any UPDATE received from the fax server to a re-INVITE towards the caller.
- **sag-recursion**—The SBC adheres to all recursion configuration affecting your **session-group**, including **stop-sag-recurse**, **sip-recursion-policy** and **stop-recurse**.
- multiple m-lines on fax codec—Disable the **fax-single-m-line** parameter in the applicable **codec-policy** to allow the SBC to initiate offers with both G711(PCMU) and T.38.
- **mm-in-realm**:
  - If enabled and the caller and fax-server are on the same realm, the SBC anchors the media flowing between caller and fax server.
  - If disabled and the caller and fax server are on the same realm, the SBC releases the media, allowing it to flow directly between the caller and the fax server.

Related Configuration Considerations

Note the following related configurations and their effect within the context of this feature:

- Ensure that you have the **fax-continue-session** parameter set to **none** (default) on both the ingress and egress realm's **sip-interface** elements.
- Ensure that you have the **fax-single-m-line** parameter set to **disabled** (default) on your **codec-policy**.
- If you have configured both G711OFD and T.38OFD in the egress **codec-policy**, the system adds an internal m-line when the SDP has only a single m-line. If the answerer selects this internal m-line, the fax fails.
- You can enable fax tone detection on both sides of your redirect configuration, caller and IVR. If so, you must consider your **reverse-fax-tone-detection-reinvite** setting. If you enable tone-detection on the IVR, and the SBC detects a fax tone (CNG), the SBC honors your **reverse-fax-tone-detection-reinvite** configuration on IVR's **codec-policy** and performs its default re-INVITE behavior instead of redirecting the call.

Reporting on FAX Group Statistics

The **show fax-group stats** command shows the number of successful and unsuccessful fax calls for all fax server groups or on a per-fax server group basis. The system rotates these statistics from the Period to the Lifetime section every 100 seconds

```
show fax-group stats | [fax-group name]
```

High Availability

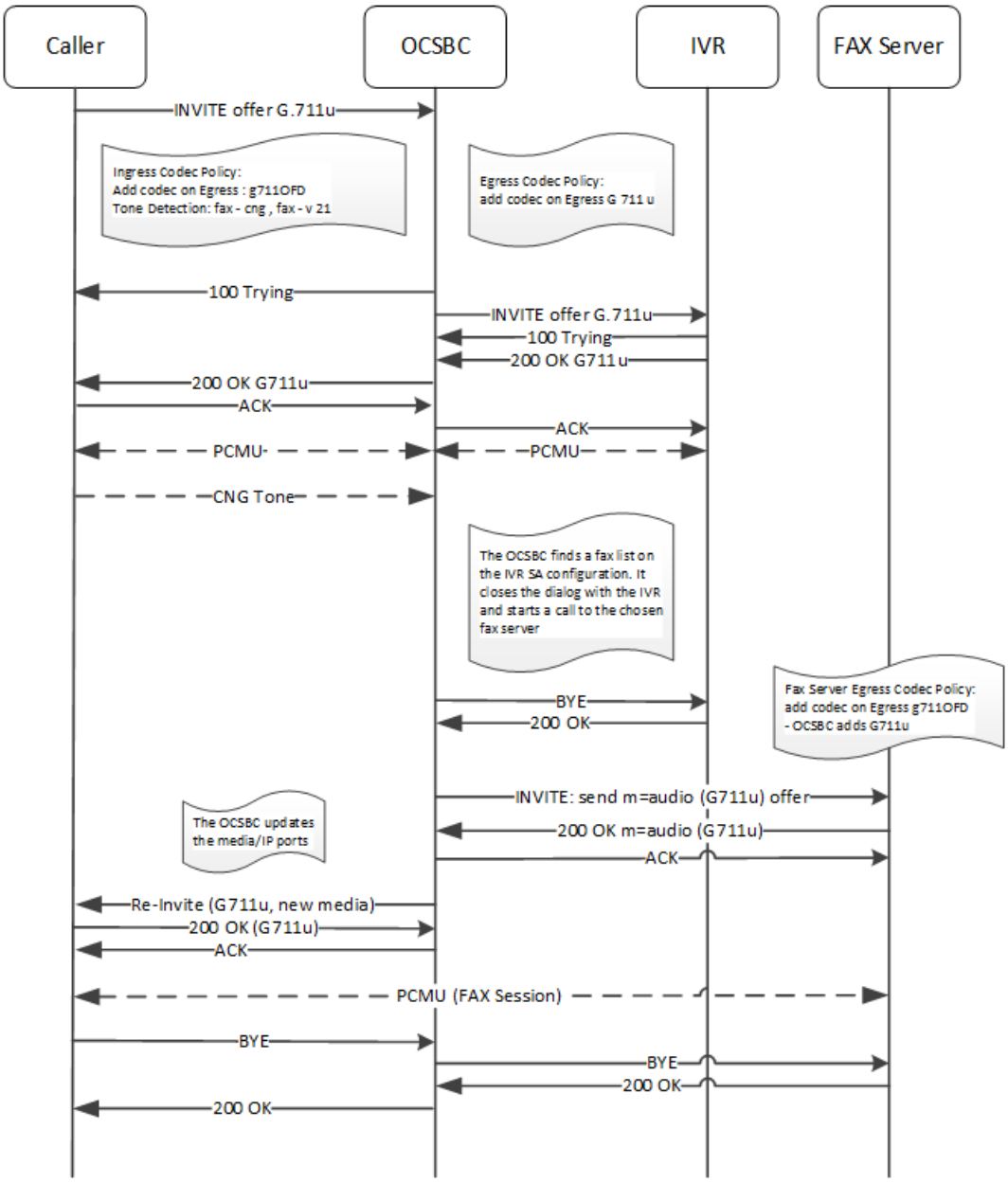
This feature fully supports HA deployments.

Call Flows for Fax with Redirect

This section presents call flows using the Fax with Redirect feature to display the signaling in greater detail.

Fax Redirect without Transcoding

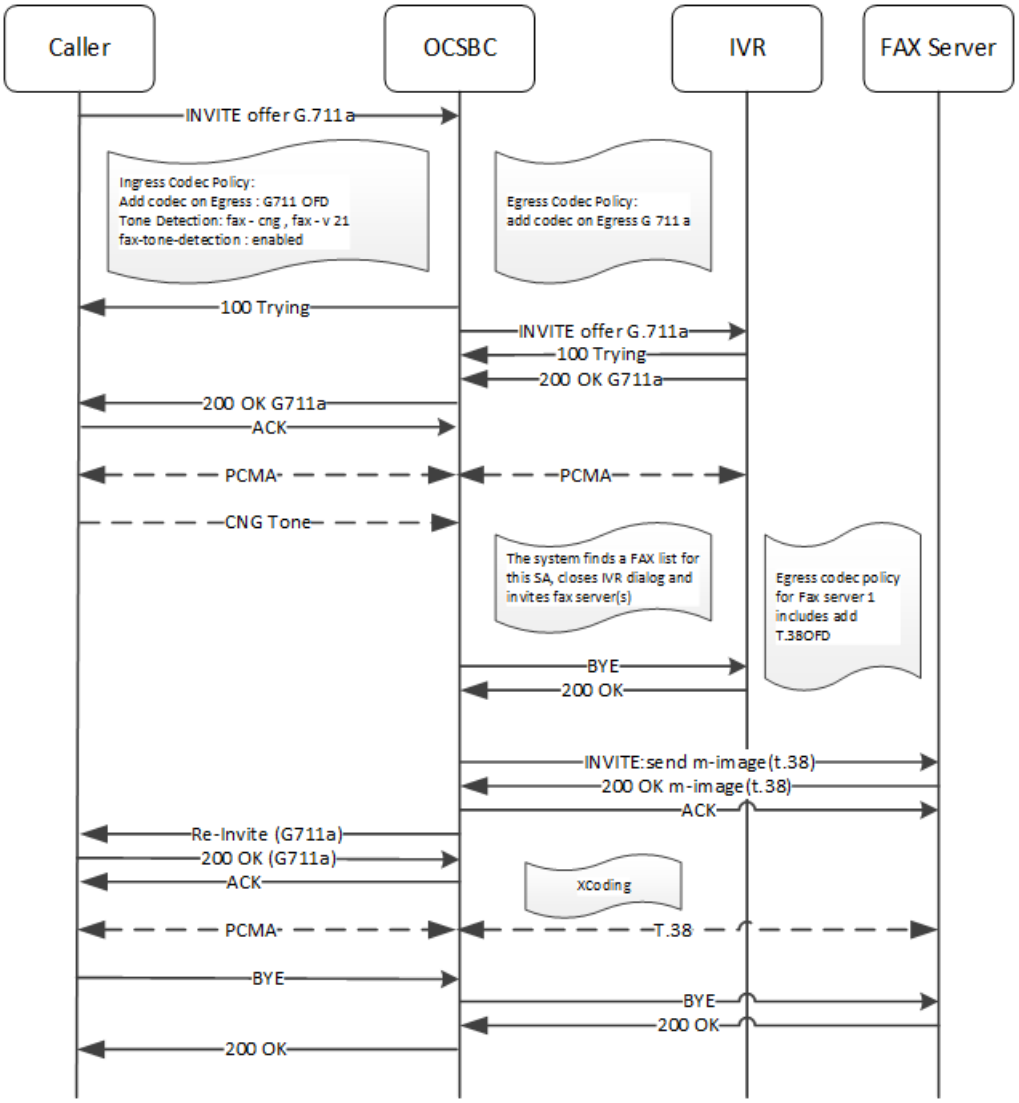
One of the simplest call flows for this FAX Handling with Redirect feature does not include transcoding of the fax media. The SBC performs the detect and redirect functions of this feature in this flow, with the results unaffected by any codec policy or manipulation.



Fax Redirect with Transcoding

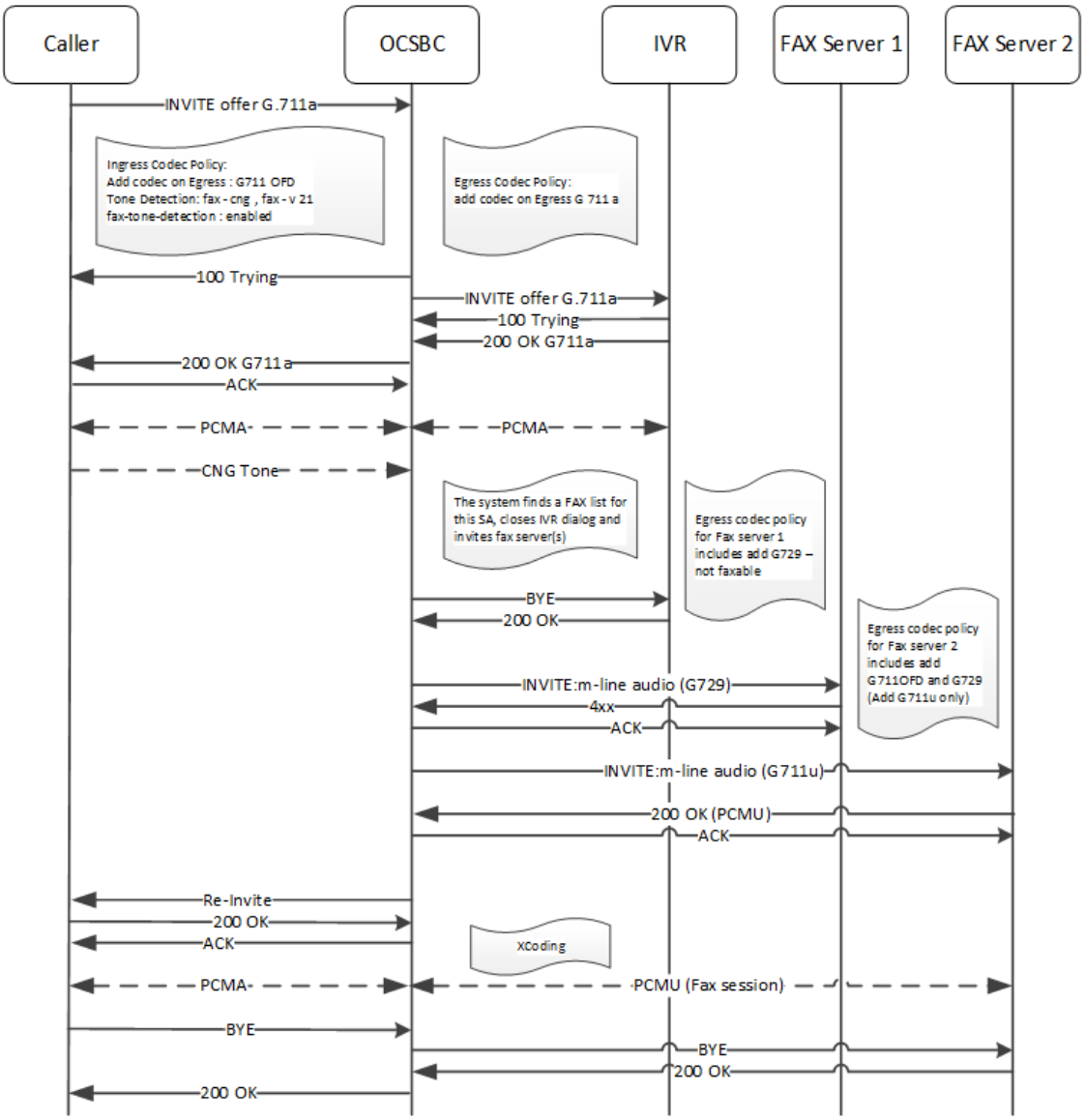
The next flow uses **codec-policy** configurations at several points within the SBC, including an ingress policy on the caller's realm, an egress policy on either the IVR realm or session agent configuration, and an egress policy on the selected fax server's session agent configuration.

After the initial signaling and fax detection, the SBC transcodes the fax media using PCMA on the caller side and T.38 on the fax server side.



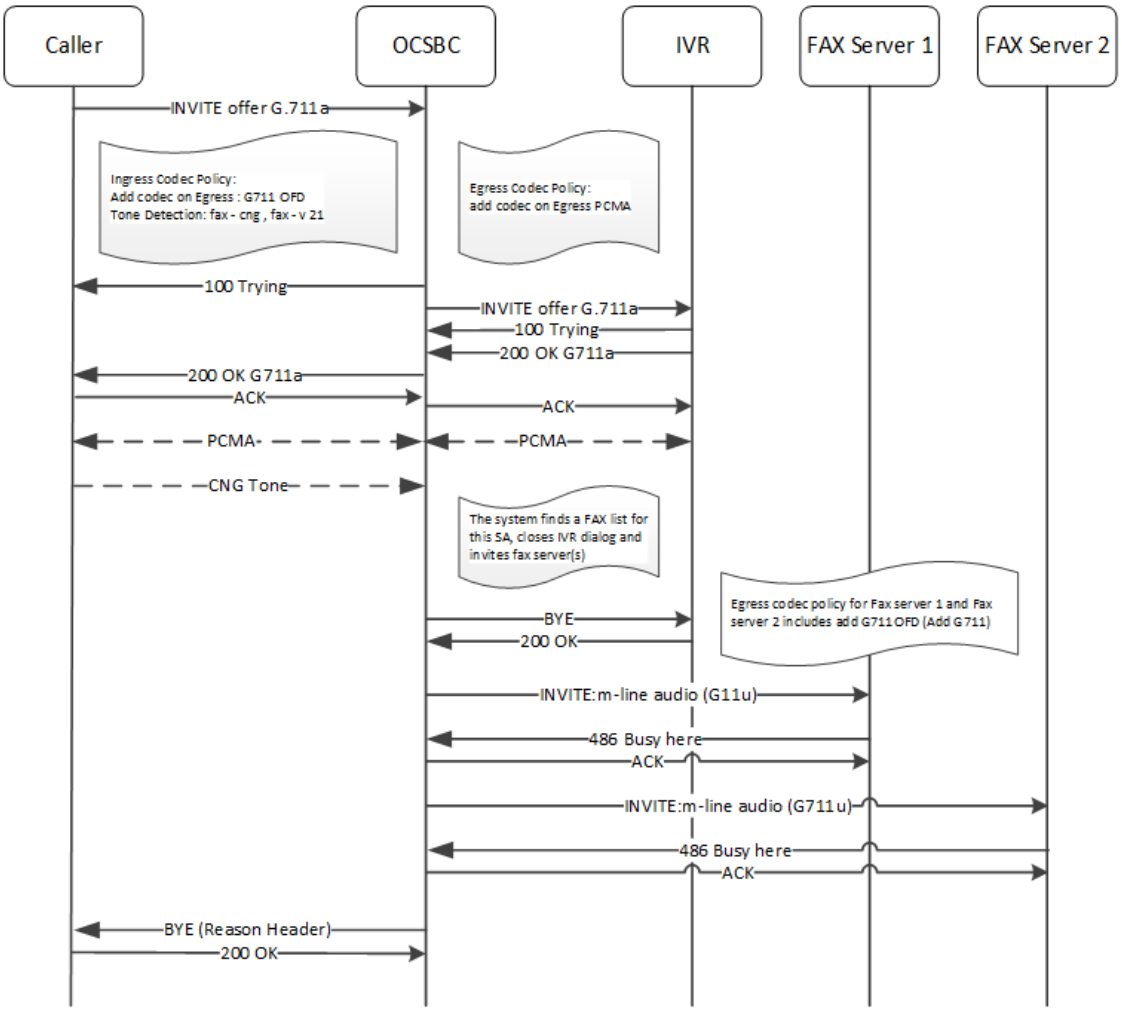
Fax Redirect with Codec Policy Error and Hunt

This next call flow depicts the SBC recursing through the fax group to find a second target. There can be many reasons to search the server list. Some issues end the call, but the issue in this flow, The session agent for FAX Server 1 has a codec policy that does not add a faxable codec to it, but the agent for FAX server 2 does. In this case, the error interrupts the flow, but the system continues with the call, ultimately succeeding with FAX Server 2.



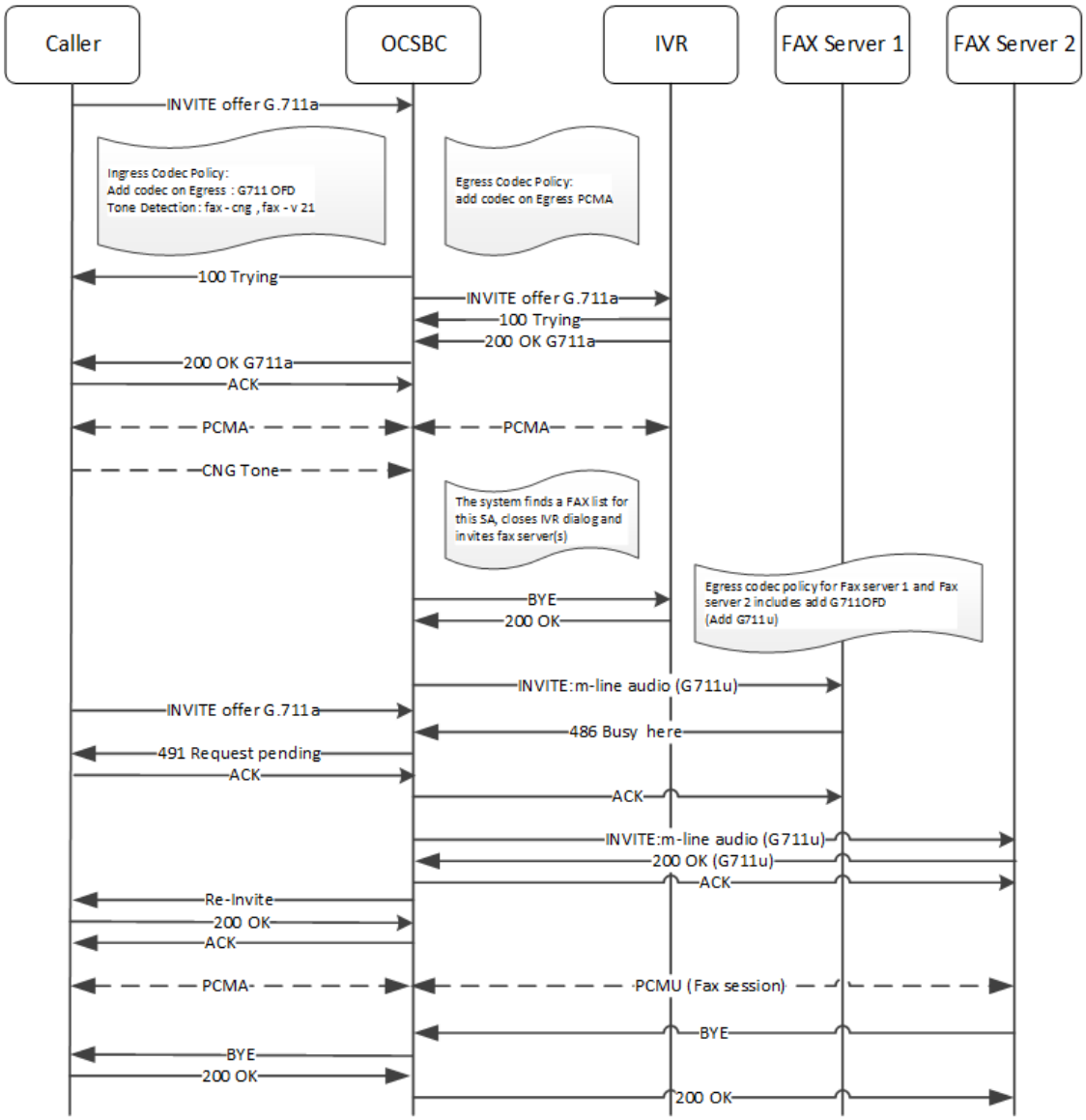
Fax Redirect Providing Reason Header to Caller

This next call flow depicts the SBC recursing through the fax group to find a second target, with both the first and second fax server responding that they are busy. The key functionality here is the SBC providing the caller with a reason header. The value of the reason header could be "exhausted fax servers", depending on your configuration.



Fax Redirect with ReINVITE During Progress

This next call flow depicts the SBC being interrupted by the caller with a ReINVITE. The SBC responds with “491 Request Pending” to this reINVITE.



Configure Fax Detect and Redirect

This copy presents task steps removed from their target location to make your review process easier.

Perform the following tasks before applying this configuration:

1. Configure a **codec-policy** to handle fax traffic.
2. Assign that **codec-policy** to the ingress realm. (This could also be on the caller's session-agent.)
3. Configure a **session-agent** for your target IVR.
4. Configure a **session-group** that lists your target fax machines.

1. Access your target IVR **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
```

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)# select
```

2. Select the **session-agent** you have configured for your target IVR.
3. Use the **fax-servers** parameter to specify the name of the session-agent-group

```
ORACLE(session-agent)# fax-servers SAG:MyFaxGroup
ORACLE(session-agent)#
```

The SBC uses your configured selection rules to determine which member of the group it sends the fax.

Configure reINVITE and Single M Line Behavior for FAX Calls

This procedure highlights the relevant parameters for enabling FAX Calls on reINVITE, and not necessarily all required parameters.

1. Access the **codec-policy** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# codec-policy
ORACLE(codec-policy)# select
```

2. Select the **codec-policy** object to edit.

```
ORACLE(codec-policy)# select
<name>:
1:  name=cp1
2:  name=cp2

selection: 2
ORACLE(codec-policy)#
```

3. **add-codecs-on-egress**—Set this parameter to either T.38OFD and/or G711OFD depending on the value the m= line should have on reINVITE. You can additionally add the OFDFB value to this parameter to enable the second fallback case.
4. **tone-detect-renegotiate-timer**—Set this parameter to a value between 50 and 3200 ms.
5. **tone-detection**—Set this parameter determine which message in FAX setup is used to start FAX transcoding.
   - fax-cng—start FAX transcoding on CNG message
   - fax-v21—start FAX transcoding on V.21 message
6. **reverse-fax-tone-detection-reinvite**—Set this parameter to enabled for the system to create a T38-ladden reINVITE and send it into the realm opposite of where tone detection is enabled.
7. **fax-single-m-line**—Set this parameter to the preferred FAX media type for Re-INVITEs to endstations that do not support multiple m-lines. The SBC issues Re-INVITEs using the configured media type only. Should the negotiation fail, the SBC issues another Re-INVITE that offers the other media type.
   - disabled—Do not change the SDP (default)
   - image_first—Sends Re-INVITE with m=image as the only m-line in the SDP.

- audio_first—Sends Re-INVITE with m=audio as the only m-line in the SDP.

8. Type **done** to save your configuration.

Maintenance and Troubleshooting

show mbcd errors

The **show mbcd errors** command displays statistics related to MBCD task errors. The following fields are explained:

- XCode Internal Errors—Number of uncategorized errors due to Transcoding session error.
- XCode Alloc Errors—Number of times that buffer allocation failed for transcoding tasks.
- XCode Update Errors—Number of errors encountered when attempting to update an entry in the Transcoding table upon receipt of the first packet for a media flow.
- XCode Delete Errors—Number of errors encountered when attempting to delete an entry in the Transcoding table.
- XCode Over Cap Errors—Number of Transcoding sessions denied once session capacity is reached.
- XCode Over License Cap—Number of Transcoding sessions denied once capacity is reached.

```
ORACLE# show mbcd errors
13:22:50-126
MBC Errors/Events                ---- Lifetime ----
                          Recent      Total   PerMax
Client Errors                  0          0        0
Client IPC Errors              0          0        0
Open Streams Failed            0          0        0
Drop Streams Failed            0          0        0
Exp Flow Events                0          0        0
Exp Flow Not Found             0          0        0
Transaction Timeouts           0          0        0
Server Errors                  0          0        0
Server IPC Errors              0          0        0
Flow Add Failed              180        180      180
Flow Delete Failed             0          0        0
Flow Update Failed             0          0        0
Flow Latch Failed              0          0        0
Pending Flow Expired           0          0        0
ARP Wait Errors                0          0        0
Exp CAM Not Found              0          0        0
Drop Unknown Exp Flow          0          0        0
Drop/Exp Flow Missing          0          0        0
Exp Notify Failed              0          0        0
Unacknowledged Notify          0          0        0
Invalid Realm                  0          0        0
No Ports Available             0          0        0
Insufficient Bandwidth         0          0        0
Stale Ports Reclaimed          0          0        0
Stale Flows Replaced           0          0        0
Telephone Events Gen           0          0        0
Pipe Alloc Errors              0          0        0
```

```
   Pipe Write Errors            0          0       0
   Not Found In Flows           0          0       0
   XCode Internal Errors        0          0       0
   XCode Alloc Errors           0          0       0
   XCode Update Errors          0          0       0
   XCode Delete Errors          0          0       0
   XCode Over Cap Errors      180        180     180
   XCode Over License Cap       0          0       0
   SRTP Capacity Exceeded       0          0       0
```

show xcode api-stats

The **show xcode api-stats** command shows the client and server side message counts for the XClient and XServer software components. The main messages are allocate, update, and free of the transcoding resource. The command uses a 100 second window to show recent counts within the sliding window as well as the total and per max (maximum in a sliding window interval). This command is useful for comparing the client and server side counts and seeing where errors may have occurred with the transcoding resources.

```
ORACLE#show xcode api-stats
                      --------- Client --------   --------- Server --------
Message/Event         Recent      Total  PerMax   Recent      Total  PerMax
                      ------  ---------  ------   ------  ---------  ------
Allocs                     0       5197    4897        0       6355    6055
Updates                    0       1776    1676        0        888     788
Frees                      0       6355    6015        0       6355    6015
Error-Allocs               0          0       0        0         45      45
Error-Updates              0          0       0        0        888     888
Error-Frees                0          0       0        0          0       0
Total                      0      13328   12588        0      14531   13791
```

show xcode dbginfo

The debug information command shows the packet API statistics for the host to DSP path. There is one session/connection opened with each DSP. The command displays the total packet counts as well as the round trip time statistics for the packets. The recent field shows the count since the last time the command was executed

```
ORACLE#show xcode dbginfo
Startup Time    : 2006-09-08 01:11:50.522
Last Clear Time : 2006-09-08 01:11:50.522
Last Read Time  : 2006-09-08 17:14:52.351
Current Time    : 2006-09-08 17:14:52.351
Up Time         : 0 Days, 16 Hours 3 Minutes 2 Seconds
                            -- Life Time --    -- Recent --
PktApiStats:
    OpenConnectionCnt    =       2
    OpenSessionCnt       =       2
    TotalPktSentCnt      =   21051              21051
    TotalPktRecvCnt      =   21041              21041
    TotalPktRecvEventCnt =       0                  0
    TotalPktRecvDataCnt  =       0                  0
    TotalPktRejectCnt    =       5                  5
    TotalPktTimeoutCnt   =       0                  0
```

```
TotalPktInvalidCnt    =       0                   0
TotalPktDropCnt       =       0                   0
TotalPktDropEventCnt  =       0                   0
TotalPktDropDataCnt   =       0                   0
TotalPktLateRspCnt    =       0                   0
LowestRoundTripMs              =       1
HighestRoundTripMs             =    2010
LowestExtractTimeMs            =       1
HighestExtractTimeMs           =   13320
HighestTransportRxTimeMs       =       0
ulHighestTransportNoRxTimeMs   =       0
```

Active and Period Statistics for EVRC and other Codecs

Codec use per-realm statistics include more detail by breaking out the EVRC codecs into their specific variants. That is, the system keeps track of EVRC0, EVRC1, EVRCB, EVRCB0, and EVRCB1 codec use per-realm on an explicit basis. These 5 counts are also available for SNMP query and are added to the `ap-codec.mib` file. Additionally, The Oracle Communications Session Border Controller now maintains counts for all codecs that appear in the **show sipd codecs <realm>** command for Active, Recent High, and Lifetime High periods.

show sipd codecs

The **show sipd codecs <realm ID>** command displays media-processing statistics per SIP traffic. This command displays statistics per realm and requires a realm argument.

Session Based Statistics

The first 3 statistics listed by the **show sipd codecs** are session based. These statistics are titled Transcoded, Transrated, and Transparent.

- transcoded—counts of sessions that use the Transcoding NIU's TCUs to transcode between two or more codes.
- transrated—counts of sessions that use the Transcoding NIU's TCUs to change the packetization interval among dialogs in the session.
- transparent—counts of sessions that require no TCU hardware intervention (all end-to-end media uses the same codec)

A value of "none" which is not counted in the statistics is set when there is no media at all or media is not yet negotiated. Sessions within the same realm are counted only once.

These are meter type counters, and thus have an "active" count as well as total lifetime values. The media-processing state of the session only can increase in precedence (highest=transcoded, transrated, transparent, lowest=none). Thus, if a session begins as transcoded, and then is re-negotiated to transparent later by a re-INVITE, it is still considered transcoded. However, if a session begins as transparent, it can go to transcoded by a re-INVITE. In such a case, the total counts for both transparent and transcoded would be incremented. If there are several media lines, the highest precedence is used for the session.

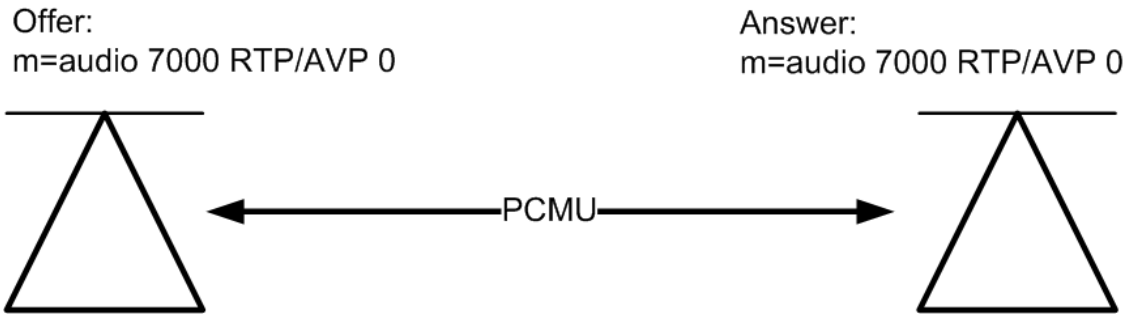
Flow Based Statistics

The remaining lines of the **show sipd codecs** command track the number of codecs in established sessions. The 'Other' type refers to unknown codecs. For all codecs listed by the **show sipd codecs** command, the Active, High- and Total- Recent Counts, and Total- PerMax-

High- Lifetime counts are displayed. These counts represent each SDP m= line emanating in the queried realm. Refer to the following examples:

Single audio stream example

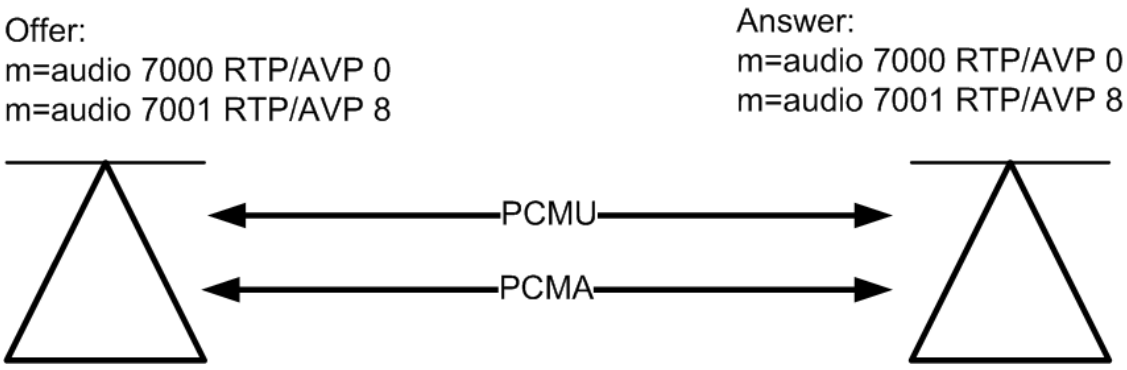
The following diagram shows an intra-realm session with one audio stream using the PCMU codec. Once the session is established, the PCMU count in the **show sidp codecs** output is 2.



If the session originator and terminator in the previous diagram exist in two different realms, you must execute the **show sidp codecs** command twice, once for each realm. A single PCMU count will be reflected in each respective query because only one m= line emanates from each realm.

Multiple audio stream example

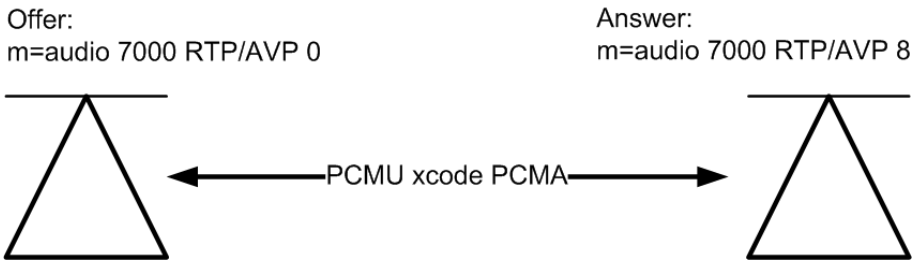
The following diagram shows an intra-realm session with two audio streams. Each stream uses a different codec. Once the session is established, the PCMU count in the **show sidp codecs** output is 2, and the PCMA count is 2.



If the session originator and terminator in the previous diagram exist in two different realms, you must execute the **show sidp codecs** command twice, once for each realm. A single PCMU count and a single PCMA count will be reflected in each respective query because two m= lines emanate from each realm.

Transcoded audio stream example

The following diagram shows an intra-realm transcoding scenario where the originator and terminator are using different audio codecs. The Oracle Communications Session Border Controller transcodes the media, which is invisible to the endpoints. Once the session is established, the PCMU count in the **show sidp codecs** output is 1, and the PCMA count is 1.



If the session originator and terminator in the previous diagram exist in two different realms, you must execute the **show sidp codecs** command twice, once for each realm. A single PCMU count appears in one query and a single PCMA count appears in the other query because only one m= line emanate from each realm.

An example **show sipd codecs <realm ID>** command follows:

```
ORACLE# show sipd codecs net172
09:20:04-30 Realm net172
Codec Statistics
                          ---- Recent ---- -------- Lifetime --------
                    Active     High    Total       Total   PerMax     High
Transcoded               0        0        0           0        0        0
Transrated               0        0        0           0        0        0
Transparent              0        0        0           0        0        0
PCMU Count               0        0        0           0        0        0
PCMA Count               0        0        0           0        0        0
G722 Count               0        0        0           0        0        0
G723 Count               0        0        0           0        0        0
G726-16 Count            0        0        0           0        0        0
G726-24 Count            0        0        0           0        0        0
G726-32 Count            0        0        0           0        0        0
G726-40 Count            0        0        0           0        0        0
G728 Count               0        0        0           0        0        0
G729 Count               0        0        0           0        0        0
GSM Count                0        0        0           0        0        0
iLBC Count               0        0        0           0        0        0
H261 Count               0        0        0           0        0        0
H263 Count               0        0        0           0        0        0
T38 Count                0        0        0           0        0        0
AMR Count                0        0        0           0        0        0
AMR-WB Count             0        0        0           0        0        0
EVRC Count               0        0        0           0        0        0
EVRC0 Count              0        0        0           0        0        0
EVRC1 Count              0        0        0           0        0        0
EVRCB Count              0        0        0           0        0        0
EVRCB0 Count             0        0        0           0        0        0
EVRCB1 Count             0        0        0           0        0        0
Other Count              0        0        0           0        0        0
```

show xcode load

The **show xcode load** command shows the current transcoding module (TCM) load both in number of sessions and percent loading. The load percentage depends on the precise mix of

codecs, ptimes, and features enabled on the active sessions. The maximum lifetime load is also displayed. Uninstalled TCMs are marked with dashes.

```
ORACLE#show xcode load -detail
02:00:16
Total Sessions:              0
Licensed AMR Sessions:       0
Licensed AMR-WB Sessions:    0
Licensed EVRC Sessions:      0
Licensed EVRCB Sessions:     0
Licensed OPUS Sessions:      0
Licensed SILK Sessions:      0
Licensed EVS Sessions:       0
                    ----- Load -----
  TCU  TCM  DSP  #Sess  Current  Maximum
  ===  ===  ===  =====  =======  =======
   0    00   0      0     0.00%   11.11%
   0    00   1      0     0.00%   11.11%
   0    01   0      0     0.00%   11.11%
[...]
```

The TCU column is populated with a 0 for a TCM in the middle slot and a 1 for a TCM in the top slot.

show xcode session-all

The **show xcode session-all** command displays all of the currently active sessions by their unique session id.

```
ORACLE#show xcode session-all
15:22:51
Requesting xclient sessions table
        Total Active Sessions: 200
        Displaying sessions 1 to 100:
        Session Id: 0x10007
        Session Id: 0x10008
        Session Id: 0x10009
        Session Id: 0x1000a
        Session Id: 0x1000b
```

> **Note:**
>
> When there are more than 100 active sessions , the command now displays only active sessions 1 to 100 as opposed to all the active session:

show xcode session-byid

The session-byid command gives more detailed information about the session. The session-byid command displays the configuration of each channel as well as a number of packet statistics for each channel. This same information can be looked up by IP address and port by

using the session-byip command. If only the configuration portion is required, use the session-config command with the session id as the argument. This command is entered as:

```
show xcode     session-byid <session_id>
```

For example:

```
ORACLE#show xcode session-byid 0xf006e
################ SESSION  0xf006e ##################
Channel 0:
  DSP device          = 14
  Source MAC          = 00:08:25:a0:9a:f3
  Destination MAC     = 00:0e:0c:b7:32:e2
  VLAN ID             = 0
  Egress Interface    = 0
  Src IP:Port         = 172.16.0.235:24448
  Dst IP:Port         = 172.16.0.87:16000
  Src RTCP IP:Port    = 172.16.0.235:24449
  Dst RTCP IP:Port    = 172.16.0.87:16001
  Codec               = G711_ULAW_PCM
  Payload Type        = 0
  Pkt Interval        = 20 msec
  2833 Payload Type   = DISABLED
  Xtone Mode          = XTONE_XTHRU
  Status              = DISABLED
DSP Counters:
  RxInPktCnt                  474
  RxInByteCnt                 75840
  RxOutPktCnt                 749
  RxInSidPktCnt               0
  RxNoPktCnt                  275
  RxBadPktTypeCnt             0
  RxBadRtpPayloadTypeCnt      0
  RxBadPktHdrFormatCnt        0
  RxBadPktLengthCnt           0
  RxMisorderedPktCnt          0
  RxBadPktChecksumCnt         0
  RxUnderrunSlipCnt           0
  RxOverrunSlipCnt            0
  RxLastVocoderType           0
  RxVocoderChangeCnt          0
  RxMaxDetectedPdv            168
  RxDecdrRate                 15
  RxJitter:
    CurrentDelay              160
    EstimatedDelay            0
    ClkDriftingDelta          0
    ClkDriftingCorrectionCnt  0
    InitializationCnt         1
  RxCircularBufferWriteErrCnt 0
  RxApiEventCnt               0
  TxCurrentVocoderType        0
  TxInPktCnt                  749
  TxOutPktCnt                 750
  TxOutByteCnt                120000
```

```
  TxInBadPktPayloadCnt        0
  TxTimestampGapCnt           0
  TxTdmWriteErrCnt            0
  RxToneDetectedCnt           0
  RxToneRelayEventPktCnt      0
  RxToneRelayUnsupportedCnt   0
  TxToneRelayEventPktCnt      0
  TxApiEventCnt               0
  TxNoRtpEntryPktDropCnt      0
  ConnectionWaitAckFlag       1
  RxMipsProtectionDropCnt     0
  TxMipsProtectionDropCnt     0
Channel 1:
  DSP device          = 14
  Source MAC          = 00:08:25:a0:9a:f4
  Destination MAC     = 00:1b:21:7a:29:b1
  VLAN ID             = 0
  Egress Interface    = 2
  Src IP:Port         = 192.168.0.235:24448
  Dst IP:Port         = 192.168.0.87:32000
  Src RTCP IP:Port    = 192.168.0.235:24449
  Dst RTCP IP:Port    = 192.168.0.87:32001
  Codec               = G729_A
  Payload Type        = 18
  Pkt Interval        = 20 msec
  2833 Payload Type   = DISABLED
  Xtone Mode          = XTONE_XTHRU
  Status              = DISABLED
DSP Counters:
  RxInPktCnt                  748
  RxInByteCnt                 14960
  RxOutPktCnt                 751
  RxInSidPktCnt               0
  RxNoPktCnt                  3
  RxBadPktTypeCnt             0
  RxBadRtpPayloadTypeCnt      0
  RxBadPktHdrFormatCnt        0
  RxBadPktLengthCnt           0
  RxMisorderedPktCnt          0
  RxBadPktChecksumCnt         0
  RxUnderrunSlipCnt           0
  RxOverrunSlipCnt            0
  RxLastVocoderType           6
  RxVocoderChangeCnt          0
  RxMaxDetectedPdv            171
  RxDecdrRate                 15
  RxJitter:
    CurrentDelay              160
    EstimatedDelay            0
    ClkDriftingDelta          0
    ClkDriftingCorrectionCnt  0
    InitializationCnt         1
  RxCircularBufferWriteErrCnt 0
  RxApiEventCnt               0
  TxCurrentVocoderType        6
  TxInPktCnt                  748
```

```
  TxOutPktCnt                748
  TxOutByteCnt               14960
  TxInBadPktPayloadCnt       0
  TxTimestampGapCnt          0
  TxTdmWriteErrCnt           0
  RxToneDetectedCnt          0
  RxToneRelayEventPktCnt     0
  RxToneRelayUnsupportedCnt  0
  TxToneRelayEventPktCnt     0
  TxApiEventCnt              0
  TxNoRtpEntryPktDropCnt     0
  ConnectionWaitAckFlag      0
  RxMipsProtectionDropCnt    0
  TxMipsProtectionDropCnt    0
```

show xcode session-byattr

The show xcode session-byattr command lists all sessions matching the specified attribute name. The only supported attribute is "fax", which will display session information only for FAX-transcoded sessions; all other attributes will return an error. For example:

```
ORACLE#show xcode session-byattr fax
17:52:17
1.   [Chan A_SRC] Ip Address: 192.168.16.1 Port: 5220
     [Chan B_SRC] Ip Address: 172.16.0.40 Port: 10230
     Session Id:0x002021d0"
2.   [Chan A_SRC] Ip Address: 192.168.16.1 Port: 3010
     [Chan B_SRC] Ip Address: 172.16.0.40 Port: 10226
     Session Id:0x00301042"
Oct 26 20:53:22.968 Total Matches:2 Total Active Sessions:2
-------------
```

show xcode session-byipp

The show xcode session-byipp command requires an IP address and port. It lists detailed information about the sessions identified by the specified IP address and port number. nformation will be provided for all transcoded call legs matching the IP address, including in both the ingress and egress directions. The show xcode session-byipp command is entered as:

```
show xcode session-byipp <ip_addr> <port_num>
```

This command displays the same information as the **session-byid** command. If a wildcard * is provided for the port number, the command will display sessions with the matching IP address only, regardless of port number.

show xcode xlist

The show xcode xlist command displays the TCU (0 = middle, 1 = top), TCM number, number of DSPs on each module, the number of active sessions, and the load percentage. It also displays the state such as Active or Boot Failure. Uninstalled TCMs are indicated by a dash.

```
ORACLE#show xcode xlist
18:22:32
  TCU  TCM  DSPs  #Sess  Load   State
  ===  ===  ====  =====  ====  =========
   0    00    2      -     -      -
   0    01    2      -     -      -
   0    02    2      -     -      -
   0    03    2      1     0%   2 Active
   0    04    2      1     0%   2 Active
   0    05    2      -     -      -
   0    06    2      -     -      -
   0    07    2      -     -      -
   [...]
   1    00    2      1     0%   2 Active
   1    01    2      0     0%   2 Active
   1    02    2      0     0%   2 Active
   1    03    2      0     0%   2 Active
   1    04    2      0     0%   2 Active
   1    05    2      0     0%   2 Active
   1    06    2      0     0%   2 Active
   1    07    2      0     0%   2 Active
   1    08    2      0     0%   2 Active
```

Logs

A log file named `log.xserv` can be used for debugging the transcoding feature. This log records the API between the host software and the DSPs and any errors that are encountered.

Alarms

The transcoding feature employs several hardware and software alarms to alert the user when the system is not functioning properly or overload conditions are reached.

Name/ID	Severity/ Health Degredation	Cause(s)	Traps Generated
No DSPs Present with Transcoding Feature Card (DSP_NONE_PRESENT)	Minor/0	A transcoding feature card is installed but no DSP modules are discovered.	apSysMgmtHardwareErrorTrap

Name/ID	Severity/ Health Degredation	Cause(s)	Traps Generated
DSP Boot Failure (DSP_BOOT_FAILURE)	Critical/0	A DSP device fails to boot properly at system initialization. This alarm is not health affecting for a single DSP boot failure. DSPs that fail to boot will remain uninitialized and will be avoided for transcoding.	apSysMgmtHardwareErrorTrap
DSP Communications Timeout (DSP_COMMS_TIMEOUT)	Critical/100	A DSP fails to respond after 2 seconds with 3 retry messages. This alarm is critical and is health affecting.	apSysMgmtHardwareErrorTrap
DSP Alerts (DSP_CORE_HALT)	Critical/100	A problem with the health of the DSP such as a halted DSP core. The software will attempt to reset the DSP and gather diagnostic information about the crash. This information will be saved in the /code directory to be retrieved by the user.	apSysMgmtHardwareErrorTrap
DSP Temperature(DSP_TEMPERATURE_HIGH)	Clear 85°C Warning 86°C / 5 Minor 90°C / 25 Major 95°C/ 50 Critical 100°C/ 100	A DSP device exceeds the temperature threshold. If the temperature exceeds 90°C, a minor alarm will be set. If it exceeds 95°C, a major alarm will be set. If it exceeds 100°C, a critical alarm will be set. The alarm is cleared if the temperature falls below 85°C. The alarm is health affecting.	apSysMgmtHardwareErrorTrap
Transcoding Capacity Threshold Alarm (XCODE_UTIL_OVER_THRESHOLD) / 131329	Clear 80% Warning 95%	A warning alarm will be raised when the transcoding capacity exceeds a high threshold of 95%. The alarm will be cleared after the capacity falls below a low threshold of 80%. This alarm warns the user that transcoding resources are nearly depleted. This alarm is not health affecting.	apSysMgmtGroupTrap

Name/ID	Severity/ Health Degredation	Cause(s)	Traps Generated
Licensed AMR Transcoding Capacity Threshold Alarm/ 131330	Clear 80% Warning 95%	A warning alarm is triggered if the AMR transcoding capacity exceeds a high threshold of 95% of provisioned session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap
Licensed AMR-WB Transcoding Capacity Threshold Alarm/ 131331	Clear 80% Warning 95%	A warning alarm is triggered if the AMR-WB transcoding capacity exceeds a high threshold of 95% of provisioned session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap
Licensed EVRC Transcoding Capacity Threshold Alarm/ 131332	Clear 80% Warning 95%	A warning alarm is triggered if the EVRC transcoding capacity exceeds a high threshold of 95% of provisioned session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap
Licensed EVRCB Transcoding Capacity Threshold Alarm/ 131333	Clear 80% Warning 95%	A warning alarm is triggered if the EVRCB transcoding capacity exceeds a high threshold of 95% of provisioned session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap

Transcoding Capacity Traps

The Oracle Communications Session Border Controller sends the apSysMgmtGroupTrap as transcoding capacity nears its limit. This trap is sent and cleared for three conditions:

- Total DSP usage exceeds 95%
- Total AMR sessions exceed 95% of provisioned capacity
- Total AMR-WB sessions exceed 95% of provisioned capacity
- Total EVRC sessions exceed 95% of provisioned capacity
- Total EVRCB sessions exceed 95% of provisioned capacity

The apSysMgmtGroupTrap contains the condition observed (apSysMgmtTrapType) and the corresponding value reached (apSysMgmtTrapValue).

```
apSysMgmtGroupTrap          NOTIFICATION-TYPE
    OBJECTS         { apSysMgmtTrapType, apSysMgmtTrapValue }
    STATUS            current
    DESCRIPTION
          " The trap will generated if value of the monitoring object
          exceeds a certain threshold. "
    ::= { apSystemManagementNotifications 1 }
```

When the resource usage retreats below a defined threshold, the Oracle Communications Session Border Controller sends an apSysMgmtGroupClearTrap.

```
apSysMgmtGroupClearTrap          NOTIFICATION-TYPE
    OBJECTS         { apSysMgmtTrapType }
    STATUS            current
    DESCRIPTION
          " The trap will generated if value of the monitoring object
          returns to within a certain threshold.  This signifies that
          an alarm caused by that monitoring object has been cleared. "
    ::= { apSystemManagementNotifications 2 }
```

The following table summarizes trigger and clear conditions for transcoding capacity alerts as sent in the the apSysMgmtGroupTrap:

Monitored Transcoding Resource	SNMP Object & OID in apSysMgmtTrapType	Trap Sent	Clear Trap Sent
Total DSP Usage	apSysXCodeCapacity 1.3.6.1.4.1.9148.3.2.1.1.34	95%	80%
AMR License Capacity Usage	apSysXCodeAMRCapacity 1.3.6.1.4.1.9148.3.2.1.1.35	95%	80%
AMR-WB License Capacity Usage	apSysXCodeAMRWBCapacity 1.3.6.1.4.1.9148.3.2.1.1.36	95%	80%
EVRC License Capacity Usage	apSysXCodeEVRCCapacity 1.3.6.1.4.1.9148.3.2.1.1.39	95%	80%
EVRCB License Capacity Usage	apSysXCodeEVRCBCapacity 1.3.6.1.4.1.9148.3.2.1.1.40	95%	80%

The following SNMP Objects are inserted into the apSysMgmtTrapType when sending and clearing a transcoding capacity trap. You mayt query them individually with an SNMP GET.

- apSysXCodeCapacity (1.3.6.1.4.1.9148.3.2.1.1.34)
- apSysXCodeAMRCapacity (1.3.6.1.4.1.9148.3.2.1.1.35)
- apSysXCodeAMRWBCapacity (1.3.6.1.4.1.9148.3.2.1.1.36)
- apSysXCodeEVRCCapacity (1.3.6.1.4.1.9148.3.2.1.1.39)
- apSysXCodeEVRCBCapacity(1.3.6.1.4.1.9148.3.2.1.1.40)

SRTP and Transcoding

Secure Real Time Transport Protocol (SRTP) allows secure media transmission. Transcoding is the ability to convert between media streams that are based upon different codecs. The Oracle Communications Session Border Controller supports IP-to-IP transcoding for SIP sessions and can connect two voice streams that use different coding algorithms with one another. Both SRTP and transcoding are available in the same call.

As of this release of the software, SRTP and transcoding may be combined for the same call. This behavior is available by default and no extra configuration is required.

Generating RTCP

The Oracle Communications Session Border Controller is capable of creating and sending RTCP reports using Transcoding resources. This produces RFC 3550 compliant RTCP report information on media traffic for active sessions. The system calculates these statistics using measurements on traffic between a target end station and itself. With respect to a given media session, the system does not produce end-to-end reports. In addition, the system can drop RTCP reports generated upstream so target stations don't receive RTCP reports that are redundant to its own.

RTCP reporting generated at the Oracle Communications Session Border Controller provides the following benefits:

- Provide RTCP reporting to core applications that may otherwise not be receiving this data.
- Provide RTCP reporting to access elements, including mobile client applications, that may otherwise not be receiving this data.
- Generate valid RTCP data for transcoded calls.
- Generate RTCP data from the Oracle Communications Session Border Controller's perspective.
- Applicable RTCP reports continue to be sent when a call is muted or placed on-hold.

The user enables and specifies the type of calls on which the system generates RTCP reports via configuration. Applicable configuration includes creating **rtcp-policy** objects and applying them to realms. These objects specify whether to issue reports for transcoded or all calls traversing the realm. There is also a configuration to disable the policy.

How it Works

The Oracle Communications Session Border Controller generates RTCP by sending RTCP sender report information to the media termination point within 5 seconds after an RTP session starts, per RFC 3550. All RTCP messaging includes Sender Reports and, if the end station is receiving media, statistics corresponding to the received media. This messaging persists, within 5-second windows for the duration of the session. The system sends a goodbye message no more than 20 ms after the call is complete. The network administrator should note this timing and ensure that network NAT configuration does not block these final messages. Depending on call type and configuration, the system listens for and drops RTCP reports generated upstream. RTCP reports generated prior to transcoding provide unreliable information.

All reports include the source description with a CNAME string. The CNAME string is the IP address and port number used for the corresponding media stream's RTCP in the format:

```
<UDP Local Port Num for RTCP>@<IP address of the applicable steering pool>
```

The system sources this data from the applicable steering pool configuration.

Functional Matrix - Configuration vs. Call Type

RTCP generation and any corresponding RTCP blocking is a function of the type of call (transcoded or non-transcoded), and configuration. Applicable **rtcp-policy** configuration includes specifying whether the system should generate RTCP for all calls, or for transcoded calls only.

The behavior is as follows:

- For Non-Transcoded Calls—Based on the RTCP-generation configuration:
  - **xcoded-calls-only**—No RTCP generated; existing RTCP passes
  - **all-calls**—RTCP generated; existing RTCP blocked
- For Transcoded Calls—RTCP generated; existing RTCP blocked for both configurations
- The RTCP generation setting includes a disabled setting (**none**) that is set by default. This specifies that the policy never generate RTCP. Realms configured with these disabled policy settings cause the system to pass existing RTCP for non-transcoded calls and block existing RTCP for transcoded calls.

When RTCP generation is enabled for non-transcoded calls using the **all-calls** setting, these calls must negotiate a codec that the Oracle Communications Session Border Controller can transcode. This ensures that the call utilizes the system's transcoding resources, which can then generate the RTCP.

> **Note:**
>
> A realm's **rtcp-policy** takes precedence over its **block-rtcp** setting. Specifically, if **block-rtcp** is disabled and the **rtcp-policy** is set to block, the system blocks existing RTCP.

RTCP Generation Platform Support

RTCP Generation requires transcoding resources as the call would be treated as a transcoding call. For transcoding resources, the top codec in the offer requires a license in virtual SBC. The following platforms require DSP hardware:

- Acme Packet 3950
- Acme Packet 4600
- Acme Packet 4900
- Acme Packet 6300
- Acme Packet 6350

Virtual SBCs (vSBCs) require one or more transcoding cores in their startup configuration.

When generating RTCP for a non-transcoded call using DSPs, the system processes the call as a transcoded, but uses a "null" transcoding methodology that sets the input codec equal to the output codec.

The system provides alarms to help you verify the status of the hardware. Refer to "Transcoding Troubleshooting and Maintenance" for more information about transcoding hardware status. Refer to the *Maintenance and Troubleshooting Guide* for information about interface hardware status.

Use the **show xcode xlist** command to verify the presence of transcoding hardware with DSP modules.

Configuring RTCP Generation

The procedure below provides the steps needed to configure RTCP generation on the Oracle Communications Session Border Controller.

Before proceeding, make sure you know whether your platform supports transcoding. As described in the section on RTCP generation, this can affect your configuration and operational results.

To have your Oracle Communications Session Border Controller generate RTCP traffic statistics reporting:

1. In Superuser mode, use the following command sequence to access the **rtcp-policy** element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# rtcp-policy
   ```

   From this point, you can configure a new RTCP policy or select an existing policy for editing.

2. **name**—Name your policy for use when applying it to a **realm-config**.

   ```
   ORACLE(rtcp-policy)# name report-allcalls
   ```

3. **rtcp-generate**—Enter the desired setting to generate RTCP. The effect of these settings is dependent on the platform and the presence of specific hardware on that platform. See the section on Generating RTCP in the ACLI Configuration Guide for these details.

   - **none**—Disables this policy.
   - **all-xcoded-calls**—The system generates RTCP report information only for the transcoded calls that pass through the realm.
   - **all-calls**—The system generates RTCP report information for all calls that pass through the realm.

   ```
   ORACLE(rtcp-policy)# rtcp-generate all-calls
   ```

4. Type **done** and **exit** to retain your configuration.
5. Access the realm to which you want to apply this rtcp-policy.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)# select
   ```

6. Apply the rtcp-policy to your realm.

```
ORACLE(realm-config)# rtcp-policy report-allcalls
```

7. Type **done** and exit configuration mode. Save and activate your configuration.

Obtaining System Information about RTCP Generation

The Oracle Communications Session Border Controller provides information about RTCP report generation within log files.

The **log.mbcd** file, when configured to capture at the DEBUG level, includes system messages related to invoking and firing **rtcp-policy** rules.

An example of applicable log information is provided below.

```
[XC] NatFlow::check_transcoding()
[XC]  SessionSetRtcpPorts for A: src=10001 dest=20001
[XC] NatFlow::check_transcoding() SessionSetRtcpOptions(A):
[XC]  RtcpPolicy "my_rtcp_policy"; rtcp-generate set to:
RTCP_GEN_MODE_ALL_CALLS
[XC]  Will generate RTCP with CNAME=192.160.1.100:10001
[XC] NatFlow::check_transcoding()
[XC]  SessionSetRtcpPorts for B: src=10001 dest=20001
[XC] NatFlow::check_transcoding() SessionSetRtcpOptions(B):
[XC]  RtcpPolicy "my_rtcp_policy"; rtcp-generate set to:
RTCP_GEN_MODE_ALL_CALLS
[XC]  Will generate RTCP with CNAME=172.16.1.100:10101
```

Error and non-error system messages related to RTCP generation can be found in **log.sipd**, **log.mbcd**, and **log.xserv**.

Forced RTCP Receiver Report Generation

When the Oracle Communications Session Border Controller (SBC) generates a Real-Time Control Protocol (RTCP) Sender Report using Digital Signalling Processors (DSP)s, the report includes blocks for both sender and receiver statistics in the same report per RFC 3550. When you want to generate a separate RTCP Receiver Report, for example to encapsulate the receiver statistics differently, add the `xcode-gratuitous-rtcp-report-generation` option in the media-manager configuration. After you add the RTCP Receiver Report generation option and reboot the SBC, the system runs RTCP Receiver Reports for all media sessions that generate RTCP from DSPs.

Only systems with transcoding resources can support RTCP Receiver Reports.

The SBC can send RTCP sender reports and receiver reports:

- for media streams.
- for media streams with IPv6 Media Bypass OFF.
- when the end point places a call on hold.
- when the system plays music on hold.
- for every secure call for every media stream the SBC receives and sends using TLS and SRTP.

- for every secure call for every media stream the SBC receives and sends using TLS and SRTP with IPv6 Media Bypass OFF.

You can configure `xcode-gratuitous-rtcp-report-generation` from the ACLI and the Web GUI.

Generate an RTCP Receiver Report

When you want to generate a Real-Time Transport Control Protocol (RTCP) Receiver Report separately from the default Sender-Receiver Report (RFC 3550), for example to encapsulate the receiver statistics differently, add the `xcode-gratuitous-rtcp-report-generation` option in the media-manager configuration. After you add the option and reboot the system, the SBC runs RTCP Receiver Reports for all media sessions that generate RTCP from DSPs.

When you add the `xcode-gratuitous-rtcp-report-generation` option, be sure to type the **+** character before the option. The **+** character appends the new option to the realm configuration's options list. Without the + character, the system overwrites any previously configured options.

1. Access the **media-manager-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

2. Type **select**, and press Enter.

   The system displays the (media-manager-config)# prompt.

3. Type **options**, insert a space, type the **+** character, and type **xcode-gratuitous-rtcp-report-generation**.

   ```
   ORACLE (media-manager-config)# options +xcode-gratuitous-rtcp-report-
   generation
   ```

4. Press Enter.
5. Type **done** to save your configuration.
6. Activate the configuration.
7. Reboot the system.

SNMP

The following sections have been removed from this document. Their content has been reformatted and appears in the MIB Reference guide in the *Codec and Transcoding MIB (ap-codec.mib)* section:

- SNMP Retrieval of Transcoding Statistics
- Acme Packet Codec and Transcoding MIB (ap-codec.mib)
- Acme Packet System Management MIB (ap-smgmt.mib)

Acme Packet Codec and Transcoding MIB (ap-codec.mib)

The following table describes the SNMP GET query names for the Oracle Codec and Transcoding MIB (ap-codec.mib).

- Object Identifier Name: apCodecMIBObjects (1.3.6.1.4.1.9148.3.7.1)
- Object Identifier Name: apCodecRealmStatsTable (1.3.6.1.4.1.9148.3.7.1.1)
- Object Identifier Name: apCodecRealmStatsEntry (1.3.6.1.4.1.9148.3.7.1.1.1)

SNMP GET Query Name	Object Identifier Name: Number	Description
apCodecRealmCountOther	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.1	Count of the SDP media streams received in the realm which negotiated to a codec not defined in this table.
apCodecRealmCountPCMU	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.2	Count of SDP media streams received in the realm which negotiated to the PCMU codec.
apCodecRealmCountPCMA	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.3	Count of SDP media streams received in the realm which negotiated to the PCMA codec.
apCodecRealmCountG722	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.4	Count of SDP media streams received in the realm which negotiated to the G722 codec.
apCodecRealmCountG723	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.5	Count of SDP media streams received in the realm which negotiated to the G723 codec.
apCodecRealmCountG726-16	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.6	Count of SDP media streams received in the realm which negotiated to the G726-16 codec.
apCodecRealmCountG726-24	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.7	Count of SDP media streams received in the realm which negotiated to the G726-24 codec.
apCodecRealmCountG726-32	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.8	Count of SDP media streams received in the realm which negotiated to the G726-32 codec.
apCodecRealmCountG726-40	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.9	Count of SDP media streams received in the realm which negotiated to the G726-40 codec.
apCodecRealmCountG728	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.10	Count of SDP media streams received in the realm which negotiated to the G728 codec.
apCodecRealmCountG729	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.11	Count of SDP media streams received in the realm which negotiated to the G729 codec.
apCodecRealmCountGSM	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.12	Count of SDP media streams received in the realm which negotiated to the GSM codec.
apCodecRealmCountILBC	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.13	Count of SDP media streams received in the realm which negotiated to the iLBC codec.

SNMP GET Query Name	Object Identifier Name: Number	Description
apCodecRealmCountAMR	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.14	Count of SDP media streams received in the realm which negotiated to the AMR codec.
apCodecRealmCountEVRC	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.15	Count of SDP media streams received in the realm which negotiated to the EVRC codec.
apCodecRealmCountH261	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.16	Count of SDP media streams received in the realm which negotiated to the H261 codec.
apCodecRealmCountH263	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.17	Count of SDP media streams received in the realm which negotiated to the H.263 codec.
apCodecRealmCountT38	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.18	Count of SDP media streams received in the realm which negotiated to the T.38 codec.
apCodecRealmCountAMRWB	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.19	Count of SDP media streams received in the realm which negotiated to the AMR-WB codec.
apCodecRealmCountEVRC0	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.20	Count of SDP media streams received in the realm which negotiated to the EVRC0 codec.
apCodecRealmCountEVRC1	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.21	Count of SDP media streams received in the realm which negotiated to the EVRC1 codec.
apCodecRealmCountEVRCB	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.22	Count of SDP media streams received in the realm which negotiated to the EVRCB codec.
apCodecRealmCountEVRCB0	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.23	Count of SDP media streams received in the realm which negotiated to the EVRCB0 codec.
apCodecRealmCountEVRCB1	apCopdecRealmStatsEntry 1.3.6.1.4.1.9148.3.7.1.1.1.24	Count of SDP media streams received in the realm which negotiated to the EVRCB1 codec.

- Object Identifier Name: apTranscodingMIBObjects (1.3.6.1.4.1.9148.3.7.2)
- Object Identifier Name: apCodecTranscodingRealmStatsTable (1.3.6.1.4.1.9148.3.7.2.1)
- Object Identifier Name: apTranscodingRealmStatsEntry (1.3.6.1.4.1.9148.3.7.2.1.1)

SNMP GET Query Name	Object Identifier Name: Number	Description
apCodecRealmSessionsTransparent	apCodecTranscodingRealmStatsEntry: 1.3.6.1.4.1.9148.3.7.2.1.1.1	Number of sessions in the realm that did not use any DSP resources for transcoding or transrating.
apCodecRealmSessionsTransrated	apCodecTranscodingRealmStatsEntry: 1.3.6.1.4.1.9148.3.7.2.1.1.2	Number of sessions in the realm that had a common codec but used DSP resources to modify packetization rate.

SNMP GET Query Name	Object Identifier Name: Number	Description
apCodecRealmSessionsTranscoded	apCodecTranscodingRealmStatsEntry: 1.3.6.1.4.1.9148.3.7.2.1.1.3	Number of sessions in the realm that had used DSP resources to transcode between codecs.

- Object Identifier Name: apSysMgmtMIBSessionObjects (1.3.6.1.4.1.9148.3.2.1.2)
- Object Identifier Name: apSigRealmStatsTable (1.3.6.1.4.1.9148.3.2.1.2.4)
- Object Identifier Name: apSigRealmStatsEntry (1.3.6.1.4.1.9148.3.2.1.2.4.1)

SNMP GET Query Name	Object Identifier Name: Number	Description
apSigRealmStatsRealmName	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.2	Nmae of the realm the following for which the following statistics are being calculated.
apSigRealmStatsCurrentActiveSessionsInbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.3	Number of current active inbouind sessions.
apSigRealmStatsCurrentSessionRateInbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.4	Current inbound session rate in CPS.
apSigRealmStatsCurrentActiveSessionsOutbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.5	Number of current active outbound sessions.
apSigRealmStatsCurrentSessionRateOutbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.6	Current outbound session rate in CPS.
apSigRealmStatsTotalSessionsInbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.7	Total number of inbound sessions.
apSigRealmStatsTotalSessionsNotAdmittedInbouind	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.8	Total number of inbound sessions rejected due to insufficient bandwidth.
apSigRealmStatsPeriodHighInbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.9	Highest number of concurrent inbound sessions during the period.
apSigRealmStatsAverageRateInbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.10	Average rate of inbound sessions during the period in CPS.
apSigRealmStatsTotalSessionsOutbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.11	Total number of outbound sessions.
apSigRealmStatsTotalSessionsNotAdmittedOutbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.12	Total number of outbound sessions rejected due to insufficient bandwidth.
apSigRealmStatsPeriodHighOutbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.13	Highest number of concurrent outbound sessions during the period.

SNMP GET Query Name	Object Identifier Name: Number	Description
apSigRealmStatsAverageRateOutbound	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.14	Average rate of outbound sessions during the period in CPS.
apSigRealmStatsMaxBurstRate	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.15	Maximum burst rate of traffic measured during the period (combined inbound and outbound).
apSigRealmStatsPeriodSeizures	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.16	Total number of seizures during the period.
apSigRealmStatsPeriodAnswers	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.17	Total number of answered sessions during the period.
apSigRealmStatsPeriodASR	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.18	Answer-to-seizure ratio, expressed as a percentage. For example, a value of 90 represents 90% or .90.
apSigRealmStatsAverageLatency	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.19	Average observed one-way signaling latency during the period in milliseconds.
apSigRealmStatsMaxLatency	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.20	Maximum observed one-way signaling latency during the period in milliseconds.
apSigRealmStatsMinutesLeft	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.21	Number of montly-minutes left in the pool per calendar month for a given realm.
apSigRealmStatsMinutesReject	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.22	Peg counts of number of rejected calls due to monthly-minutes constraints exceeded.
apSigRealmStatsShortSessions	apSigRealmStatsEntry: 1.3.6.1.4.1.9148.3.2.1.2.4.1.23	Lifetime number of sessions whose duration was less than the configured short session duration.

Acme Packet System Management MIB (ap-smgmt.mib)

The following VARBINDs are used in Transcoding related traps. They may not be polled and retrieved using an SNMP GET.

- Object Identifier Name: apSysMgmtMIBObjects (1.3.6.1.4.1.9148.3.2.1)
- Object Identifier Name: apSysMgmtGeneralObjects (1.3.6.1.4.1.9148.3.2.1.1)

SNMP Object Name	Object Identifier Name: Number	Description
apSysXCodeCapacity	apSysMgmtGeneralObjects 1.3.6.1.4.1.9148.3.2.1.1.34	Percentage of transcoding utilization.
apSysXCodeAMRCapacity	apSysMgmtGeneralObjects 1.3.6.1.4.1.9148.3.2.1.1.35	Percentage of licensed AMR transcoding sessions.
apSysXCodeAMRWBCapacity	apSysMgmtGeneralObjects 1.3.6.1.4.1.9148.3.2.1.1.36	Percentage of licensed AMR-WB transcoding sessions.

SNMP Object Name	Object Identifier Name: Number	Description
apSysXCodeEVRCCapacity	apSysMgmtGeneralObjects 1.3.6.1.4.1.9148.3.2.1.1.39	Percentage of licensedEVRC transcoding sessions.
apSysXCodeEVRCBCapacity	apSysMgmtGeneralObjects 1.3.6.1.4.1.9148.3.2.1.1.39	Percentage of licensedEVRCB transcoding sessions.

Acme Packet 6300 NIU Hotswap Guidelines

TheOracle Communications Session Border Controller 6300 provides a 3-slot chassis. Each chassis is monitored by a dedicated Hot-Swap sensor that maintains state information for the resident Network Interface Units.

The Oracle Communications Session Border Controller 6300 is the first multi-slot SBC that runs C series software. A hotswap sensor, located on each of the 6300's three slots, provides the ability to track the state of individual network interface units (NIUs) in the 6300 chassis.

There are 8 possible associated states as shown below.

```
PRESENT/NOT PRESENT       Checks the physical presence of a NIU inserted in a
6300 slot
LATCH OPEN/LATCH CLOSED   Checks the physical latch in the back of a NIU
POWER GOOD/ POWER BAD     Checks the power conditions of an NIU
READY/NOT READY           Checks all of the three previous state
```

A fully functional NIU is always in the following state: present, latch closed, power good, and ready.

The status of each NIU can be check by issuing the command **show platform hotswap** from the ACLI.

Network Interface Unit Removal/Replacement -- Standalone Node

A user or field engineer can remove a Network Interface Unit (NIU) from the 6300 chassis. When an NIU is removed, all services associated with the NIU (for example, media transmission, signaling, transcoding, and so on) will be interrupted and the system will be out-of-service (OOS).

Use the following procedure to remove an NIU.

1. Working from the back of the chassis, carefully open the NIU's latch. The hotswap LED placed directly above the NIU will transition from green (present/latch close/ready) to blinking blue. After 10 seconds it will stop blinking and remain blue (present/latch open/not ready).
2. You can now safely remove your NIU.

Removing the NIU generates a critical alarm notifying the user that the NIU has been removed. Additionally, the Vacuum Flourescent Display (VFD) displays a hardware alarm and starts blinking.From the ACLI, issue a "reboot force" command.

To see the alarm use: "display-alarms" from the ACLI.

To replace an NIU, insert an NIU of a different or the same type as the one used before.

The alarm manager clears the critical alarm, and issues new minor alarm to alert the user that an NIU was inserted after the boot sequence. At this point the system is still OOS.

After rebooting the system, service will be restored, and the new NIU will be functional

NIU Removal/Replacement -- High Availability Deployment

You cannot remove an NIU from the chassis of a 6300 SBC functioning in the active mode. If an NIU is accidentally removed, (1) all services will be interrupted; (2) a critical alarm will be triggered to notify the user that the NIU has been removed; (3) the system will relinquish its active state; and (4) proceed to reboot itself.

Under this scenario, service will not be interrupted because the standby node will assume the active role

In a 6300 functioning in standby mode, the user or field engineer can remove an NIU from the chassis. All services associated with the NIU (for example, media transmission, signaling, transcoding, and so on) will be interrupted and the standby node will be OOS. Removal of an NIU from the standby, however, will not affect the active node which remains in service.

Use the following procedure to remove an NIU from the standby node.

1. Working from the back of the chassis, carefully open the NIU's latch. The hotswap LED placed directly above the NIU will transition from green (present/latch close/ready) to blinking blue. After 10 seconds it will stop blinking and remain blue (present/latch open/not ready)
2. You can now safely remove the NIU. To replace an NIU, insert an NIU of the same type as the one used before.

Removing the NIU generates a critical alarm notifying the user that the NIU has been removed. Additionally, the VFD displays a hardware alarm and starts blinking

To see the alarm use: "display-alarms" from the ACLI.

The alarm manager clears the critical alarm, and issues a new minor alarm to alert the user that an NIU was inserted after the boot sequence. At this point the system is still OOS.

From the ACLI, issue a "reboot force" command.

After rebooting the standby, it will revert to standby mode.

Minimum TCM3 Versions on the Acme Packet 3950/4900

The SBC uses the TCM3 module on the AP3950 and AP4900 platforms. The TCM3 vendor periodically releases these modules with upgrades, including new memory. TCM3 operation on the SBC is, therefore, dependent on SBC software version. You must ensure compatibility between TCM3 and SBC software. Oracle provides you with multiple means of verifying software and TCM3 compatibility, including ACLI command output, SNMP traps, alarms and log messages.

Older SBC software versions does not operate properly with new TCM3 memory, but does allow the TCM3 cards to boot. Furthermore, system behavior when you use older software with this new memory is unpredictable. New SBC software provides backward compatibility, supporting both the most recent as well as the older TCM3 memory operating on the same SBC.

See the *Transcoding* section of your version's *Release Notes* to see the specific TCM3 cards supported by your software.

If you have booted the SBC to a version that does not support the correct TCM3 memory, the system persists with informing you of this incompatibility using several mechanisms including printing an incompatibility message to the console and SSH sessions every time you press the Enter key.

```
TCM 5 (Options 2) is not supported by Transcoding Vocallo 05.04.03-B571-PR
(HDT=01.07.01-B1898) (aid:xxxx, tid: tttt)
```

Additional means of verifying TCM3 support are presented below.

Applicable show Commands

The SBC provides two show commands, including **show xcode dbginfo** and **show xcode xlist**, that provide information from which you can determine TCM3 status.

The output of the **show xcode dbginfo** appears as follows.

```
Oracle# show xcode dbginfo
07:43:21
TRANSCODING SUPPORT:
Octasic DSP firmware version : Vocallo 05.04.04-B591-PR (HDT=01.07.01-B1898)
Startup Time        : 2023-02-02 20:34:48.114
Last Clear Time     : 2023-02-02 20:34:48.114
Last Read Time      : 2023-02-08 07:43:08.174
Current Time        : 2023-02-08 07:43:21.559
Up Time             : 5 Days, 11 Hours 8 Minutes 57 Seconds
                                       -- Life Time -- -- Recent --
PktApiStats:
   OpenConnectionCnt                = 12
   OpenSessionCnt                   = 12
   TotalPktSentCnt                  = 144                72
   TotalPktRecvCnt                  = 32943              88
  …
```

> **Note:**
>
> The Hardware Debugging Toolset (HDT) and Vocallo version are system software components.

The output of the **show xcode xlist** can provide insight into TCM3 module status. The state row displays operational TCM3 modules as ‘Active’ and presents the string ‘Boot Failure’, which can be a result of TCM and SBC software incompatibility.

```
Oracle# show xcode xlist
09:39:17
TCM DSPs #Sess Load State
=== ==== ===== ==== =========
00     1     0   0%   1 Active
01     1     0   0%   1 Active
02     -     -   -    1 Boot Failure
03     1     0   0%   1 Active
04     1     0   0%   1 Active
05     -     -   -    1 Boot Failure
```

```
06      -      -    -
07      -      -    -
```

Applicable SNMP Trap

The SBC generates an SNMP trap when the TCM3 modules are not supported. The system uses the generic HARDWARE_ERROR_TRAP trap to notify you of this failure. The trap string appears as follows.

```
DSP Boot Failure.
```

Applicable Alarms

The SBC generates an alarm if the BOOT FAILURE has occurred due to unsupported TCM3 modules. The system uses the generic BOOT FAILURE trap to notify you of this failure. The trap string appears as follows.

```
DSP#2 Boot Failure!
```

The system only generates this alarm the first time it detects an unsupported module regardless of how many unsupported TCM3 modules are present. The system also presents this alarm when you run the **display-alarms** command.

Applicable Logging

The SBC writes log messages to the log.xserv file when it detects both supported and unsupported TCM3 modules. The system generates either a supported or unsupported log messages for each installed TCM3.

There are two versions of this log message. A successful log message appear as follows.

```
Apr 25 14:55:43.328 [XSERV] (0) [NOTICE]
Vocallo 05.04.04-B591-PR (HDT=01.07.01-B1898) supports Options 0 (TCM-4)
```

An unsuccessful log message appear as follows.

```
Apr 25 14:55:43.328 [XSERV] (0) [NOTICE]
Vocallo 05.04.04-B591-PR (HDT=01.07.01-B1898) does not support Options 2
(TCM-5)
```

21

DTMF Interworking

Multimedia devices and applications can exchange user-input DTMF information end-to-end over IP networks. The Oracle Communications Session Border Controller provides the capabilities required to interconnect networks and devices that use different DTMF indication signaling protocols.

DTMF Indication

There are three ways to convey DTMF information for packet-based communications:

- DTMF audio tones: DTMF digit waveforms are encoded inline with voice packets. This method only works with uncompressed audio codecs like G.711. Compressed audio codecs like G.729 and G.723 are incompatible with DTMF audio. DTMF audio is also referred to as in-band tones.
- Out-of-band signaling events:
  SIP INFO messages with Content-Type: application/dtmf-relay define out-of-band signaling events for transmitting DTMF information. SIP INFO messages separate DTMF digits from the voice stream and send them in their own signaling message.
- RTP named telephony events (NTE): RFC 2833 telephone-events are a standard that describes how to transport DTMF tones in RTP packets according to section 3 of RFC 2833. Of the three RTP payload formats available, the Oracle Communications Session Border Controller supports RTP NTE.

RFC 2833 telephone-event

RFC 2833 specifies a way of encoding DTMF-indications in RTP media streams. It does not encode the audio of the tone itself, instead data represents the sent tone. RFC 2833 can be used with SIP.

RFC 2833 defines the format of NTE RTP packets used to transport DTMF digits and other telephony events between two peer endpoints. With the NTE method, the endpoints perform per-call negotiation of the DTMF transfer method. They also negotiate to determine the payload type value for the NTE RTP packets.

The NTE payload takes the place of codec data in a standard RTP packet. The payload type number field of the RTP packet header identifies the contents as 2833 NTE. The payload type number is negotiated per call. The local device sends the payload type number to use for RFC 2833 packets using SDP, which tells the other side what payload type number to use when sending the named event packets to the local device. Most devices use payload type number 101 for RFC 2833 packets, although no default is specified in the standard.

The RFC 2833 packet's RTP header also makes use of the timestamp field. Because events often last longer than the RFC 2833 packets sending interval, the timestamp of the first 2833 packet for an event represents the beginning reference time for subsequent RFC 2833 packets for that same event. For events that span multiple RTP packets, the RTP timestamp identifies the beginning of the event. As a result, several RTP packets might carry the same timestamp.

SIP INFO Messages

SIP INFO messages can send indications of DTMF audio tones between peers as part of the signaling path of the call. Upon receipt of a SIP INFO message with DTMF content, the gateway generates the specified DTMF tone on the receiving end of the call.

DTMF Transfer Processing Overview

Enabling 2 UAs to communicate different DTMF indications with each other is facilitated in two steps. First, the Oracle Communications Session Border Controller takes an active role in the SDP negotiation between two UAs, this is the capability negotiation step. You can configure your system to suggest the DTMF indication methods that each endpoint can and can not use.

The second step is translation evaluation. After the SDP negotiation is complete, based on system configuration and the media support on each call leg, the Oracle Communications Session Border Controller performs DTMF indication conversion or passive forwarding of DTMF indication messages between UAs.

Capability Negotiation

SDP capability negotiation is the first phase of enabling DTMF transfer. The completion of the SDP offer/answer exchange yields a set of supported codecs between each UA and the Oracle Communications Session Border Controller .

For DTMF transfer consideration, SDP manipulation is directed by parameters set in one of three configuration elements:

- codec policy
- signaling interface's RFC 2833 mode
- session agent's RFC 2833 mode

The Oracle Communications Session Border Controller performs SDP manipulation (addition, removal, or modification of supported codecs) toward the called party first by any applicable codec policies. If one or more of the actions which define telephone-event Modification by Codec Policy occurs, then SDP manipulation is only performed by the codec policy configuration, not by RFC 2833 mode parameters in the signaling interface or session agent.

If a codec policy attached to either the ingress or egress realm triggers SDP manipulation, then the other realm uses codec policy for any telephone-event SDP manipulation; none of the RFC 2833 Mode configurations are used for SDP manipulation.

> **Note:**
>
> If the call does not trigger the evaluation of any codec policies, all DTMF transfer processing is only subject to RFC 2833 Mode rules.

If none of the telephone-event Modification by Codec occur, then the Oracle Communications Session Border Controller performs SDP manipulation according to the RFC 2833 Mode parameters in the signaling interface or session agent.

SDP Manipulated by Codec Policy

When a call is received, any applicable codec policies are applied and evaluated. If telephone-event SDP is modified by codec policies, then SDP manipulation by RFC 2833 Mode is not performed for either side of the call.

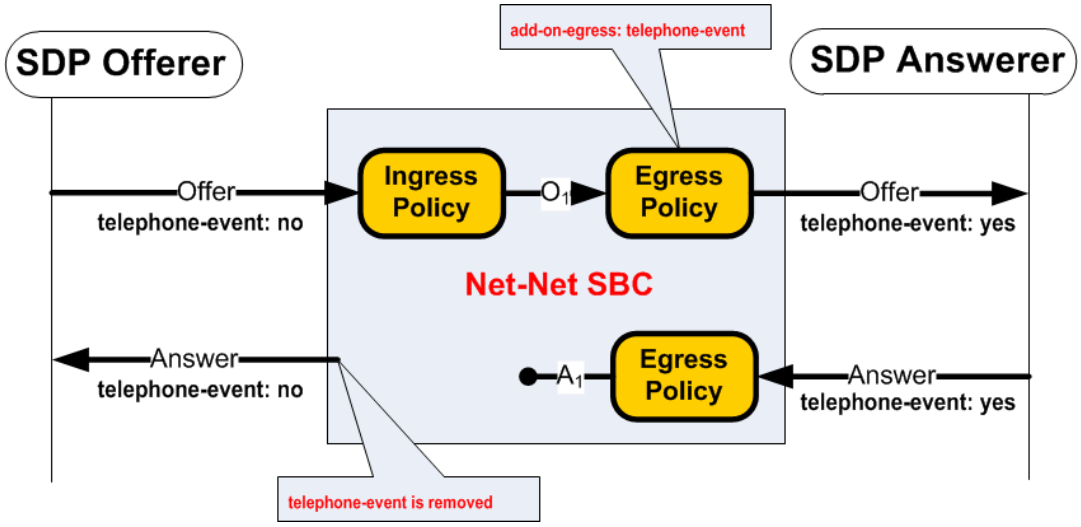
telephone-event Modification by Codec Policy

For qualifying if telephone-event modification in SDP was performed by codec policy, one of the following events had to have happened:

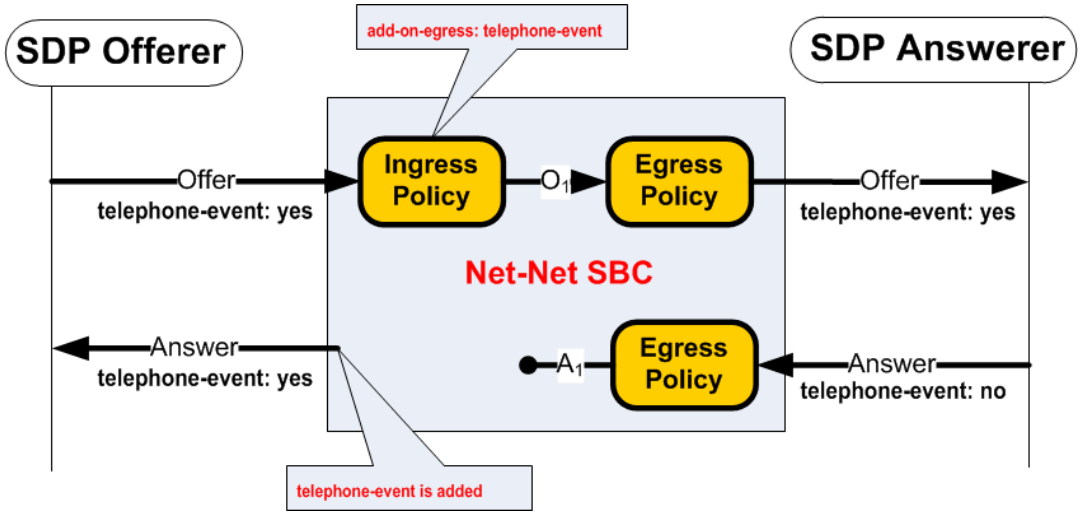
- Codec policy explicitly deleted telephone-event by configuring **allow telephone-event:no**
- Codec policy explicitly added telephone-event by configuring **add-on-egress telephone-event**
- Codec policy implicitly denied telephone-event by allowing one or more codecs but not adding telephone-event to the allow list
- Codec policy has **audio:no** configured in the allow list

The following three cases highlight how codec policies can manipulate telephone-event SDP. Once any of these cases occurs, SDP manipulation by RFC 2833 Mode parameter will not be performed.

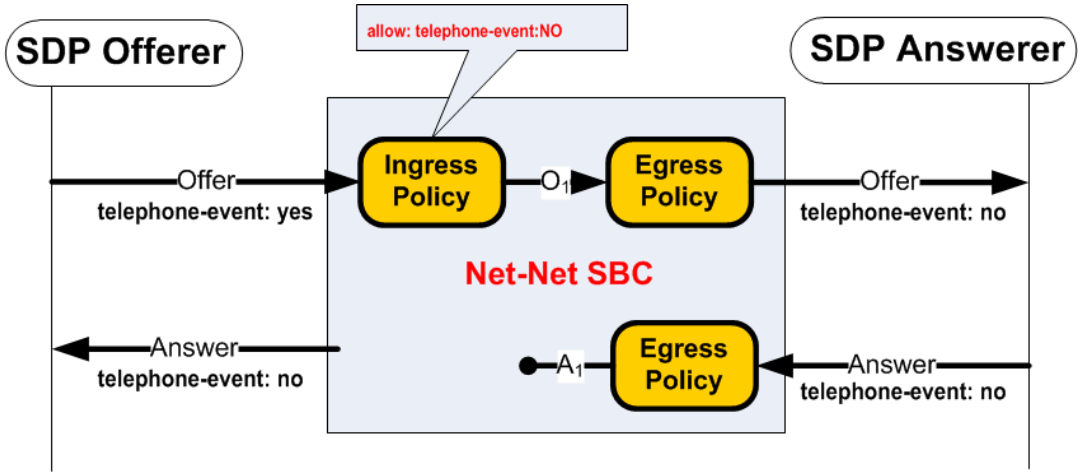
1. telephone-event added to SDP: The codec policy adds telephone-event to the SDP sent to the egress realm. The UA supports telephone-event.



2. telephone-event maintained in SDP: The codec policy maintains the offered telephone-event in the SDP sent into the egress realm. Although telephone-event was not answered in the egress realm, telephone-event is added back to the offerer-side SDP because of the add-on-egress setting in the ingress realm.



3. telephone-event removed from SDP: codec policy removes telephone-event from the initial SDP offer. telephone-event is not forwarded to the Answerer, and subsequently not returned from the answerer, or forwarded again to the offerer.



SDP Manipulated by RFC 2833 Mode

If none of the telephone-event Modification by Codec Policy events occur, as previously explained, SDP may still be modified by RFC 2833 Mode parameter if preferred or dual mode is configured.

The RFC 2833 mode parameter functions similarly to the add-on-egress parameter; it suggests telephone-event support, by adding it in SDP if not already there. This parameter must be set to preferred or dual to add telephone-event to SDP.

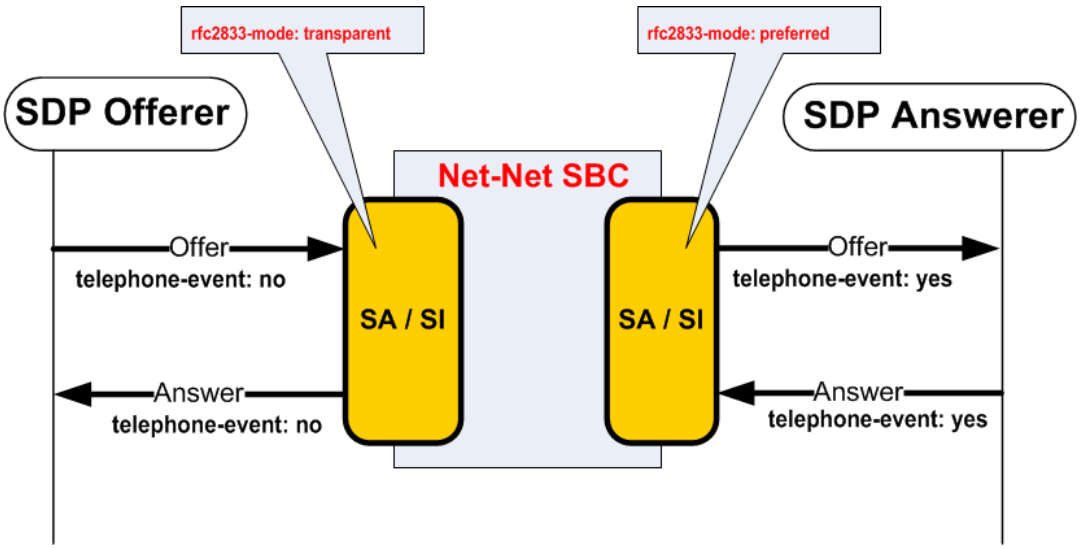
Transparent RFC 2833 Support

Setting a signaling interface or session agent's RFC 2833 mode to transparent disables the addition of RFC 2833 telephone-event to SDP upon egress. The Oracle Communications Session Border Controller passes the offered SDP capabilities to the next-hop signaling element.

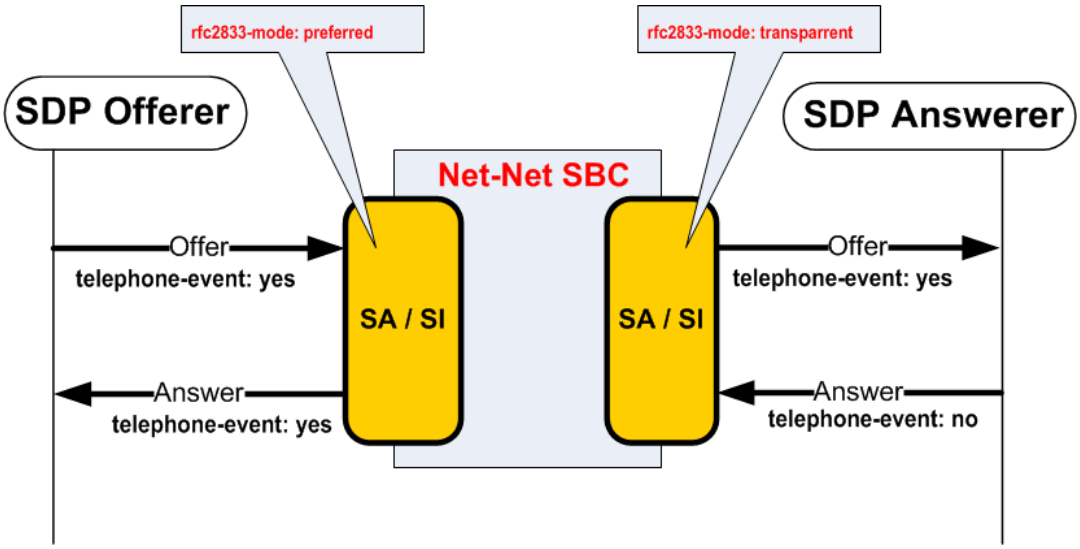
Preferred RFC 2883 Support

Setting a signaling interface or session agent's RFC 2833 mode to preferred indicates that the RFC 2833 telephone-event DTMF transfer method is the preferred method for sending a DTMF indication. In the capability negotiation phase a telephone-event media type will be inserted in the outgoing SDP offer, if it was not present in the original offer.

1. In the following example RFC 2833 mode is set to preferred on the egress side of the call. Because there is no telephone-event in the SDP, and RFC 2833 mode is set to preferred, the Oracle Communications Session Border Controller adds telephone-event to the SDP offer.



2. In the following example, RFC 2833 is set to preferred mode on the SDP offer side of the call. The Oracle Communications Session Border Controller maintains the telephone-event support even though telephone-event is not supported on the SDP answerer's side of the call.



RFC 2833 Payload Type Mapping

The Oracle Communications Session Border Controller does not require that call legs use the same media type for telephone-event. If each call leg uses a different media type value, the Oracle Communications Session Border Controller facilitates payload type mapping to ensure the telephone-event media stream be reliably transported across the call.

- On the SDP offer side, when the Oracle Communications Session Border Controller returns its SDP answer, it uses the same media type that the SDP offerer offered.
- The Oracle Communications Session Border Controller forwards the originally offered telephone-event media type to the SDP answerer. If telephone-event was added by RFC 2833 mode, the Oracle Communications Session Border Controller adds telephone-event with the media type value configured in the RFC 2833 payload parameter. If telephone-event was added by a codec policy, the Oracle Communications Session Border Controller adds telephone-event with the media type value configured in the media profile.
- If the SDP answerer returns a new value for telephone-event, the Oracle Communications Session Border Controller still supports RFC 2833 on that side of the call and uses the media type that the answerer sent.

Translation Evaluation

After SDP has been negotiated, the Oracle Communications Session Border Controller determines what types of DTMF translation takes place for the call. The Oracle Communications Session Border Controller sequentially evaluates the following rules for each call leg to determine what DTMF indication type it will forward to an endpoint.

1. RFC 2833—When the SDP offer/answer exchange resolves to both the Oracle Communications Session Border Controller and the endpoint supporting RFC 2833 on one side of the call, the Oracle Communications Session Border Controller will send DTMF indications in RFC 2833 format.
2. DTMF audio tones—Three conditions must be met for the Oracle Communications Session Border Controller to support DTMF audio tones, as transcoded from another DTMF indication form:
   - The applicable codec policy's dtmf in audio parameter is set to preferred
   - The endpoint and Oracle Communications Session Border Controller have negotiated to a DTMFable codec (G711)
   - Transcoding resources are available

   > **Note:**
   >
   > Because of rule number one, rule number two can not happen if RFC 2833 is supported in SDP—Only one media-based DTMF transfer method, RFC 2833 or DTMF audio tones may be used on a call leg.

3. If neither RFC 2833 nor DTMF Audio tones are supported on a call leg, as a result of SDP negotiation, then the Oracle Communications Session Border Controller forwards DTMF indication messages to that side in signaling message format (SIP INFO).

   In the following images that illustrate DTMF transfer scenarios, a gears icon appears when relevant. This icon indicates that the Oracle Communications Session Border Controller

performs DTMF indication processing, creating DTMF audio tones or RFC 2833 telephone-event messages from another form of DTMF indication.

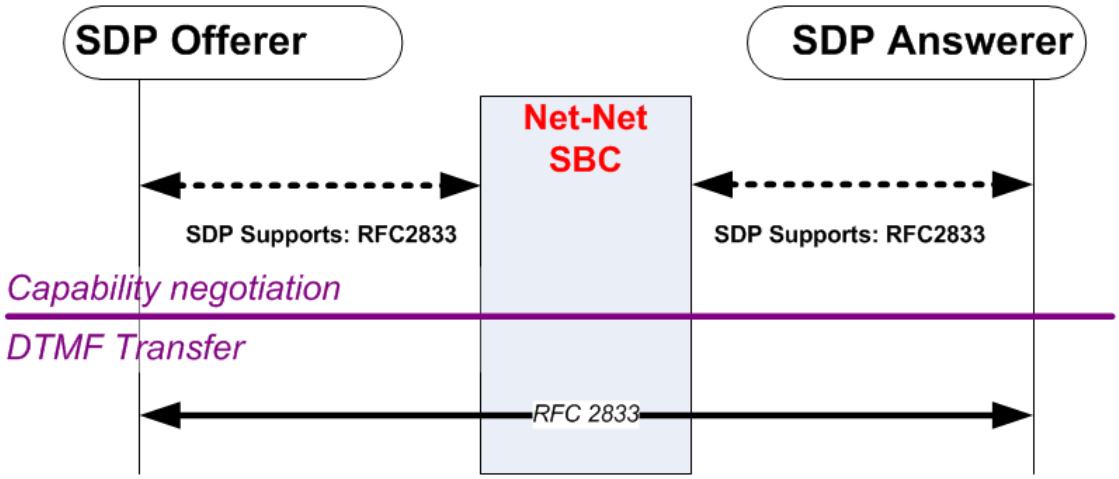
RFC 2833 Sent by Offerer

In the following three examples, the SDP offerer sends DTMF indication messages in RFC 2833 format. The SDP answerer can receive DTMF indications in the format identified in each example.

RFC 2833 to RFC 2833

When the SDP offer and answer sides of a call both support RFC 2833, the Oracle Communications Session Border Controller forwards RFC 2833 messages between both sides of the call. No processing is used to transform these DTMF-indication massages to another format.

> **Note:**
>
> When the audio stream is transcoded, DTMF audio is completely removed from the audio stream.



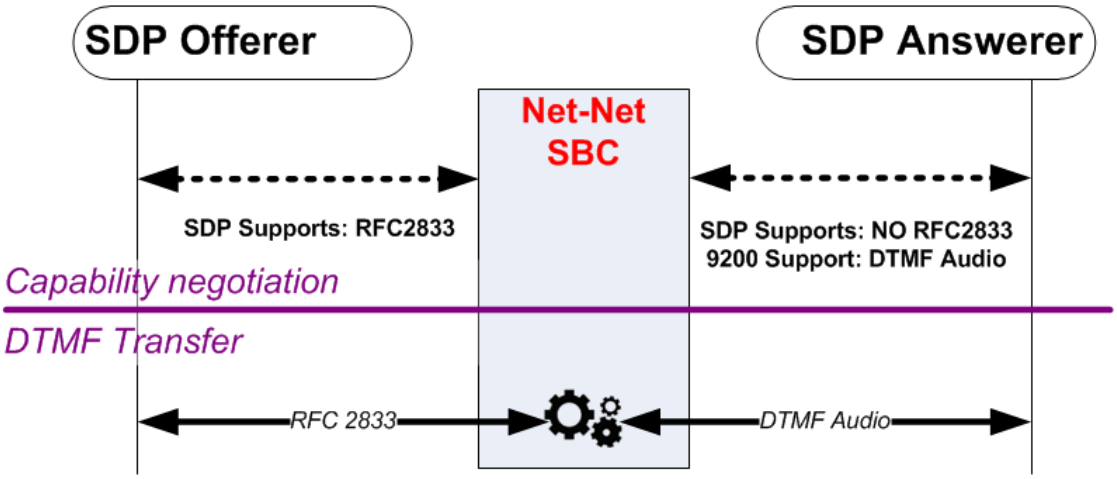
A SIP INFO message received from either the offerer or answerer is forwarded unconverted to the other side of the call.

If there is no audio transcoding enabled for this call, and the egress side is set to dual, a received SIP INFO message will not be converted to both RFC 2833 and SIP INFO messaged for sending to the other side of the call.

If DTMF audio tones are received from either the offerer or answer, they are forwarded unconverted to the other side (when the audio portion of the call is not transcoded).

RFC 2833 to DTMF Audio Tones

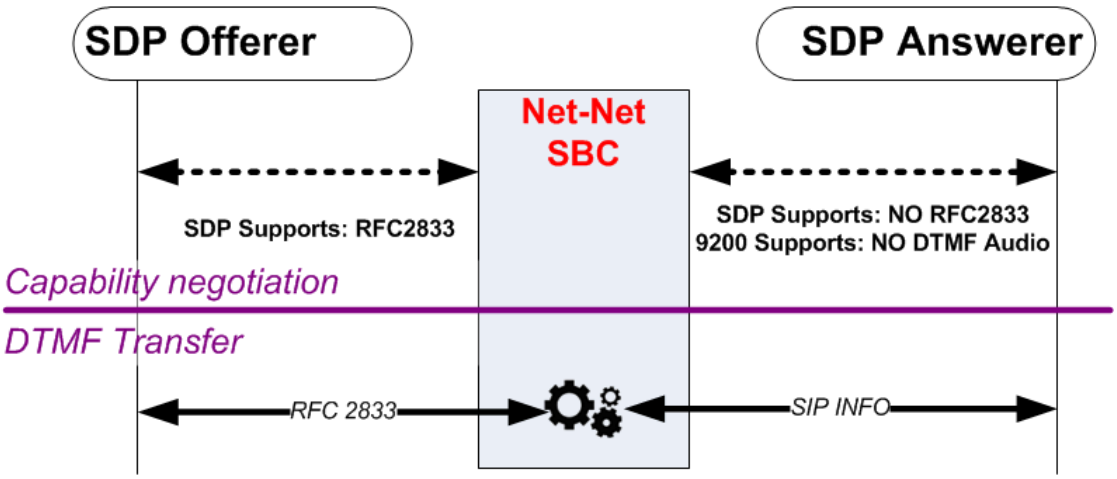
When the SDP offer side supports RFC 2833, and the SDP answer side supports the three DTMF Audio Tone conditions and does not support RFC 2833, the Oracle Communications Session Border Controller converts from RFC 2833 to DTMF audio tones for the call.



A SIP INFO message received by the Oracle Communications Session Border Controller from either the offerer or answerer is converted into the DTMF transfer method that the previous diagram shows for the egress side of the message. In this case, transcoding resources are used.

RFC 2833 to SIP INFO

When the SDP offer side supports RFC 2833 and the SDP answer side does not support the DTMF conditions nor RFC 2833, the Oracle Communications Session Border Controller converts from RFC 2833 to SIP INFO.

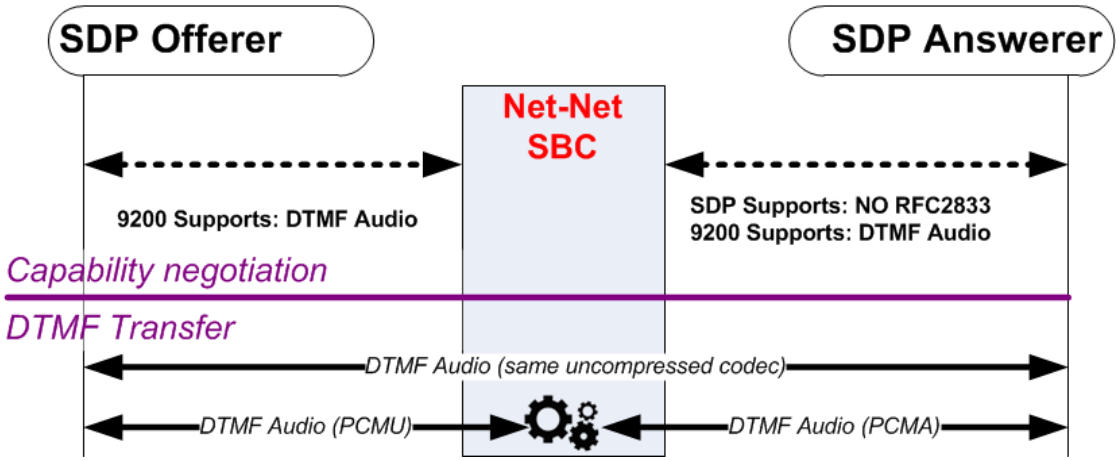


DTMF Audio Tones Sent by Offerer

In the following three examples, the SDP offerer sends DTMF indication messages in DTMF audio tones format. The SDP answerer can receive DTMF indications in the format identified in each example.

DTMF Audio to DTMF Audio

If the SDP offer and answer sides both support the same type of G711 codec, the audio stream is forwarded between the two sides without processing.

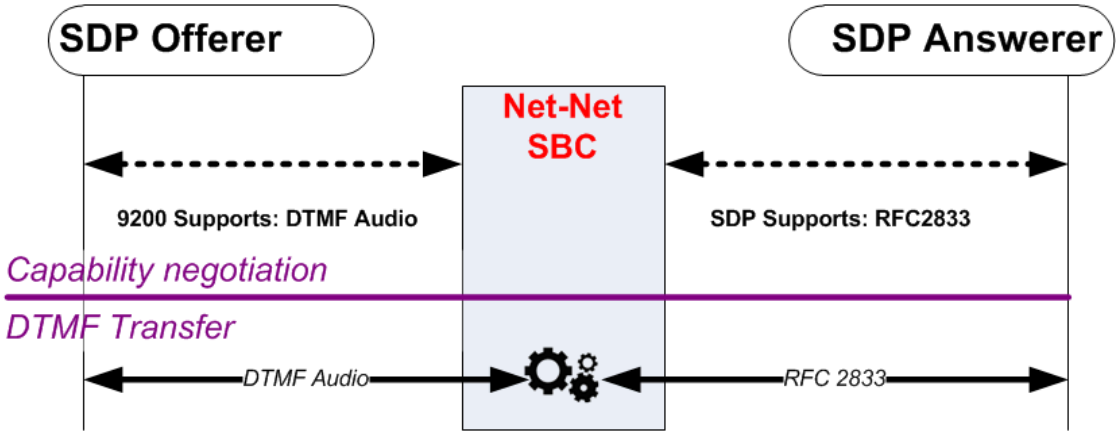


If the two sides of the call support DTMF audio tones, but use different audio codecs, and the SDP answer side supports the three DTMF Audio Tone conditions and does not support RFC 2833 then the Oracle Communications Session Border Controller will preserve DTMF audio tone indication across the call.

Transcoding resources are used only if different audio codecs are used or the Override Preferred DTMF Audio feature is enabled.

DTMF Audio to RFC 2833

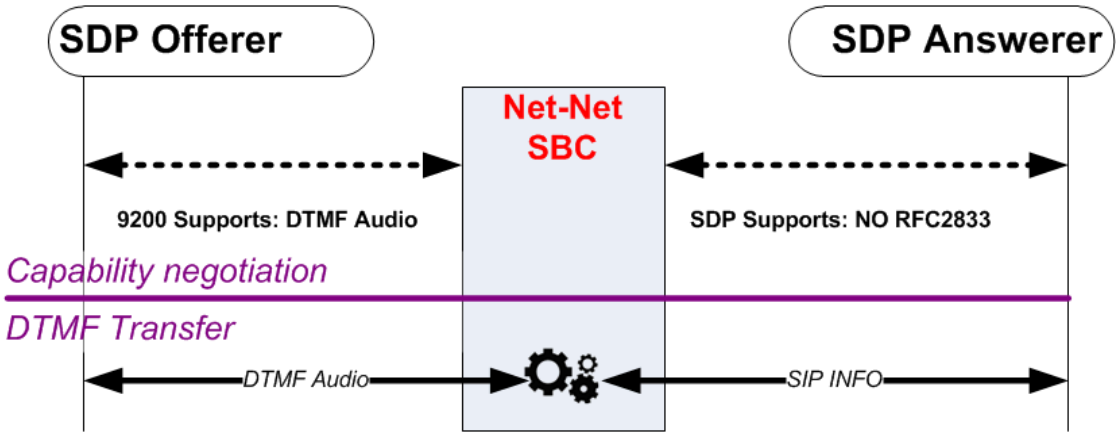
When the SDP offer side supports DTMF audio tones, and the SDP answer side supports RFC 2833, and transcoding resources are available, and does NOT support either or both of the first two DTMF Audio tone conditions, then the Oracle Communications Session Border Controller will convert incoming DTMF audio tones to outgoing RFC 2833 packets.



Transcoding resources are always required in this scenario.

DTMF Audio to SIP

When the SDP offer side supports DTMF audio tones, and the SDP answer side does not support RFC 2833, and does not support the three DTMF Audio Tone conditions, then the Oracle Communications Session Border Controller converts incoming DTMF audio tones to SIP INFOmessages.



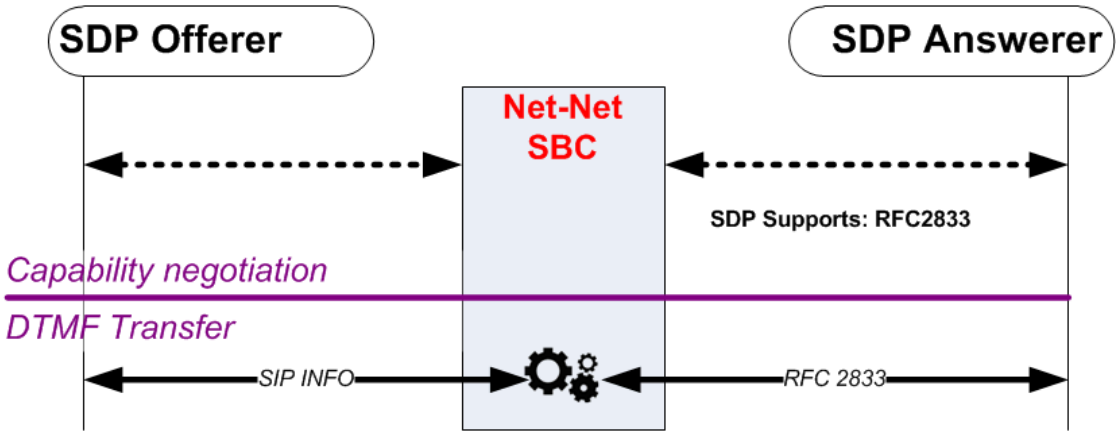
Transcoding resources are always required in this scenario.

SIP INFO Sent By Offerer

In the following three examples, the SDP offerer sends DTMF indication messages in SIP INFO message format. The SDP answerer can receive DTMF indications in the format identified in each example.

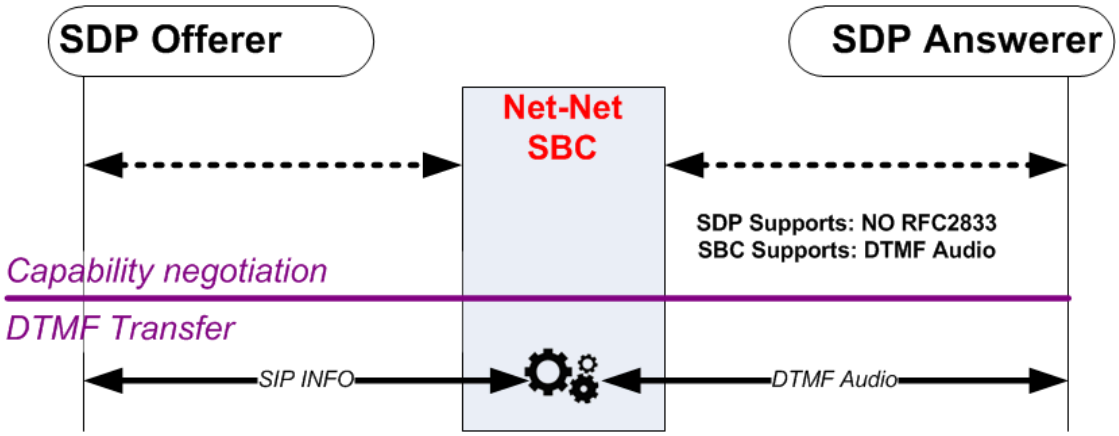
SIP INFO to RFC 2833

When the SDP offer side sends a SIP INFO message, and the SDP answer side supports RFC 2833, then the Oracle Communications Session Border Controller will convert incoming SIP INFO messages to outgoing RFC 2833 packets.



SIP INFO to DTMF Audio

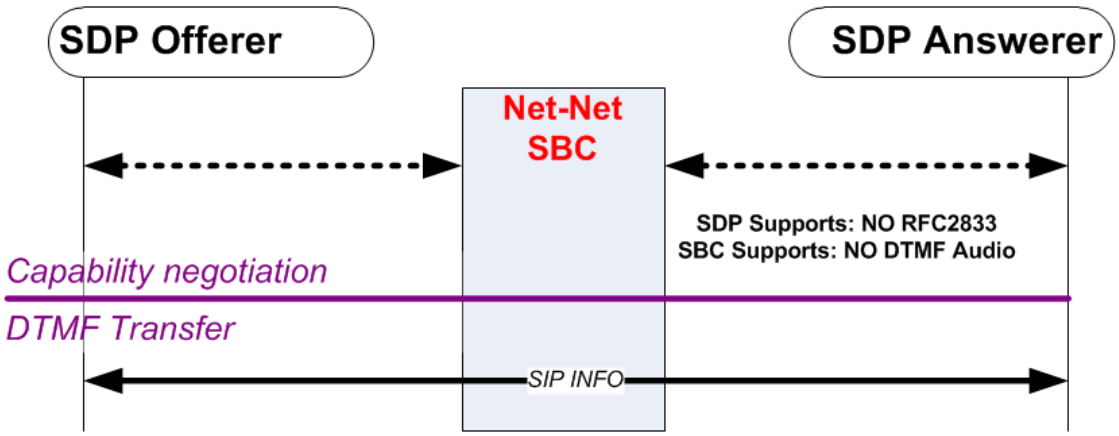
SIP INFO will only be converted to DTMF audio tones only if RFC 2833 is not supported, dtmf-in-audio is enabled, the answer side supports a G711 codec, and transcoding resources are available.



Transcoding resources are always required in this scenario.

SIP INFO to SIP INFO

When the SDP offer side sends a SIP INFO message and the SDP answer side does not support RFC 2833 and does not support the three DTMF audio tone conditions, the SIP INFO message will always be forwarded as the like SIP INFO message.



Dual Mode

Dual mode is used to send both RFC 2833 and the protocol-specific DTMF indication to a UA when possible: SIP INFO from a SIP interface.

To consider dual mode scenarios, the Oracle Communications Session Border Controller sets up the call SDP. At the conclusion of the capability negotiation, the Oracle Communications Session Border Controller is configured to support DTMF Audio tones or RFC 2833 independently for each side of the call.

When the call leg supports RFC 2833 as the means of DTMF transfer, the Oracle Communications Session Border Controller checks if the SIP interface's (or session agent's) RFC 2833-mode parameter is configured to dual. If it is, the Oracle Communications Session Border Controller sends both RFC 2833 and SIP INFO messages to the UA on this side of the call.

> **Note:**
>
> Whether RFC 2833 support was initiated between the Oracle Communications Session Border Controller and the UA by a codec policy or by the rfc2833-mode parameter, the Oracle Communications Session Border Controller looks to the rfc-2833 parameter to consider if dual mode is supported.

When the call leg supports DTMF audio tones as the means of DTMF transfer, the Oracle Communications Session Border Controller checks if the codec policy's dtmf-in-audio parameter is configured to dual. If it is, the Oracle Communications Session Border Controller sends both DTMF audio tones and SIP INFOmessages to the UA on this side of the call.

P-Dual-Info Header

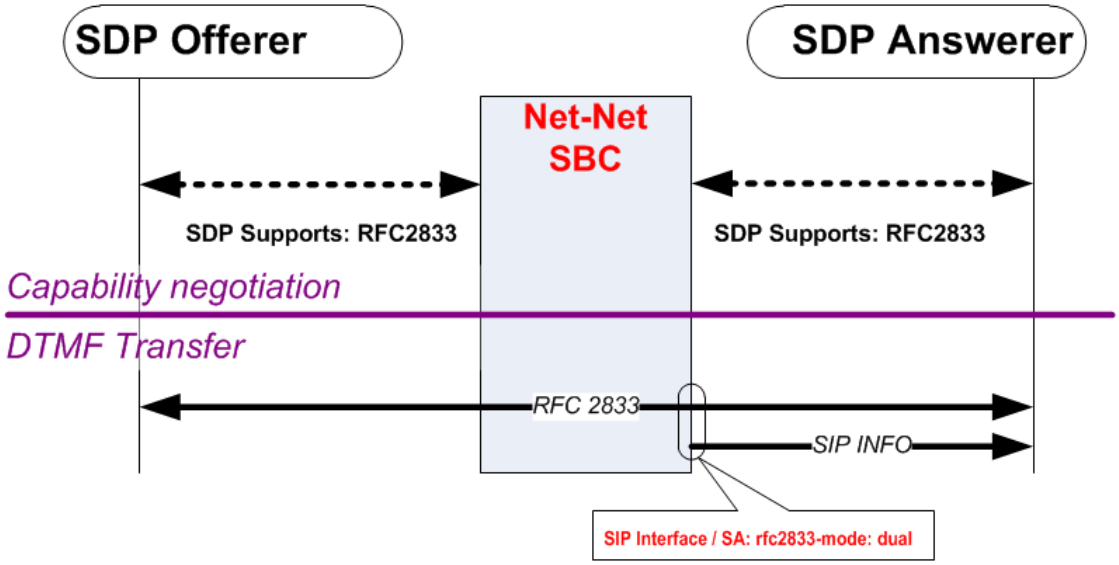
When the Oracle Communications Session Border Controller forward both media DTMF indication and signaling based DTMF indication for the same received DTMF indication, a P-Dual-Info header is added to the forwarded signaling message. You can configure the appearance of the header with the **dual-info** option. The default header appearance is:

```
P-dual-info: true
```

P-Dual-Info headers are only inserted into SIP INFO messages.

Example 1

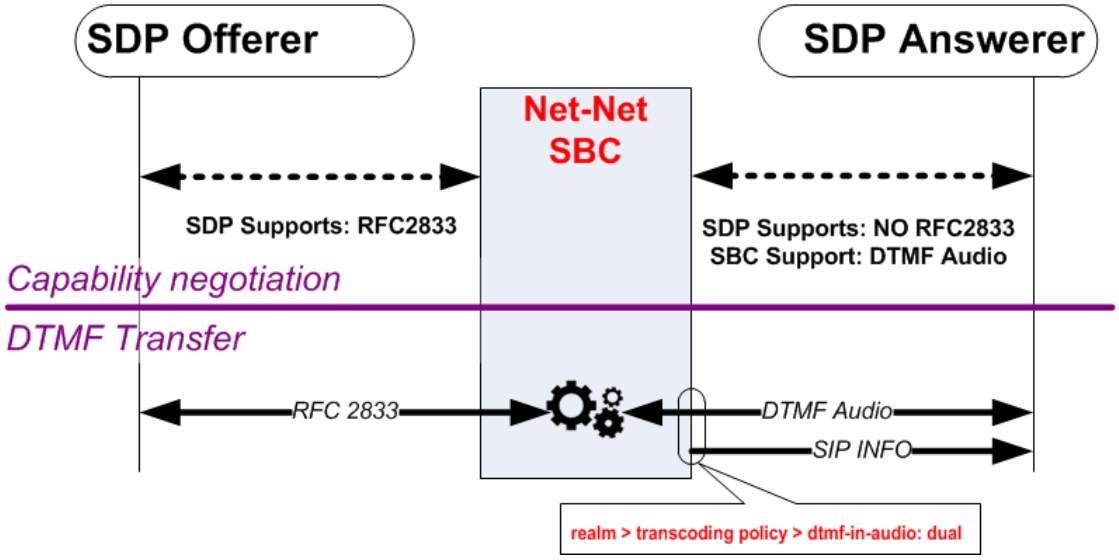
In this example, RFC 2833 is supported on the egress side of the call. The egress SIP interface or session agent's rfc2833-mode is set to dual mode. When the Oracle Communications Session Border Controller forwards RFC 2833 to the SDP answerer, it also creates and forwards a corresponding SIP INFO message toward the target.



Example 2

In this example RFC 2833 telephone-event is not supported on the egress side of the call, but DTMF audio tones are. If the Oracle Communications Session Border Controller receives an

RFC 2833 message, it is converted to DTMF audio tones. When the Oracle Communications Session Border Controller forwards DTMF audio tones to the SDP answerer, it also creates and forwards a corresponding SIP INFO message toward the target.



DTMF Transfer for Spiral Calls

A spiral call occurs when a call's signaling messages loop back through the Oracle Communications Session Border Controller. Most commonly the signaling path is from one UA, through the Oracle Communications Session Border Controller, to a call server, back through the Oracle Communications Session Border Controller, to another UA. The media path is from one UA, through the Oracle Communications Session Border Controller, to the other UA. For DTMF indication processing, only the call legs between the endpoints and Oracle Communications Session Border Controller are considered.

The Oracle Communications Session Border Controller evaluates that the signaling path to and from the call server terminates on the same IP address and port on the Oracle Communications Session Border Controller. This means that it's a spiral call. In addition, the media IP addresses and ports in the SDP indicate that the Oracle Communications Session Border Controller does not need to send the media into the call server's realm.

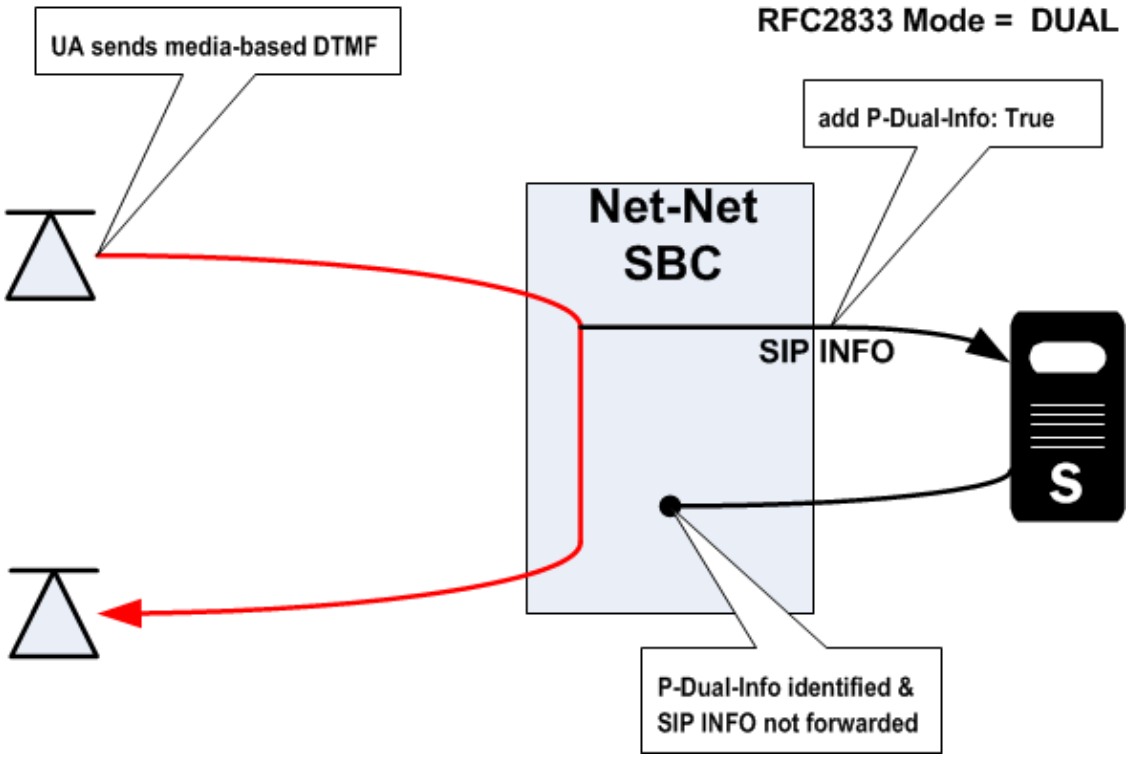
An issue occurs when DTMF indication is relayed either by RFC 2833 or DTMF audio tones for a spiral call. Since the DTMF indication is in the media path, the call server remains unaware of the signaling; no media-based DTMF indication digits will ever reach the call server. In order to include the call server in the DTMF-indication signaling, you should set the dtmf-in-audio and/or rfc-2833 mode used for the realm or signaling interface where the call server is to dual.

P-Dual-Info Header

The P-Dual-Info header is used in a spiral call scenario, when the call leg to (and from) the call server is set to dual. While the media portion of a spiral call goes from endpoint to endpoint through the Oracle Communications Session Border Controller, the signaling portion of the call loops through a call server.

The Oracle Communications Session Border Controller inserts a P-Dual-Info header into a SIP info message sent to the call server, which in turn forwards the SIP INFO message back to the Oracle Communications Session Border Controller. Seeing the P-Dual-Info header, the Oracle Communications Session Border Controller knows not to forward this SIP INFO message to

the target endpoint because it would be a duplication of DTMF indication already sent to the endpoint in media format.



DTMF Transfer Hardware Processing

DTMF transfer processing, the conversion between two DTMF transfer types, occurs in either the transcoding NIU's Digital Signal Processors (DSPs) or the Network Processors (NPs). Understanding where the processing takes place is important to determine which subsystem uses extra processing load per conversion.

There are a few rules you can use to determine which subsystem performs the DTMF transfer processing:

- If audio transcoding is enabled for the call, DTMF transfer processing occurs in the transcoding modules.
- If DTMF audio tones are generated from RFC 2833 or from signaling messages (SIP INFO), DTMF transfer processing occurs in the transcoding modules.
- If the global translate non inband event parameter is enabled, DTMF transfer processing occurs in the transcoding modules.
- If signaling to RFC 2833 processing occurs in either direction of the call, and the previous 3 conditions are not valid, DTMF transfer processing occurs in the NPs.

DTMF Transfer Configuration

RFC 2833 Session Agent Configuration

Session agents, used as a way to classify and act on a subset of a signaling interface's traffic, also have **rfc2833-mode** and **rfc2833-payload** parameters. The configurations of these

parameters overrides the configuration of the same-named parameters on the signaling interface where the session agent resides. You can set the **rfc2833-mode** parameter to **none** for a session agent to revert to the parent signaling interface's two RFC 2833 settings.

ACLI Configuration and Instructions

This section explains how to configure the RFC 2833 mode on a signaling interface and on a session agent configured for that signaling interface. The session agent's configuration takes precedence over the signaling interface, unless the session agent's rfc2833-mode is set to none. In that case, the signaling interface's configuration is used for applicable traffic.

SIP Interface

To configure the RFC 2833 mode on a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. If you are adding support for this feature to a pre-existing SIP interface, then you must select the specific configuration instance, using the ACLI **select** command.
5. **rfc2833-mode**—Set this parameter to either **transparent**, **preferred**, or **dual** based upon the behavior your want for this SIP interface.
   - transparent—does not add RFC 2833 telephone-event into SDP if not present, and does not prefer.
   - preferred—adds RFC 2833 telephone-event media type into SDP and prefers to use this method for DTMF indication.
   - dual—adds RFC 2833 telephone-event media type into SDP and sends both SDP and signaling-based DTMF indications if possible.
6. **rfc2833-payload**—Set this parameter to the media-type value you wish to use when inserting RFC 2833 telephone-events into an SDP offer. 101 is the generally accepted media type for RFC 2833 telephone-events.
7. Save and activate your configuration.

Session Agent

Session agent RFC 2833 mode configurations override those on the signaling interface where they exit. The **none** parameter is used to defer to the signaling interface.

To configure the RFC 2833 mode on a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# session-router
```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. Select the session agent where you want this feature.

```
ORACLE(session-agent)# select 1
```

5. **rfc2833-mode**—Set this parameter to either **none, transparent**, **preferred**, or **dual** based upon the behavior your want for this SIP interface.
   - none—defaults to the behavior of the SIP interface for traffic that matches this session agent.
   - transparent—does not add RFC 2833 telephone-event into SDP if not present, and does not prefer.
   - preferred—adds RFC 2833 telephone-event media type into SDP and prefers to use this method for DTMF indication.
   - dual—adds RFC 2833 telephone-event media type into SDP and sends both SDP and signaling-based DTMF indications if possible.
6. **rfc2833-payload**—Set this parameter to the media-type value you wish to use when inserting RFC 2833 telephone-events into an SDP offer. 101 is the generally accepted media type for RFC 2833 telephone-events.
7. Save and activate your configuration.

Codec Policy

To configure a codec policy to support DTMF audio tones, as transcoded:

1. In Superuser mode, type configure terminal and press Enter.

```
ORACLE# configure terminal
```

2. Type media-manager and press Enter.

```
configure# media-manager
```

3. Type **codec-policy** and press Enter.

```
ORACLE(media-manager)# codec-policy
ORACLE(codec-policy)#
```

4. If you are adding support for this feature to a pre-existing configuration, then you must select the specific configuration instance, using the ACLI **select** command.

```
ORACLE(codec-policy)# select
<name>:
1: private
```

```
2: public
selection:1
ORACLE(codec-policy)#
```

5. **dtmf-in-audio**—Set this parameter to **disabled**, **preferred**, or **dual** based upon how the Oracle Communications Session Border Controller should support the conversion of signaling messages or RFC 2833 to DTMF Audio tones in the realm where this codec policy is active.
   - disabled—does not support DTMF audio tones as transcoded in this realm.
   - preferred—supports DTMF audio tones as transcoded in this realm.
   - dual—supports both transcoded DTMF audio tones and signaling-based DTMF indications if possible.
6. Save and activate your configuration.

Translate Non2833 Event Behavior

To configure the exceptional behavior:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter to begin configuring this feature.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **translate-non-rfc2833-event**—Set this parameter to enabled to use non-default behavior described in Override Preferred RFC 2833.
5. Save and activate your configuration.

P-dual-info Header Appearance

Customizing the P-Dual-Info header is performed globally from the sip config.

To configure how the P-dual-info header appears:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. Type options followed by a Space.
5. After the Space, type the P-Dual-Info header information in the following format:

```
+dual-info="<header-name>"
```

   For example:

```
ORACLE(sip-config)# options dual-info=P-Dual-Info
```

6. Save your work using the ACLI **done** command.

RFC 2833 Customization

RTP Timestamp

As a media flow with injected RFC 2833 telephone-event packets exits the Oracle Communications Session Border Controller, the newly generated RTP packets are timestamped in one of two ways. If RFC 2833 generation is preformed in the NPs, the default method of creating the timestamp for a generated RFC 2833 packet is to use the previous RTP packet's timestamp and add 1.

Alternatively, the Oracle Communications Session Border Controllercan estimate the actual time that the injected RFC 2833 telephone-event packet will leave the system and use that. This method tags the injected DTMF indication packet more accurately than the than previous packet + 1 method. As an additional bonus, the packet's checksum is regenerated. This alternate timestamp creation behavior is configured by setting the **rfc2833-timestamp** parameter to enabled.

When RFC 2833 telephone-event generation is preformed by the transcoding modules, the packets' timestamps are the set to the time that the packets leave the Oracle Communications Session Border Controller,.

RFC 2833 telephone-event duration intervals

If an incoming SIP INFO message's DTMF indication duration is unspecified, the Oracle Communications Session Border Controller, uses a default 250 ms duration for the generated RFC 2833 telephone-event. Otherwise, the SIP INFO's specified event duration is used. RFC 2833 telephone-event packets are still generated at 50 ms intervals upon egress. At the conclusion of the DTMF indication, the three end-event packets are sent. The packet arrangement when the user presses the digit 5 for 160ms, with the default 50ms interval follows:

1. RFC2833 packet 1, digit: 5, duration 50 ms
2. RFC2833 packet 2, digit: 5, duration 100 ms
3. RFC2833 packet 3, digit: 5, duration 150 ms
4. RFC2833 packet 4 (end packet), digit: 5, duration 160 ms

5. RFC2833 packet 5 (end packet), digit: 5, duration 160 ms
6. RFC2833 packet 6 (end packet), digit: 5, duration 160 ms

When either no DTMF event duration is specified, or the event duration is less than the 50ms default minimum, you can set the default RFC 2833 telephone-event duration using **default-2833-duration** parameter in the media manager configuration. This is the value that the Oracle Communications Session Border Controller, uses for the duration of a telephone event when none is specified in the incoming message. The **default-2833-duration**'s valid range is 50-5000ms. The Oracle Communications Session Border Controller, also uses this configured value when it receives a SIP INFO message with a duration less than the minimum signal duration.

You can configure the minimum duration at which RFC 2833 telephone-events are generated by the Oracle Communications Session Border Controller, using the **min-signal-duration** option in the media manager configuration, thus changing the lower threshold of the **default-2833-duration** parameter from 50 ms to your own value. If the duration the Oracle Communications Session Border Controller, receives is less than the threshold, it uses the value configured in the **default-2833-duration** parameter.

> **Note:**
>
> Timestamp and duration changes will not take effect when the 2833 timestamp (rfc-2833-timestamp) and default-2833-duration parameter is altered in the media manager configuration during a SIP INFO to DTMF Interworking scenario.

RFC 2833 End Packets

When the Oracle Communications Session Border Controller, generates RFC 2833 telephone-event packets, they are forwarded from the egress interface every 50 ms by default. Each packet includes the digit and the running total of time the digit is held. Thus DTMF digits and events are sent incrementally to avoid having the receiver wait for the completion of the event.

At the conclusion of the signaled event, three end packets stating the total event time are sent. This redundancy compensates for RTP being an unreliable transport protocol.

You can configure your Oracle Communications Session Border Controller, to generate either the entire start-interim-end RFC 2833 packet sequence or only the last three end 2833 packets for non-signaled digit events using the **rfc2833-end-pkts-only-for-non-sig** parameter. If the parameter were enabled, the RFC 2833 telephone-event packets for the same event would appear as follows:

1. RFC2833 packet 1 (end packet), digit: 5, duration: 160 ms
2. RFC2833 packet 2 (end packet), digit: 5, duration: 160 ms
3. RFC2833 packet 3 (end packet), digit: 5, duration: 160 ms

ACLI Instructions and Examples

To configure RFC 2833 customization:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **media-manager** and press Enter to begin configuring this feature.

```
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

4. **rfc2833-timestamp**—Set this parameter to enabled to use the estimated real departure timestamp on an injected RFC 2833 telephone-event packet.
5. **default-2833-duration**—Set this parameter to the default value you wish to use when the Oracle Communications Session Border Controller, creates or generates DTMF-indication messages.
6. **rfc2833-end-pkts-only-for-non-sig**—Set this parameter to enabled for the Oracle Communications Session Border Controller, to only send the three RFC 2833 telephone-event end-packets to indicate a total event duration, rather than the running total from time=0.
7. **options**—Set the options parameter by typing **options**, a Space, the option name **min-signal-duration=x** (where **x** is the value in milliseconds you want to use for the threshold) with a plus sign in front of it. Then press Enter.
8. Save and activate your configuration.

RFC2833 and KPML Interworking

Keypad Markup Language (KPML) is used to indicate DTMF tones in SIP messaging. KPML is used by the Key Press Stimulus Package as a SIP Event Notification Package, transmitting DTMF tone indications via NOTIFY messages. You can configure the Oracle Communications Session Border Controller (SBC) to perform Event Notification with an endpoint on one call leg and perform digit encapsulation to RFC 2833 telephone-events on the other call leg.

KPML to RFC2833

KPML to RFC2833 interworking requires that the INVITE request or response's SDP does not contain telephone-event, and is received from

- A session agent with **kpml-interworking** set to enabled
- A session agent with **kpml-interworking** set to inherited from the SIP interface (with **kpml-interworking** set to enabled)
- A previous hop that is not a session agent, and the SIP interface the message received on has **kpml-interworking** set to enabled.

The egress side of the call must have **rfc2833-mode** set to **preferred** in either the SIP interface or session agent, so that the SBC inserts `telephone-event` in the SDP. Setting **rfc2833-mode** to dual is unsupported for KPML interworking. The answerer must respond to the invite, accepting the `telephone-event` media. The `Allow-Event: kpml` header is removed from the INVITE request or response when KPML-interworking is not set to enabled on the next hop.

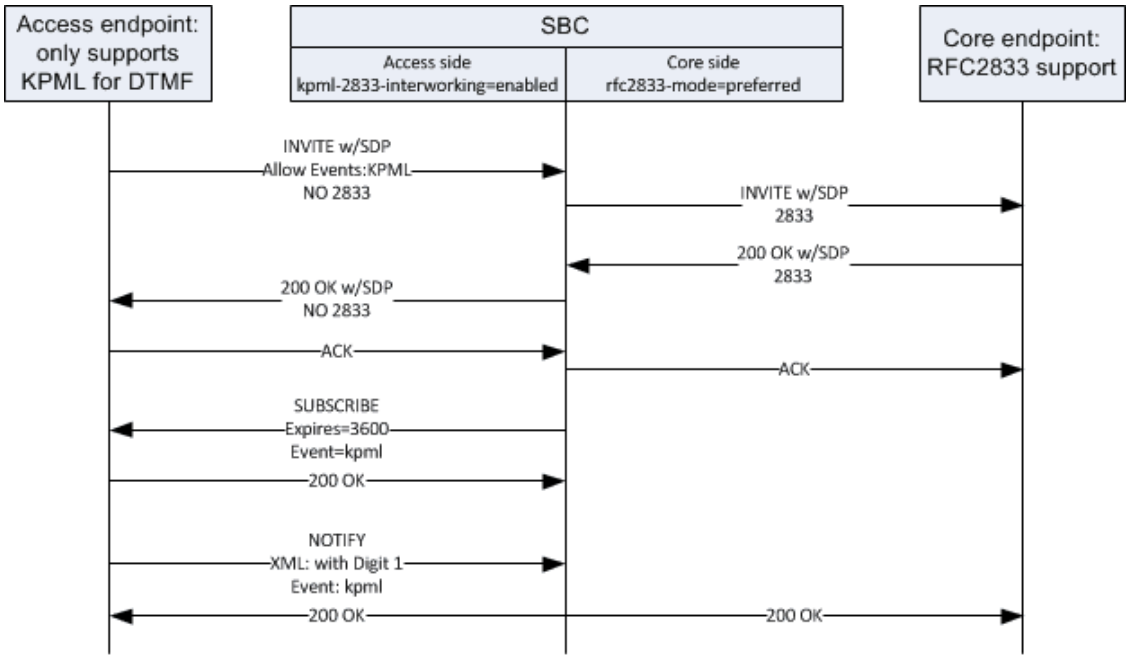
If the INVITE succeeds, the SBC replies with a SUBSCRIBE request for the KPML event to the caller.

If the caller replies to the SUBSCRIBE with a 2xx, all subsequent NOTIFY request received on the dialog are processed and replied to with a 200 OK. Each digit in the NOTIFY request will generate a `telephone-event` on the egress side of the call.

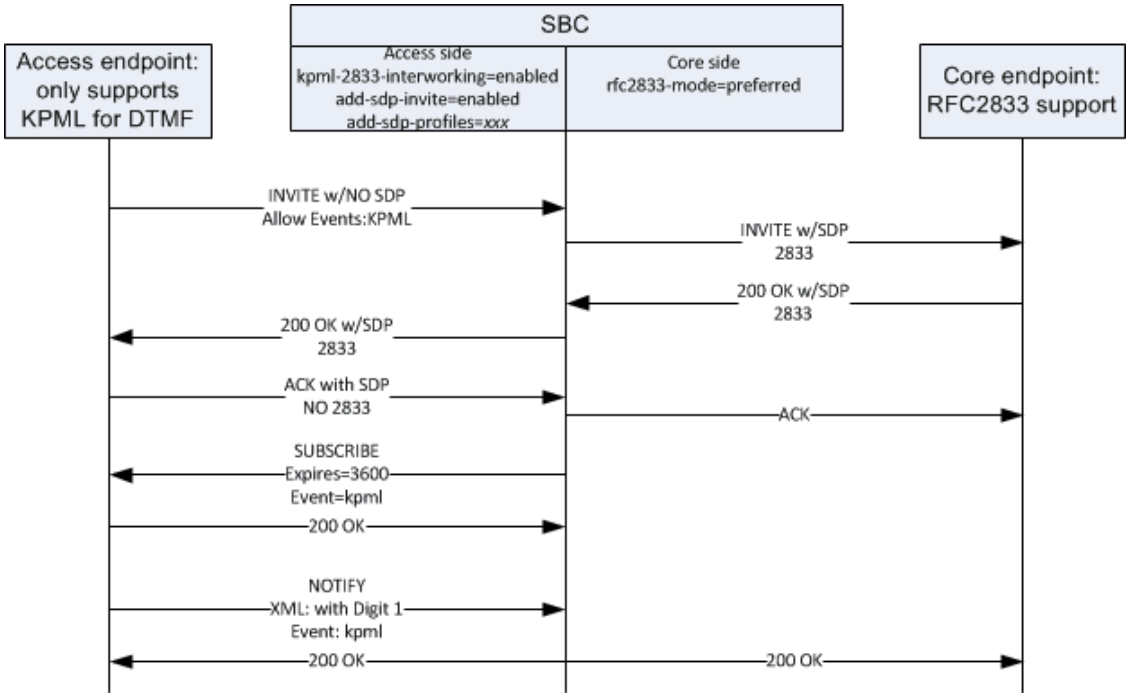
The SBC generates a refresh SUBSCRIBE before the subscription expires.

If the call negotiates to RFC 2833, no KPML interworking is performed.

The following diagram shows the standard case where KPML to RFC 2833 interworking is performed.



KPML to RFC 2833 translation is also supported when used in conjunction with SDP insertion on the KPML side of the call. For example:



> **Note:**
>
> Oracle Communications Session Border Controller does not support multiple m lines with KPML - 2833 interworking.

RFC 2833 to KPML

RFC2833 to KPML interworking requires that the INVITE request or response's does not contain `Allow-Event: kpml`. When sending an INVITE when RFC2833 interworking applies, an `Allow-Event: kpml` header is added to the INVITE when the message's next hop on egress is either :

- A session agent with **kpml-interworking** set to enabled
- A session agent with **kpml-interworking** set to inherited from the SIP interface (with **kpml-interworking** set to enabled)
- Not a session agent, and the SIP interface the message is sent from has **kpml-interworking** set to enabled.

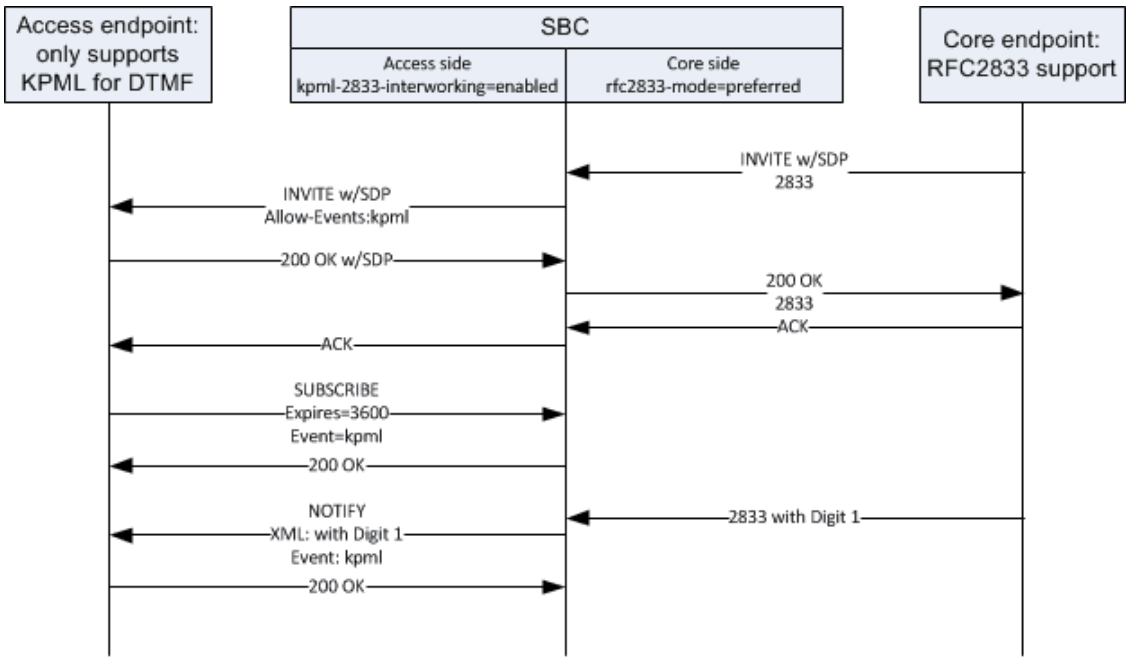
If the scenario passes the above test and the `Allow-Event: kpml` header is added to the INVITE:

When the SBC subsequently receives a SUBSCRIBE request within the INVITE created dialog for the kpml event, it accepts the subscription and responds with a 200 OK. At this point, the SBC can generate a KPML NOTIFY request with a KPML digit corresponding to each 2833 `telephone-event` received from the far end until:

- The INVITE dialog is terminated due to a BYE;
- The subscription expires; or
- The subscription is terminated with a subscribe request `Expires: 0`.

The following diagram shows a typical RFC2833 to KPML interworking call flow. Note the following:

- While the example has SDP in the INVITE, delayed offer scenarios where the SDP exchange occurs in the 20 0OK and the ACK are supported
- The **rfc2833-mode** parameter in the in either the SIP interface or session agent, is considered in the interworking. See next section:
- If the call negotiates to RFC 2833 to RFC 2833, no KPML interworking is performed.
- In RFC 2833 to KPML scenarios, if the SUBSCRIBE is not received quickly enough, RFC 2833 digits are dropped.



Originator	Terminator	SBC Behavior
rfc2833-mode=transparent, offers 2833 SDP	rfc2833-mode=transparent, answers NO 2833 SDP	Forwards 2833 offer, Adds `Allow-Event: kpml`, passes non-2833 capability to Originator (no 2833 support)
rfc2833-mode=preferred, offers 2833 SDP	rfc2833-mode=transparent, answers NO 2833 SDP	Forwards 2833 offer, Adds `Allow-Event: kpml`, inserts 2833 capability to Originator, performs 2833 to KPML translation
rfc2833-mode=transparent, offers 2833 SDP	rfc2833-mode=preferred, answers NO 2833 SDP	Forwards 2833 offer, Adds `Allow-Event: kpml`, passes non-2833 capability to Originator (no 2833 support)
rfc2833-mode=preferred, offers 2833 SDP	rfc2833-mode=preferred, answers NO 2833 SDP	Forwards 2833 offer, Adds `Allow-Event: kpml`, inserts 2833 capability to Originator, performs 2833 to KPML translation

Additional Configuration Consideration

If using a session agent to establish this interworking on the DTMF/RDC2833 side, check the **rfc2833-payload** and **rfc2833-mode** values in your **session-agent** and **sip-interface**. The payload type used for RFC 2833 and mode must be the same on both the **session-agent** and **sip-interface**.

The RFC default for rfc2833 payload type is 101, but you may have configured a media profile with a different payload type. If not, you can set to **rfc2833-payload** to 101.

Specific configuration on the **session-agent** on the **sip-interface** that is supporting DTMF/RFC2833 includes:

- **rfc2833-payload**—Set this parameter either to 101, if using the default on your sip-interface, or to a value that is the same on both the **session-agent** and the **sip-interface**.

- **rfc2833-mode**—Set this parameter to transparent.

Interworking RFC 2833 and KPML for Hairpin Sessions

The SBC supports RFC 2833-KPML interworking scenarios that include forwarded calls that hairpin to an endpoint out the original interface. If the initial callee supports one of these digit encapsulation methods, and the caller and final callee support the other, the default SBC behavior of preferring RFC 2833 may block the KPML digit tranmission. You can configure the SBC to support interworking within these hairpin scenarios in the egress direction.

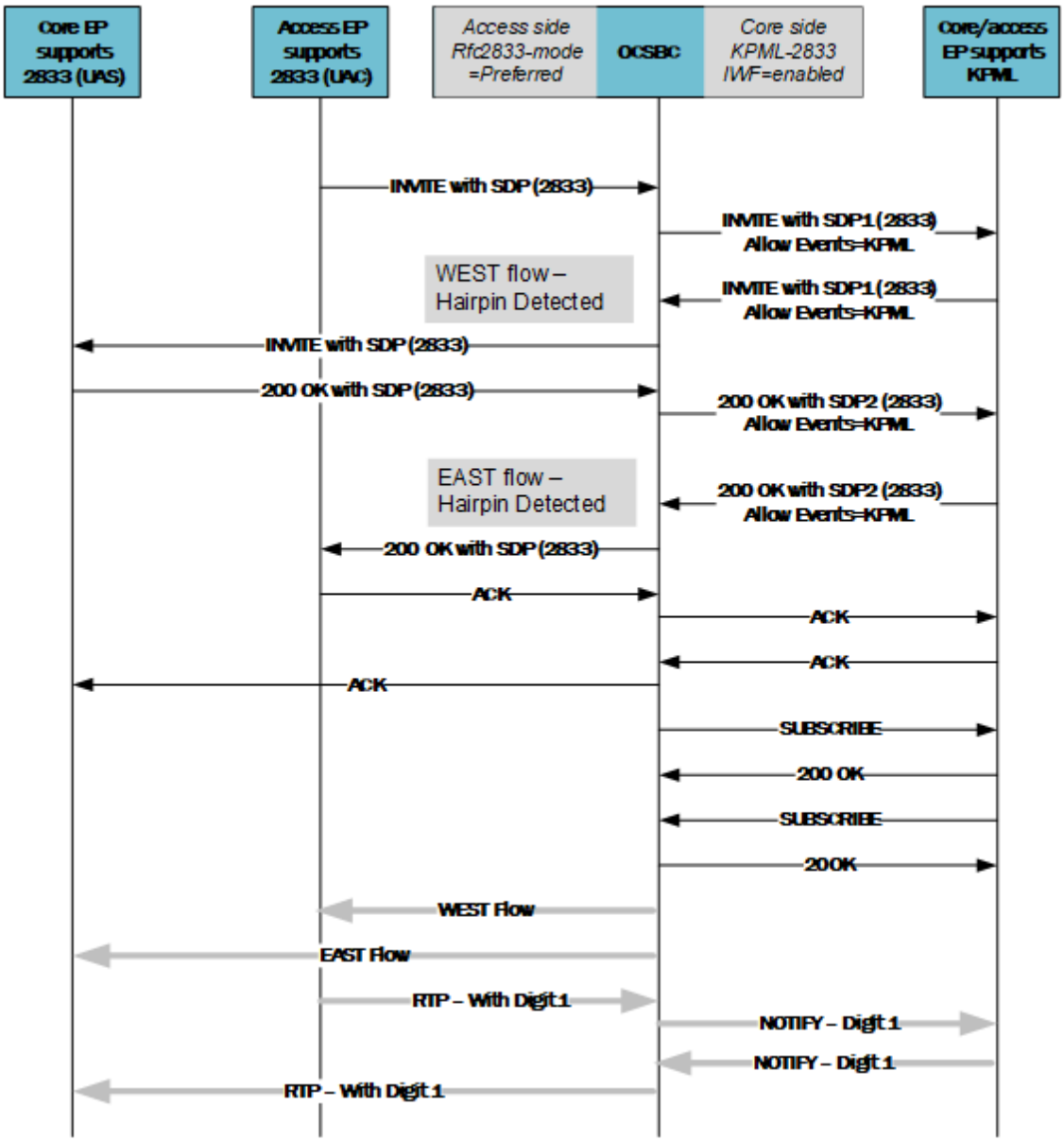
Forwarded hairpin calls can cause a problem to RFC 2833-to-KPML interworking for which the SBC can recognize and compensate. The issue exists for both RFC 2833-to-KPML and KPML-to-RFC 2833 calls, wherein a called endpoint that requires this interworking forwards the call back to the SBC, which hairpins the call to an endpoint that supports the initial encapsulation.

By default, the SBC uses RFC 2833 for digit transmission if there is an SDP attribute that includes **telephone-event**. This remains true even if there is also an **allow event** parameter set to **kpml**. The above scenario results in an SDP that includes both, causing the SBC to present digit encapsulation towards the KPML endpoint within RFC 2833 RTP.

To compensate for environments that need to support this hairpin forwarding, you can enable the **kpmlRFC2833-iwf-on-hairpin** parameter on the applicable session-agent or sip-interface. When configured for this hairpin support:

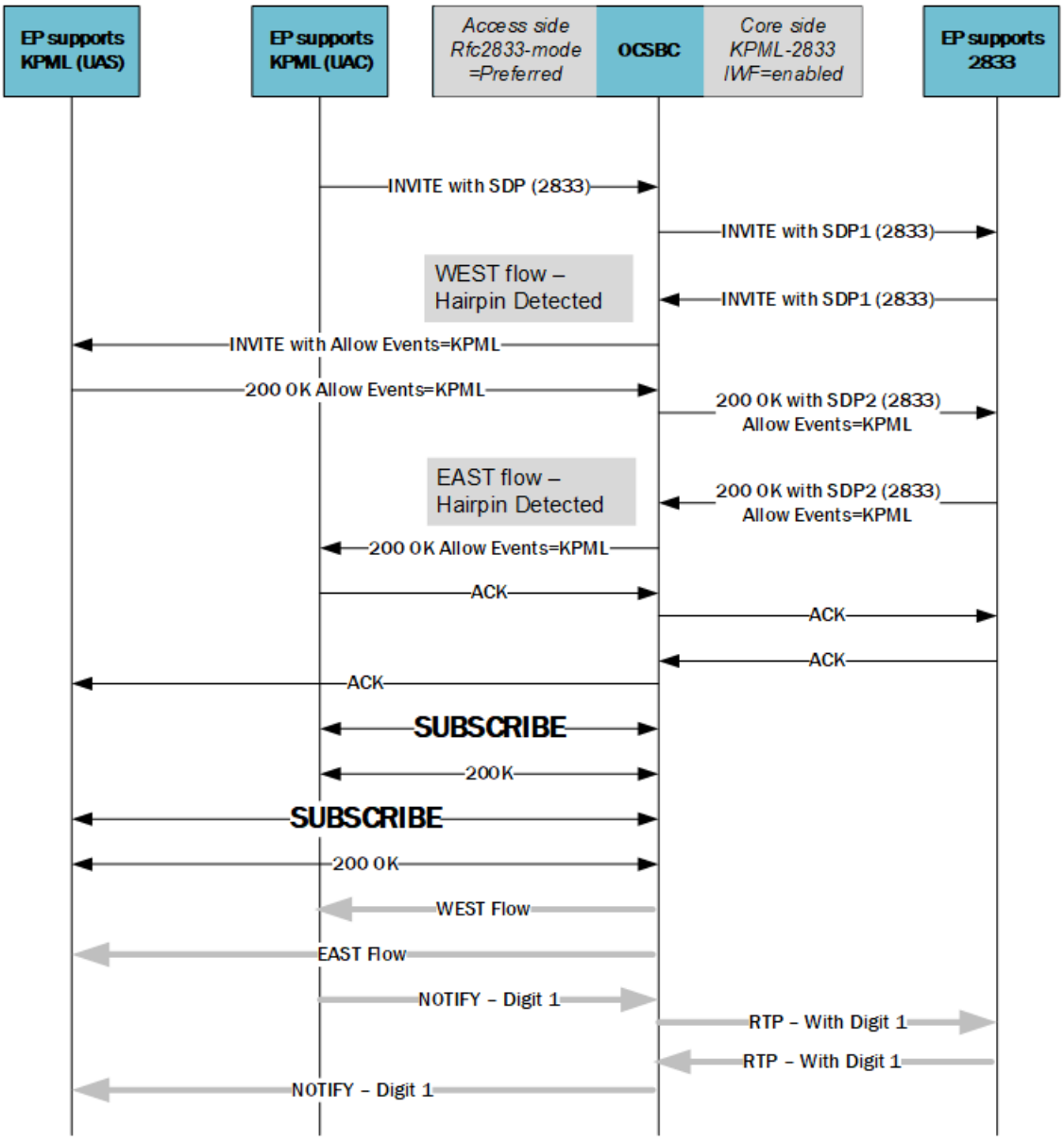
- RFC 2833-to-KPML with hairpin to a RFC 2833 endstation—Although the SBC receives an SDP answer that supports both KPML and RFC 2833 from KPML side, it honors the SUBSCRIBE request from the KPML endpoint and generates NOTIFY requests with DTMF digits towards the KPML end point.
- KPML-to-RFC 2833 with hairpin to a KPML endstation—Although the SBC receives an SDP offer that supports both KPML and telephone event from the KPML endpoint, it sends NOTIFY requests with the DTMF digits towards the RFC 2833 end point.

The diagram below shows an endpoint supporting RFC 2833 initiating a call, which terminates via hairpin on another RFC 2833 endpoint. The call initially targets an endpoint supporting KPML, but is forwarded back to the SBC. SDP2 includes the KPML allow event parameter and the telephone-event attribute. But this configuration causes the SBC to send digits to the KPML endpoint using NOTIFY message, and encapsulate those same digits in RTP for the RFC 2833 endpoints.



Bi-Directional Subscriptions

Within the context of hairpinned sessions that starts from a KPML endpoint and ultimately targets a KPML endpoint, the SBC both accepts the **SUBSCRIBE** from the called endpoint, and sends a **SUBSCRIBE** to the calling endpoint. In the diagram below, The SBC hairpins a call originating from a KPML endpoint, towards an RFC 2833 endpoint, and back out the initial interface to another KPML endpoint. Within the context of the **INVITE** signaling, the SBC issues and receives KPML **SUBSCRIBE** messages from both endpoints. Given the preference outlined in RFC 4370 that KPML subscriptions be unique to each **DIALOG**, the SBC monitors the local tag (From:) to discriminate between subscriptions, thereby supporting bi-directional subscriptions.



Configuration Considerations

You configure this support by enabling the **kpmlRFC2833-iwf-on-hairpin** parameter on the applicable **sip-interface** and/or **session-agent**.

Important configuration considerations include:

- Set the **rfcRFC 2833-mode** parameter to **preferred** on the leg with the RFC 2833 endpoint.
- Set the **kpml-interworking** and **kpmlRFC2833-iwf-on-hairpin** parameters to **enabled** on the leg with the KPML end point.
- The **session-agent** configuration takes precedence.
- If you target a configured session-agent and configure this interworking on the **sip-interface**, configure **kpml-interworking** to **inherit** on the **session-agent**.

KPML-2833 Interworking on a SIP Interface Configuration

To configure KPML - 2833 interworking on a SIP interface:

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **kpml-interworking**—Set this parameter to enabled to use KPML-2833 interworking.
4. Type **done** to save your configuration.

KPML-2833 Interworking on a Session Agent Configuration

To configure KPML - 2833 interworking on a session agent:

Enter the prerequisites here (optional).

Enter the context of your task here (optional).

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. Select the **session-agent** object to edit.

```
ORACLE(session-agent)# select
<hostname>:
1: 192.168.100.101:1813

selection: 1
ORACLE(session-agent)#
```

3. **kpml-interworking**—Set this parameter to enabled for the Oracle Communications Session Border Controller to interwork RFC2833 from the other call leg to the call leg running through this session agent. This parameter may be set to inherit to use the **kpml-interworking** value configured on the receiving SIP interface.
4. Type **done** to save your configuration.

22

RCS Services

Message Session Relay Protocol

The Oracle Communications Session Border Controller (SBCsupports Message Relay Protocol (MSRP) sessions initiated by Session Description Protocol (SDP) messages exchanged through the Session Initiation Protocol (SIP) offer and answer model. MSRP usage with SDP and SIP is described in Section 8 of RFC 4975, The Message Relay Protocol. The SBC functions as a Back-to-Back User Agent (B2BUA) for MSRP sessions, terminating incoming MSRP, proxying for the MSRP session originator, initiating outgoing MSRP to the endpoint peer, and providing Network Address Translation (NAT) services.

The Oracle Session Border Controller (SBC) supports the re-creation of a Message Session Relay Protocol (MSRP) session after a connection interruption, as specified in section 5.4 of RFC 4975. A User Agent engaged in an MSRP session with the SBC can send a reINVITE to the SBC to set up a new MSRP session to replace the existing MSRP session when the TCP connection is interrupted, disconnected, or otherwise unresponsive.

MSRP Platform Support

All platforms except the Acme Packet 1100 support Message Session Relay Protocol (MSRP).

The SBC supports MSRP over IPv6 and IPv4-IPv6 Inter-working function for MSRP.

Entitlement Configurations for MSRP on Virtualized Platforms

To support 500 or more MSRP sessions on virtualized SBCs, in some cases you must reconfigure the entitlements. When the existing entitlements show IMS-AKA Endpoints set to non-zero value, do the following:

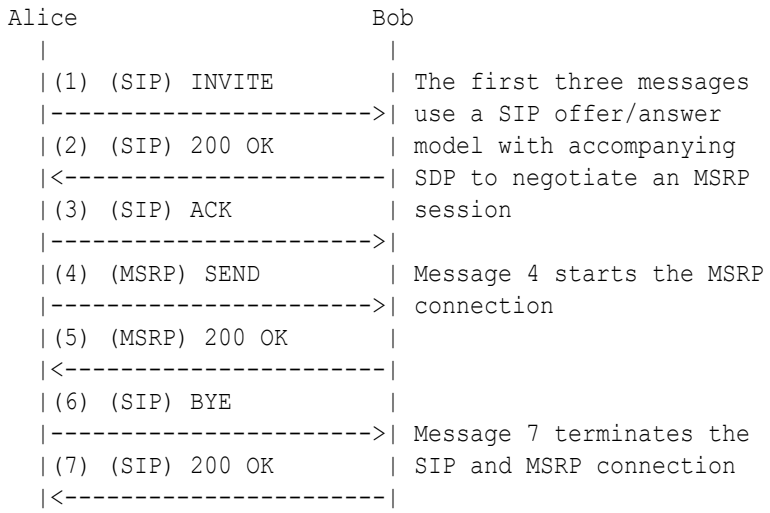
1. With the **setup entitlements** command, set IMS-AKA Endpoints to 0.
2. Perform a system reboot.
3. With the **setup entitlements** command, set MRSP B2BUA sessions to a number greater than 499.

MSRP IP Address Family Support

The Oracle Communications Session Border Controller supports MSRP over IPv4 and IPv6. The Oracle Communications Session Border Controller also can perform IPv4-to-IPv6 and IPv6-to-IPv4 interworking. This support is available automatically and does not require any configuration.

MSRP Operational Description

A sample RFC 4975-compliant Offer/Answer SDP exchange for an MSRP session is shown below.



1. Alice sends an INVITE request with accompanying SDP to Bob.

   The SDP media (M) line is defined in RFC 4975, and adheres to the format

   m=<media> <port> <protocol> <format-list>

   MSRP operations require the following values:

   - media—message
   - protocol—TCP/MSRP (for an unencrypted connection)
   - format-list—*
   - port—the TCP port (7777 in the SDP example, although any valid port number can be specified) monitored by the message originator for a response to the SDP offer

   The required SDP attributes, accept-types and path, are also defined in RFC 4975.

   accept-types contains a list of media types that the message originator is willing to receive. It may contain zero or more registered media-types, or an * wildcard character in a space-delimited string.

   path contains the MSRP URI of the message originator. An MSRP URI is constructed as shown below.

   a. scheme
      - msrp (for an unencrypted connection), or
      - msrps (for an encrypted connection)
   b. //
   c. address
      - IP address of the message originator, or
      - FQDN of the message originator
   d. ;

e. port
   - the port (7777 in the SDP example, although any valid port number can be specified) monitored by the message originator for MSRP responses

f. session-id
   - a random local value generated by the message originator used to produce an ephemeral MSRP URI lasting only for the duration of the current MSRP session

g. ;

h. protocol
   - tcp

```
Alice->Bob (SIP):

INVITE sip:bob@example.com
SDP:
v=0
o=alice 2890844557 2890844559 IN IP4 alicepc.example.com
s=
c=IN IP4 alicepc.example.com
m=message 7777 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://alicepc.example.com:7777/iau39soe2843z;tcp
```

2. Bob accepts the SDP offer, generates a local session-id (contained in his MSRP URI specified by the path attribute), and issues a 200 OK response to Alice.

   The port parameter in the Media line indicates that Bob listens for MSRP messages on TCP port 8888

```
Bob->Alice (SIP):

SIP/2.0 200 OK
SDP:
v=0
o=bob 2890844612 2890844616 IN IP4 bob.example.com
s=
c=IN IP4 bob.example.com
m=message 8888 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://bob.example.com:8888/9di4eae923wzd;tcp
```

3. Alice ACKs Bob's answer, establishing a SIP session between the two.

```
Alice->Bob (SIP):

ACK sip:bob@example.com
```

4. Alice initiates an MSRP session with an MSRP SEND request to Bob.

   All MSRP requests begin with the MSRP start line, which contains three elements.

   - protocol-id—MSRP
   - transaction-id—an ephemeral transaction identifier (d93kswow in the following MSRP example) used to correlate MSRP requests and responses, and to frame the contents of the MSRP message

- method—SEND (MSRP method that supports data transfer)

The MSRP start line is followed by the To-Path and From-Path headers, which contain destination and source addresses — the MSRP URIs exchanged during the MSRP negotiation.

The Message-ID header contains a random string generated by the message originator. This ephemeral value whose lifetime is measured from message start to message end, is used to correlate MSRP status reports with a specific message, and to re-assemble MSRP message fragments (chunks in MSRP terminology).

The Byte-Range header contains the message length, in bytes, and the specific byte range carried by this message. Contents of this header are generally of interest only if the message has been fragmented.

The Content-Type header describes the message type, and must conform to the results of the MSRP negotiation.

The actual message follows the Content-Type header.

Finally, the SEND request is closed with an end-line of seven hyphens, the transaction-id, and a

$ to indicate that this request contains the end of a complete message, or

+ to indicate that this request does not contain the message end

```
Alice->Bob (MSRP):

MSRP d93kswow SEND
To-Path: msrp://bob.example.com:8888/9di4eae923wzd;tcp
From-Path: msrp://alicepc.example.com:7777/iau39soe2843z;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-16/16
Content-Type: text/plain
Hi, I'm Alice!
-------d93kswow$
```

5. Bob acknowledges receipt with an MSRP 200 OK response to Alice.

   Note that the response includes the initiator-originated transaction-id, d93kswow.

```
Bob->Alice (MSRP):

MSRP d93kswow 200 OK
To-Path: msrp://alicepc.example.com:7777/iau39soe2843z;tcp
From-Path: msrp://bob.example.com:8888/9di4eae923wzd;tcp
-------d93kswow$
```

6. Alice sends a BYE request to Bob.

   Alice sends a BYE request to terminate the SIP session and MSRP sessions. Alice can, of course, send more SEND requests to Bob before sending the BYE.

```
Alice->Bob (SIP):

BYE sip:bob@example.com
```

7. Bob sends a 200 OK response to Alice.

```
Bob->Alice (SIP):

SIP/2.0 200 OK
```

The SIP session and the MSRP session are terminated.

Secure MSRP Session Negotiation

An Offer/Answer SDP exchange for a TLS (secure) MSRP session is specified in RFC 4572, Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP). RFC 4572 defines the syntax and symantics for an SDP fingerprint attribute that identifies the certificate (most likely a self-signed certificate) that will be presented during the TLS negotiation. A sample SDP exchange is shown below.

The protocol field (TCP/TLS/MSRP) of the media (M) line designates a TLS encrypted connection.

The fingerprint attribute is constructed as follows:

- protocol—identifies the hashing method used to produce the certificate fingerprint, SHA-1 in the following sample SDP
- hash value—a sequence of uppercase hexadecimal bytes separated by colons with the sequence length determined by the hash function

Offer SDP: (Alice to Bob)

```
v=0
o=usera 2890844526 2890844527 IN IP4 alice.example.com
s=
c=IN IP4 1.1.1.1
m=message 7394 TCP/TLS/MSRP *
a=accept-types:text/plain
a=path:msrps://alice.example.com:7394/2s93i9ek2a;tcp
a=fingerprint:SHA-1
4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Answer SDP: (Bob to Alice)

```
v=0
o=userb 2890844530 2890844532 IN IP4 bob.example.com
s= -
c=IN IP4 2.2.2.2
t=0 0
m=message 8493 TCP/TLS/MSRP *
a=accept-types:text/plain
a=path:msrps://bob.example.com:8493/si438dsaodes;tcp
a=fingerprint:SHA-1
DA:39:A3:EE:5E:6B:4B:0D:32:55:BF:EF:95:60:18:90:AF:D8:07:09
```

MSRP Session Setup

The B2BUA processes MSRP media descriptions in offer/answer SDPs to negotiate and establish MSRP sessions and then constructs internal data flows for the actual MSRP sessions. After establishing an MSRP session, the B2BUA forwards MSRP requests and responses from and to the session participants.

Initiating MSRP Sessions

After accepting an offer SDP with MSRP session initiation, the B2BUA constructs an egress offer SDP as follows.

1. The B2BUA sets the transport protocol of the m= line to the transport protocol of the selected egress profile.
2. If the value of listen-port of the selected egress profile is not zero, the B2BUA sets the port of the m= line to the value of listen-port. If the value of listen-port is zero, the port in the m= line is chosen from a steering port of the egress realm.
3. The B2BUA adds an a=setup attribute to the SDP. The value of this attribute is determined by the value of the **preferred-setup-role** ACLI command. For example, if the value of **preferred-setup-role** is passive (the default value) the B2BUA adds the attribute a=setup:passive.
4. The B2BUA performs NAT on the MSRP Universal Resource Identifier (URI) in the a=path attribute.

   The B2BUA does not include an a=fingerprint in the offer SDP if the selected egress profile has TCP/TLS/MSRP transport protocol. However, if the egress profile specifies both TCP/MSRP and TCP/TLS/MSRP the B2BUA selects the TCP/TLS/MSRP transport protocol, resulting in an egress offer containing an m= line with TCP/TLS/MSRP transport protocol. The B2BUA offers only a single media line, TCP/MSRP or TCP/TLS/MSRP. The B2BUA does not perform recursion (that is, first initiation attempt with TCP/TLS/MSRP and re-attempt with TCP/MSRP if first attempt is rejected).

Connection Negotiation

In compliance with RFC4145 and RFC6135, the SBC can act as offerer or answerer when using SDP to negotiate MSRP sessions. As answerer, the SBC receives MSRP signaling from a UA that wants to start an MSRP session. As offerer, it acts as an MSRP B2BUA and starts an MSRP session. Within a connection-oriented media architecture, SDP negotiates these roles from the a=setup parameter, which determines which end station is responsible for initiating the session as active, and which is passive.

As an MSRP session offerer, a user agent client can follow RFC 4145 and include an a=setup attribute in a MSRP media line, or it may follow the default connection direction specified in RFC 4975, which specifies that the endpoint that sent the original offer is responsible for connecting to its remote peer. The B2BUA, in contrast, always includes an a=setup attribute in its SDP offer. You can configure this setup value on the SBC by configuring the **preferred-setup-role** parameter within the **tcp-media-profile**, **profile-list** element. Parameter values include:

- **active**—The SBC explicitly requests the active role.
- **passive**—The SBC sends actpass to the UA as its setup role. This allows the SBC to be either active or passive, depending on the response from the UA. If the UA responds requesting active, the SBC takes the passive role, and vice-versa. This actpass value ensures that the SBC can either:

  - Initiate an outgoing connection to a remote UA
  - Accept an incoming connection from a remote UA

  The absence or value of the “a=setup” attribute in the answer SDP received from the remote UA determines the actual role performed by the SBC, as specified by RFC 4145.

  - When the SBC receives “a=setup:active”, it performs the passive role, listening on the advertised port.
  - When the SBC receives “a=setup:passive”, it performs the active role.
  - When the SBC receives an offer SDP with the attribute a=setup:actpass attribute, it sets the a=setup attribute of its SDP answer to the value set in the **preferred-setup-role** parameter.
  - When the SBC receives an answer SDP that does not include an a=setup attribute, it assumes that the user agent server does not support connection negotiation per RFC 4145, takes the active role as specified in RFC 4975, and makes the outgoing connection.

The default and recommended **preferred-setup-role** configuration is **passive**, which allows the remote UA to choose its role.

Whether taking the active or passive role on the caller side (ingress), the SBC initiates an outgoing connection towards the callee (egress) on-demand and its MSRP requests to the callee. Again, the SBC as B2BUA sends its SDP offer with the setup line using its configured value.

Key behavior also includes:

- When active, the SBC refers to the applicable **tcp-media-profile**, **profile-list** to see if you have configured the **listen-port**. If you have configured a **listen-port**, the SBC listens for traffic on that port. By default, the **listen-port** is set to 0, which causes the SBC to listen using a port it allocates from the **steering-pool** of the applicable realm.
- When it is the answerer, the SBC is forced into a specific role and does not always use the configured **preferred-setup-role** value. The SBC uses the actpass value to negotiate in scenarios where:
  - The remote UA is negotiating the connection using the passive value or,
  - When the remote UA is using passive improperly during the negotiation, which is prohibited by RFC 6135.
- Sending “a=setup:actpass” value in an offer from the SBC is also compliant if the remote UA does not support the COMEDIA mechanism described in RFC 4145, and is, therefore, always a passive endpoint. If the remote UA sends back an answer with SDP that indicates the role it is going to take, the SBC takes the other role.
- When configured to propose actpass, via the **passive** setting, it is possible that SBC could end up performing the active role on both the ingress and egress sides of the session. In this case, there may not be any MSRP request received from the UA on one side of the SBC to trigger the outgoing connection on the other side. It is, therefore, mandatory for the SBC to establish a connection right after successful negotiation of offer/answer exchange and send an empty MSRP request on the established connection. This behavior is in compliance with RFC 4145.
- The SBC initiates the egress outbound TCP connection right after it receives the answer SDP from the ingress remote UA. The TCP initiation processing from the data path happens when the host sends a MSRP session provision message to the data path after receiving the answer SDP. From the call flow perspective, this outbound TCP initiation by the SBC can happen before it sends any SIP ACK message.

SBC as Answerer

The table below shows the values the SBC inserts into its SDP setup attribute when it answers an offer from a remote UA.

Offer	Answer	Rationale
Active	Passive	The SBC is forced into this role so it can comply with the section 4.1 of RFC 4145.
Passive	Active	The SBC is forced into this role so it can comply with the section 4.1 of RFC 4145.
Actpass	Use configured value	The user configures the SBC with an appropriate value according to the network requirement in which it is deployed.
No “a=setup” attribute	Passive	Since the remote UA does not support RFC 4145, per RFC 4975, the SBC takes the passive role as the answerer.

SBC as Offerer

The table below shows the values the SBC inserts into its SDP setup attribute when it sends an offer to a remote UA. When initiating an SDP offer, the SBC prefers to use the configured value of the **preferred-setup-role** parameter. As shown in the table, this is not always the case.

Configured Preferred-setup-role	Offer	Answer	Final role performed by the SBC	Rationale
Active	Active	Passive	active	The SBC uses the configured value in its offer SDP, per RFC 4145.
Passive	Actpass	Active/Passive	Opposite role of what comes in the answer SDP	RFC 6135 prohibits the use of “a=setup:passive”, So the SBC offers “a=setup:actpass”. This allows the remote UA to choose the role. *
Active/Passive	Active/actpass	No “a=setup” attribute	active	The remote UA does not support RFC 4145. As offerer, the SBC takes the active role, per RFC 4975.

* Note the second row. Even though you configured the **preferred-setup-role** to passive, the SBC uses a=setup:actpass when it sends an offer to a remote UA. The remote UA may:

- If compliant with RFC 6135, send an answer SDP with “a=setup:active” because the UA knows that it is located behind a NAT.
- If the remote UA knows that it is not behind a NAT, it should send an a=setup:passive.

Reporting on MSRP Session Setup

Key reporting on MSRP setup signaling using the **show msrp statistics** command includes:

- Total Requests Sent—When the SBC successfully sends the MSRP SEND request, it increments the **show msrp statistics** counters.
- Total message send failure—In case of failure to send out this message for any reasons, then SBC increments this counter in CDR/ACR information. (The current implementation does not display this in the **show msrp statistics** command).

Also, the SBC includes the transmitted bytes for this message and the MSRP SEND request count for the calling party in MSRP CDR and diameter ACR data.

When the SBC issues an MSRP SEND message with no body. it includes all mandatory header fields, including the following header fields with specific values:

- Success—Report header field with a value of “no”.
- Failure—Report header field with a value of “no”.

This message prevents the receiving MSRP UA from sending any response to this request or REPORT request back to the SBC.

Specifying the Connection Delay Timer

When the SBC takes the active role, it can initiate an MSRP TCP/TLS outbound connection towards the remote MSRP UA immediately after it receives the answer SDP. But the remote UA may not be ready to accept the MSRP connection request on its advertised IP/port right away. Per RFC 3264, a SIP UA should listen on its advertised IP and port immediately to receive connections, but there are end stations that do not comply with this. To alleviate the risk of failed sessions, you can configure the **conn-setup-delay-timer** parameter under the **msrp-config** within the **media-manager** element to wait the configured number of milleseconds before initiating the outbound connection.

When setting up sessions with UAs that are not compliant in this respect, the TCP connection attempt from the SBC can fail. This can happen if the SBC receives the TCP-RST from the remote UA for the SYN it sent, and the stops attempting the connection. This presents the risk of experiencing SIP session and MSRP connection termination, for example, when the flow guard timer expires. A remote UA cannot send any MSRP (instant) message towards the other participant through the SBC until the SBC successfully establishes the connection.

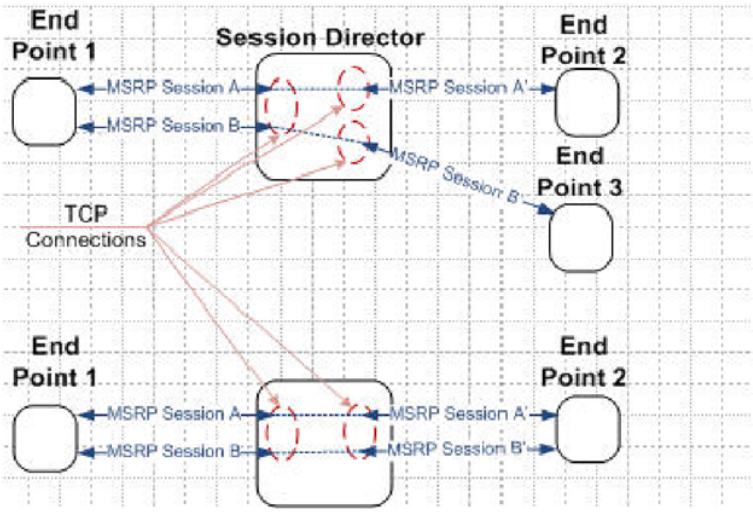
You can avoid this issue and provide flexibility for non-standard UAs by configuring the **conn-setup-delay-timer** parameter. This setting delays the initiation of the outbound TCP connection towards the remote UA until the SBC takes the active role. The default is 0 and the range is from 0 to 1500 milleseconds.

Configure this value if you have UA’s that do not listen on advertised ports within this window to ensure the SBC:

- Delays TCP initiation.
- Buffers any data received on the other call leg, waiting for the connection to be established.
- Sends this data after the timer expires.

Multiple MSRP Connections

The MSRP B2BUA supports sharing of a single TCP connection by multiple MSRP sessions. In such a topology, each TCP connection maintains a list active MSRP sessions.



With regard to the above figure, the upper Oracle Communications Session Border Controller's TCP connection between Endpoint 1 and the MSRS B2BUA is shared by 2 MSRP sessions. At bottom, each MSRP session uses a separate TCP connection. When the B2BUA assumes the active role, it always initiate a separate connection to the MSRP endpoint peers, endpoints 2 and 3 as shown above.

When the list of active MSRP sessions for a shared TCP connection becomes empty as a result of SIP session terminations or disconnections of peer TCP connections, B2BUA disconnects the shared TCP connection. If the shared connection is disconnected by the peer, the B2BUA disconnects all the separate TCP connections it initiated for the MSRP sessions that use the shared connection.

Accepting Connections

When the B2BUA is in passive mode, it listens for incoming connection from the active party, monitoring the port specified by the **listen-port** ACLI command.

Making Connections

When the B2BUA is in active mode, it makes the connection to the passive party, selecting a port allocated from the steering-pool of the applicable realm.

MSRP Session Termination

An MSRP session is terminated by sending or receiving a BYE request in the parent SIP session, that is the session that set up the MSRP exchange, followed by the disconnect of the TCP connection that supports MSRP message exchange.

Some SIP endpoints close their MSRP TCP connections upon receiving a BYE possibly before its peer finishes sending all the MSRP messages and closes the connection. To facilitate graceful session completion, the B2BUA offers a configurable time delay between the receipt

of a BYE request from an MSRP endpoint and the transmission of the BYE to the recipient endpoint peer.

With the configurable BYE delay enabled, the MSRP B2BUA upon receiving a BYE request acknowledges the request with a 200 OK response. The B2BUA, however, does not immediately forward the BYE to the other MSRP endpoint.

Rather, the B2BUA triggers two user-configurable timers that monitor the specific MSRP session initiated by the current SIP exchange. The first timer measures inactivity intervals on media flows (calling-to-called and called-to-calling) associated with the MSRP session to be closed. The second timer sets an unconditional outer limit at which point the delayed BYE is transmitted to the MSRP endpoint and the MSRP is terminated.

Expiration of either timer generates an internal stop event which generates transmission of the delayed BYE to the MSRP endpoint and termination of the underlying SIP connection.

Note that the MSRP-specific timers, the session inactivity timer and the MSRP delayed-BYE timer, are roughly analogous to two existing TCP timers, the TCP subsequent guard timer and the TCP flow time limit timer. The following sections summarize timer operations.

MSRP interval timer

- Purpose: Measures inactivity periods within MSRP data sessions. The timer is triggered in the absence of MSRP data. If new MSRP is not detected prior to timer expiration, a stop event is generated resulting in delayed BYE transmission and MSRP connection termination. If new MSRP traffic is detected prior to timer expiration, the timer is reset.
- Associated ACLI Command: **session-inactivity-timer**
- Allowable command values: 0 | 5 to 10 (seconds). When set to 0, session monitoring is disabled. No MSRP session monitoring is done when the corresponding SIP session receives a BYE request.
- Default value: 5
- Timer Expiration: Initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the SIP connection.

TCP subsequent guard timer

- Purpose: Measures the maximum interval allowed between media-over-TCP packets.
- Associated ACLI Command: **tcp-subsq-guard-timer**
- Allowable command values: 0 to 999999999 (seconds)
- Default value: 300
- Timer Expiration: Initiates tear down of the TCP connection. In the possible, but unlikely event that the value assigned to this timer is less than the value assigned to the MSRP interval timer, expiration initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the TCP connection.

MSRP delayed -BYE timer

- Purpose: Measures inactivity periods within MSRP traffic sessions. The timer is triggered in the absence of MSRP data. If new MSRP is not detected prior to timer expiration, a stop event is generated resulting in delayed BYE transmission and MSRP connection termination. If new MSRP traffic is detected prior to timer expiration, the timer is reset.
- Associated ACLI Command: **msrp-delayed-bye-timer**

- Allowable command values: 0 to 60 (seconds)
- Default value: 15
- Timer Expiration: Initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the SIP connection.

TCP flow time limit timer

- Purpose: Measures the maximum allowed lifetime of a media-over-TCP connection.
- Associated ACLI Command: **tcp-flow-limit-timer**
- Allowable command values: 0 to 999999999 (seconds)
- Default value: 86400 (1 day)
- Timer Expiration: Initiates tear down of the TCP connection. In the possible, but unlikely event that the value assigned to this timer is less than the value assigned to the MSRP delayed-BYE timer, expiration initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the TCP connection.

Network Address Translation

If the value of the **uri-translation** ACLI command is **enabled**, the B2BUA performs network address translation on the MSRP URI in the a=path attribute and in the From-Path and To-Path of MSRP requests and response. The host part of the URI will be the IP address of the steering pool of the realm. The port will be chosen as follows:

If the B2BUA role is in passive mode, and the listen-port ACLI command is non-zero, the B2BUA monitors that specified port.

If the B2BUA role is in passive mode, and the listen-port ACLI command is zero, the B2BUA selects a port from the steering pool of the applicable realm and monitors that chosen port.

If the B2BUA role is in active mode, the port is chosen from the steering pool of the applicable realm. Since B2BUA is active, that port is used only to support the outgoing connection.

Double Steering Pool Port Allocation

You can enable the **double-port-allocation** parameter in the **msrp-config** to make the SBC use two ports per MSRP call instead of one on a per-realm basis.

Recall that you can configure a realm with multiple steering-pools and set a strategy for each pool. The SBC creates a port list from which it selects ports on a per-call basis. By default, the SBC assigns the first two ports that are sequential for audio calls and single ports for MSRP calls. When a call is completed, the SBC puts the newly available port (or ports) at the top of the list. Because calls can terminate at any time, the system may re-populate the available port list with non-consecutive port pairs.

You can enable this parameter to make the system assign two sequential ports for MSRP calls as well as audio calls. This results in the system re-populating its available port list with consecutive port pairs for each audio and MSRP call.

Consider the SBC behavior with respect to **double-port-allocation** configuration and port allocation based on call type:

- **double-port-allocation** disabled:
  - Audio Calls:

        - The system uses ports from **pair** strategy pools first
        - When **pair** port pools are not configured or exhausted, the system selects ports from **mixed** strategy pools
        - The system never uses ports from **single** strategy pools
    - MSRP Calls:
        - The system uses ports from **single** strategy pools first
        - When **single** port pools are not configured or exhausted, the system selects ports from **mixed** strategy pools
        - The system never uses ports from **pair** strategy pools
    - **double-port-allocation** enabled:
        - Audio Calls—The system does not change is port pool selection order
        - MSRP Calls:
            - The system uses ports from **pair** strategy pools first
            - When **pair** port pools are not configured or exhausted, the system selects port from **mixed** strategy pools
            - The system never uses ports from **single** strategy pools

As shown above, the SBC effectively eliminates the use of **single** strategy pools in realms with **double-port-allocation** enabled, ensuring the available port list always uses two consecutive ports to support audio and MSRP calls.

Oracle recommends that you do not use existing **mixed** strategy pools within realms when you enable **double-port-allocation** on a realm. Instead, use the following procedure so that the existing unordered list becomes a consecutive port list:

1. Enable **double-port-allocation**.
2. Delete the **mixed** strategy pool(s) from the applicable realm.
3. Save and activate your configuration.
4. Recreate the deleted pool(s) from scratch using the same port range and strategies.
5. Save and activate your configuration again.

This process enables the system to clear the available port list, create a fresh available consecutive port list, and start using the port allocation rules above for when **double-port-allocation** is enabled. You enable this feature using the syntax below:

```
Oracle (msrp-config)# double-port-allocation enabled
```

Certificate Fingerprint

If the value of the field **require-fingerprint** in the ingress and/or egress tcp-media-profile is enabled, and the transport protocol is TCP/TLS/MSRP the B2BUA requires the offer and/or the answer SDPs, respectively, to have an a=fingerprint attribute as specified in RFC 4572, Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP).

If the B2BUA receives an offer SDP without a=fingerprint attribute, it rejects the offer SDP. If the rejected SDP is in an INVITE request, the B2BUA issues a 488 Not Acceptable Here response. If the rejected SDP is in a 200 OK response, the B2BUA ACKs that 200 OK, sends a BYE to the server user agent, and a 503 Service Unavailable response to the client user agent.

If the B2BUA receives an answer SDP without a a=fingerprint attribute, it terminates the related SIP session. If the rejected SDP is in a 200 OK response, the B2BUA ACKs that 200 OK, sends a BYE to the server user agent, and a 503 Service Unavailable response to the client user agent. If the rejected SDP is in an ACK, the B2BUA simply sends a BYE to both the server and client user agent.

If the value of the field **require-fingerprint** in the profile is disabled, the B2BUA accepts offer and answer SDP that do not include an a=fingerprint attribute.

Because the B2BUA certificate is expected to be signed by a trusted Certification Authority, an a=fingerprint attribute is not included in offer and answer SDPs sent by the B2BUA.

The B2BUA maintains a fingerprint mismatch counter for each MSRP session. This counter is incremented when the B2BUA cannot match the certificate it receives during the TLS handshake with the fingerprint it receives in the SDP.

MSRP B2BUA Support for NG911

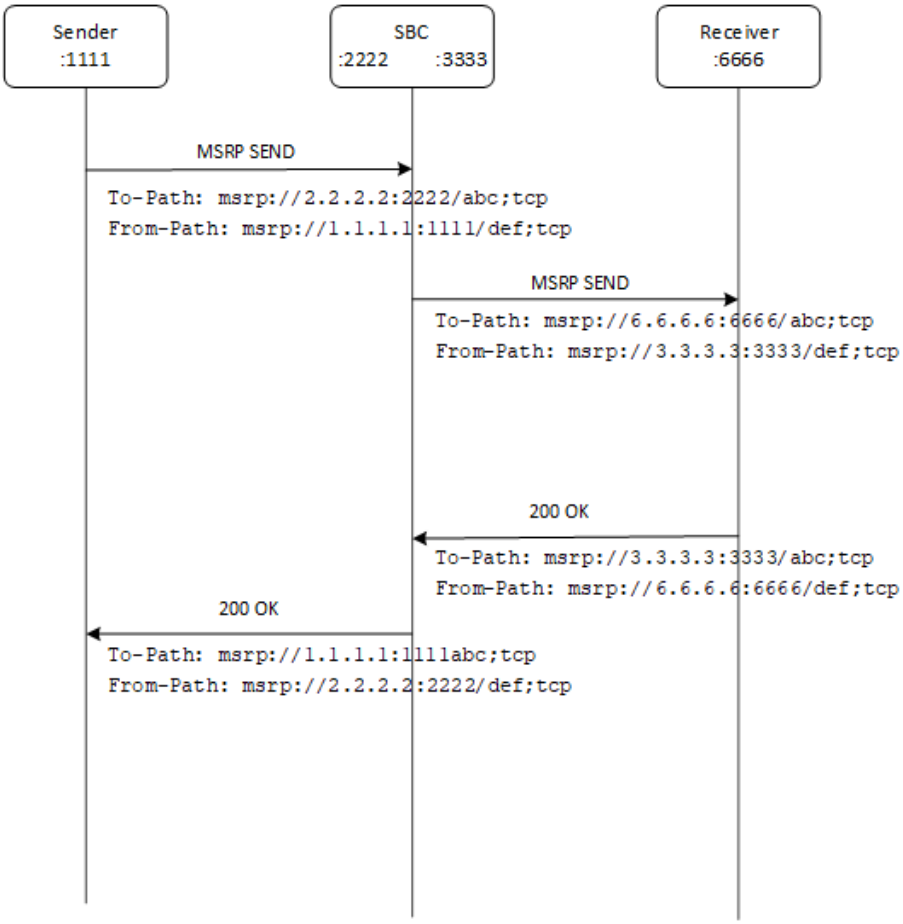
The Oracle Communications Session Border Controller (SBC) supports Message Session Relay Protocol (MSRP) and Back to Back User Agent (B2BUA) networking for Next Generation 911 (NG911). MSRP is used for messaging and file sharing in Session Initiation Protocol (SIP), which is widely used in NG911 markets that utilize MSRP with either an MSRP application server or in Peer-to-Peer (P2P) mode. Such support enables (SBC) customers to deploy the SBC as a Public Safety Answering Point (PSAP). The SBC supports this functionality only on Virtual Machines.

In P2P mode, the SBC uses MSRP, SIP, and the Session Delivery Protocol (SDP) offer-answer to establish a TCP or TLS media bearer plane between two MSRP endpoints. For MSRP B2BUA operation, the SBC terminates the bearer plane from one peer, and routes the message to the bearer of the other peer that is also anchored by the SBC.

For client-server MSRP, the SBC terminates the TCP or TLS bearer plane from the client and then connects it with a separate TCP or TLS bearer plane initiated by the SBC towards the server.

In addition to terminating and re-originating TCP or TLS for MSRP, the B2BUA modifies the To-Path and From-Path headers of the MSRP messages. The following example shows the modifications to the MSRP "To-Path" and "From-path" headers during traversal:

Figure 22-1 MSRP Flow with No Middlebox Present



MSRP and Middlebox Traversal Using the CEMA Extension and Session-ID

The Oracle Communications Session Border Controller (SBC) requires the Connection Establishment for Media Anchoring (CEMA) extension (RFC6714) and the session-id matching mechanism to allow the SBC to exchange Message Session Relay Protocol (MSRP) messages through middleboxes that do not act as MSRP Back to Back User Agents (B2BUA). When such a middlebox passes the MSRP messages through without updating the SDP a=path attribute, the SBC cannot establish a TCP connection through the middlebox. The CEMA mechanism makes the connection possible. In a scenario where the middlebox does update the SDP a=path attribute, the MSRP messages will not pass validation and will be dropped. The Session-id matching mechanism prevents that situation. The SBC supports this functionality only on Virtual Machines.

With the CEMA extension enabled, the SBC detects the presence of a middlebox that anchors the media and does not update the SDP a=path attribute by comparing the SDP c and m lines to the SDP a=path attribute. When the CEMA-enabled SBC plays the active role in establishing the TCP connection it establishes the connection to the endpoint identified by the c and m lines instead of the a=path.

Figure 22-2 Signaling Flow with a Middlebox and CEMA Enabled

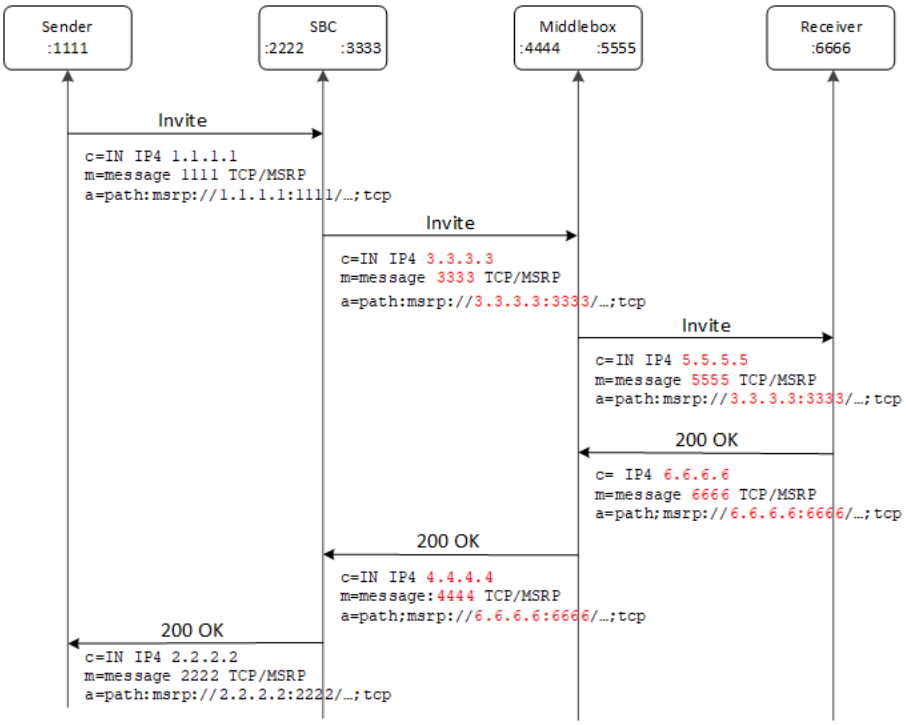
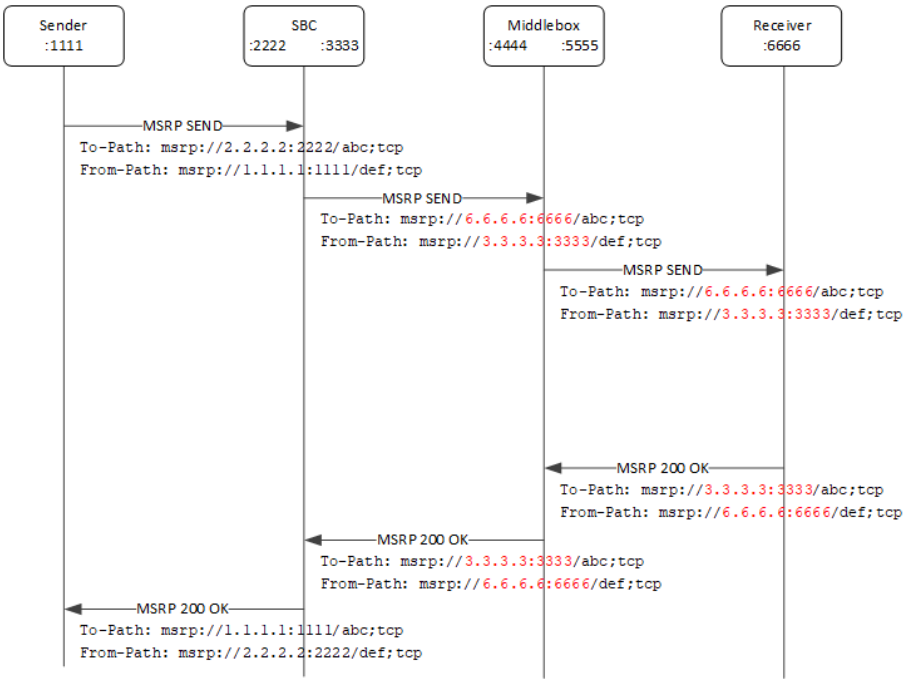


Figure 22-3 MSRP Flow with a Middlebox and CEMA Enabled



The tcp-media-profile configuration object includes the **msrp-cema-support** attribute, which you enable to allow the SBC to negotiate CEMA support with parties in a given realm.

- Disabled (default)-When playing the active role, the SBC establishes the TCP connection to the IP address and port number specified in the SDP a=path attribute of the peer. If the SDP a=path attribute contains a DNS name, the SBC attempts to use the c line. If the c line also contains a DNS name, the SBC rejects the session.
- Enabled-When the SBC detects the presence of a middlebox, it tries to negotiate the CEMA support by including the a=msrp-cema-support media attribute. When playing the active role, the SBC establishes a TCP connection to the IP address and port number indicated in the peer's SDP c and m lines rather than the a=path media attribute. If you enable **msrp-cema-support**, you must disable **msrp-sessmatch**.

> **Note:**
>
> The SBC does not perform DNS name resolution for either the SDP a=path or the c and m lines.

To-path Authority Validation

The presence of middleboxes that anchor the media and update the SDP a=path attribute to match the updated SDP c and m lines cannot be detected in the signaling plane. An MSRP B2BUA that is not enabled for CEMA correctly sets up TCP connections to the middlebox because the SDP a=path attribute points to the middlebox. Because the middleboxes do not accordingly update the MSRP message To-Path headers, MSRP messages passing through such a middlebox cannot validate because the authority part of the To-Path header does not match the authority part of the SDP a=path attribute. In such scenarios the validation of the MSRP URI is based only on the session-id part of the MSRP URI, the MSRP scheme, and transport (Session-Id matching).

To solve the problem, the tcp-media-profile configuration object includes the **msrp-sessmatch** attribute that controls whether or not the URI comparison of the To-Path header in the MSRP messages received from the respective realm includes the authority part.

- Disabled (default)-The MSRP URI comparison between the SDP a=path attribute and the To-Path header in the MSRP messages received from a realm includes the MSRP URI scheme, authority IP address, port number, session-id, and transport. If the comparison is unsuccessful and the sender requires a report, the SBC returns an MSRP 481 error response to the sender.
- Enabled-The MSRP URI comparison between the SDP a=path attribute and the To-Path header in the MSRP messages received from a realm includes only the MSRP URI scheme, session-id, and transport. If the comparison is unsuccessful and the sender requires a response, the SBC sends an MSRP 481 error response to the sender. If you enable **msrp-sessmatch**, you must disable **msrp-cema-support**.

Figure 22-4 Signaling Flow with Session Matching Enabled

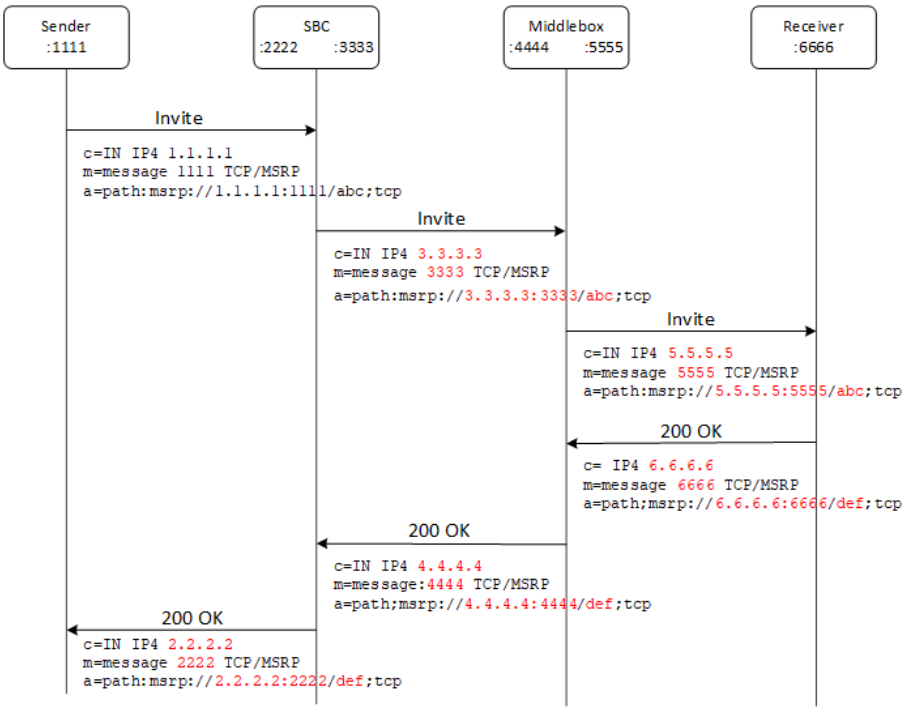
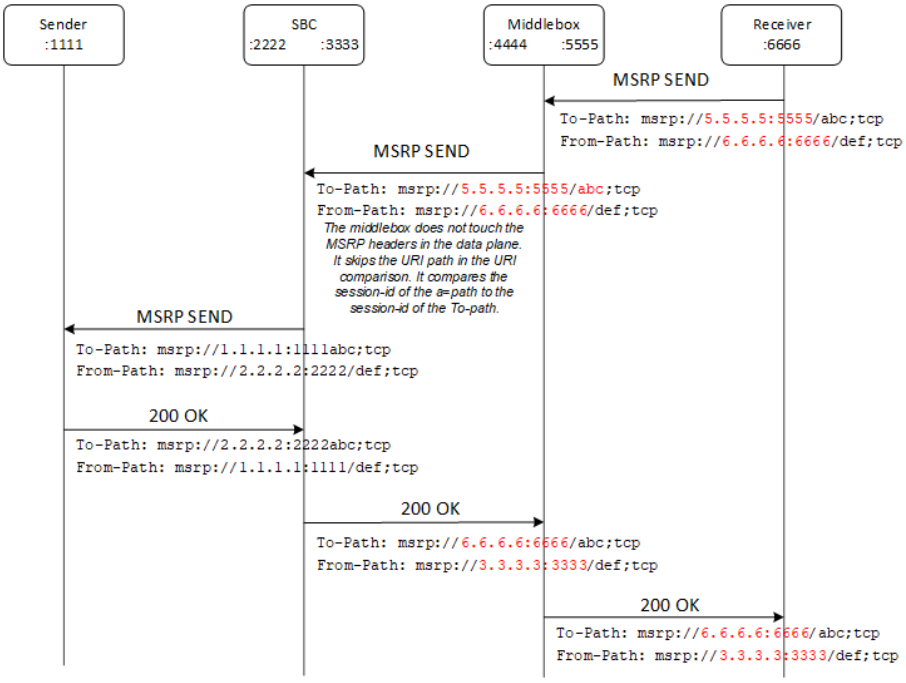


Figure 22-5 MSRP Flow with Session Matching Enabled



MSRP Media Types Filtering

You can configure the Oracle Communications Session Border Controller (SBC) to use an allowlist to filter media types and sub-types that you want the system to allow. During SIP signaling, the SBC removes media types not listed on the allowlist from the SDP offer. The SBC also rejects Message Session Relay Protocol (MSRP) SEND requests that announce media types in the MSRP content-type header other than those negotiated during the SDP offer-answer exchange. The SBC supports this functionality only on Virtual Machines.

Use the **msrp-types-allowlist** parameter in the tcp-media-profile configuration to set a list of media types and sub-types that you want the system to accept in MSRP messages. Each entry represents one media type and sub-type. When the parameter contains a valid value, the system checks that incoming MSRP SEND requests contain only the media types specified in the SDP a=accept-types attribute. Leave the **msrp-types-allowlist** parameter empty to tell the system not to perform any media types filtering. Use the asterisk character (*) to allow all MSRP media types. Valid values: empty | MsrpMediaTypeList | *. Default: empty.

The SBC does not provide filtering for the attributes that may accompany a media type, for example, the attribute 'charset' in,`text/html;charset=ISO-8859-1`. When an SDP offer contains a media type on the allowlist that includes attributes, the SBC does not touch the attributes.

MSRP Message Size Limiting

To set a limit on the Message Session Relay Protocol (MSRP) size of the message that the Oracle Communications Session Border Controller (SBC) can receive from a given realm, the tcp-media-profile configuration includes the **msrp-message-size**, **msrp-message-size-file**, and **msrp-message-size-enforce** options. The SBC supports this functionality only on Virtual Machines.

The maximum size of the MSRP messages that an MSRP Back to Back User Agent (B2BUA) can receive is advertised:

- in the signaling plane - in the SDP a=max-size attribute
- in the data plane - in the MSRP Byte-Range header.

The **msrp-message-size** parameter sets the size limit for interactive messages. The default is 0, which means that the SBC does not intervene in limiting the size for the interactive message. The valid range is 0-4194304 bytes. When the msrp-message-size-file value is greater than zero, the SBC:

- fills in the SDP offer-answer and a=max-size attribute, if missing.
- adjusts its value to the configured limits, if the value of the SDP a=max-size attribute exceeds the limit.

The **msrp-message-size-file** parameter sets the size limits for file transfer over MSRP that includes an SDP a=file-selector attribute. A file transfer over MSRP is identified by the presence of the SDP a=file-selector attribute in the MSRP media description. The default is 0, which means the SBC does not intervene in limiting the size for the file transfer. The valid range is 0-4294967295. Note that a file is transferred in the MSRP message in message/cpim format. You can set the msrp-message-size-file to a larger value than the maximum file to be transferred to provide room for the `message/cpim` header. (512 to 1024 bytes fits most scenarios.) When the msrp-message-size-file value is greater than zero, the SBC:

- fills in the SDP offer-answer a=max-size attribute, if missing.

- adjusts its value to the configured limit, if the value of the SDP a=max-size attribute exceeds the limit.

The **msrp-message-size-enforce** parameter, when enabled, performs byte counting on the MSRP messages to enforce compliance with the negotiated maximum MSRP message size. The negotiated maximum MSRP message size is the minimum between the SDP a=max-size attribute value (if one is provided by the B2BUA) and the configured value for **msrp-message-size** or **msrp-message-size-file**. If the SBC detects that the actual size of the MSRP chunk does not match the negotiated maximum size, it immediately stops forwarding the chunk. When the MSRP session is a file transfer, the SIP session terminates by sending a BYE on both the originating and the terminating legs.

> **Note:**
>
> The maximum negotiated size applies to MSRP chunks rather than MSRP messages.

MSRP and High Availability

Following a High Availability (HA) switchover on platforms that support Message Session Relay Protocol (MSRP), the new active Oracle Communications Session Border Controller (SBC) responds with a TCP RST to the first MSRP message received on an MSRP session that a UA established with the former active SBC. This response provides for a timely detection of the HA switchover and enables the UA to re-initiate the MSRP session by sending a SIP re-INVITE.

Upon a switchover, the first MSRP packet arriving at the newly active SBC triggers a TCP RST to be sent back immediately because the newly active does not have the TCP connection to receive the packet. This timely response allows the UA that sends the packet to quickly detect the connection interruption and send a reINVITE to set up a replacement session.

MSRP Configuration

MSRP configuration consists of the following steps.

1. Configure the msrp-config configuration object that governs MSRP global behavior.
2. Configure one or more tcp-media-profile configuration objects that define MSRP operations within a realm.
3. Assign a tcp-media-profile to a target realm.
4. If MSRP sessions are secured with TLS, create and assign tls-profile configuration objects to the tcp-media-profile of the target realm.
5. Create and assign steering-pools configuration objects to target realms.

msrp-config Configuration

Use the following procedure to perform MSRP global configuration.

1. From superuser mode, use the following command sequence to access msrp-config configuration mode. While in msrp-config mode, you configure global MSRP behavior.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
```

```
ORACLE(media-manager)# msrp-config
ORACLE(msrp-config)# ?
state                       state
uri-translation             perform translation of MSRP URI
session-inactivity-timer    timer value (seconds) for session inactivity
 monitoring period
conn-setup-delay-timer      timer value (seconds) to wait for UA to enter
session negotiation
msrp-kpi                    To enable MSRP kpi statistics
double-port-allocation      To allocate 2 ports for MSRP calls
select                      select msrp config to edit
no                          delete msrp config
show                        show msrp config
done                        write msrp config information
exit                        return to previous menu
ORACLE(msrp-config)#
```

2. Use the **state** parameter to enable MSRP operations.

   Retain the default value, **enabled**, to enable MSRP operations.

   If necessary, you can use **disabled** to temporarily suspend all MSRP operations.

   ```
   ORACLE(msrp-config)# state enabled
   ORACLE(msrp-config)#
   ```

3. Use the **uri-translation** parameter to enable or disable NAT of URIs found in the From-Path and To-Path headers of MSRP requests and responses, and in a=path attributes found in SDP offers.

   NAT is enabled by default.

   Retain the default value (**enabled**) to enable NAT; use **disabled** to disable NAT.

   ```
   ORACLE(msrp-config)# uri-translation enabled
   ORACLE(msrp-config)#
   ```

4. Use the **session-inactivity-timer** parameter in connection with the **msrp-delayed-bye-timer** parameter to implement the delayed transmission of SIP BYE requests, thus establishing a configurable transition interval allowing for the completion of active MSRP sessions.

   The **session-inactivity-timer** parameter specifies the maximum inactivity interval (defined as the absence of transmitted data) tolerated before the MSRP connection is terminated.

   Retain the default value (**5**), or specify another inactivity interval within the range 5 to 10 seconds.

   ```
   ORACLE(msrp-config)# session-inactivity-timer 7
   ORACLE(msrp-config)#
   ```

5. Use the **conn-setup-delay-timer** parameter to specify the maximum time before the system sets up a session with the UA even though it has not received any input or response from that UA.

Retain the default value (**0**), or specify another inactivity interval within the range 0 to 1500 seconds.

```
ORACLE(msrp-config)# conn-setup-delay-timer 7
ORACLE(msrp-config)#
```

6. Use the **msrp-kpi** parameter to enable MSRP KPI statistics.

   Retain the default value, **disabled**, for standard MSRP statistics.

   If necessary, you can use **enabled** to enable MSRP KPI statistics.

```
ORACLE(msrp-config)# double-port-allocation enabled
ORACLE(msrp-config)#
```

7. Use the **double-port-allocation** parameter to allow the system to assign two steering-pool ports for MSRP sessions.

   Retain the default value, **disabled**, for normal steering pool port assignment.

   If necessary, you can use **enabled** to allow the system to assign two steering-pool ports for MSRP sessions.

```
ORACLE(msrp-config)# double-port-allocation enabled
ORACLE(msrp-config)#
```

8. Use **done**, **exit**, and **verify-config** to complete MSRP global configuration.
9. If you wish to implement the delayed transmission of SIP BYE requests, use the following command sequence to access sip-config configuration model

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

10. Use the **msrp-delayed-bye-timer** parameter to enable the delayed transmission of SIP BYE requests, thus establishing a configurable transition interval allowing for the completion of active MSRP sessions.

    The **msrp-delayed-bye-timer** parameter specifies the maximum delay period allowed before transmitting the delayed BYE request.

    Retain the default value (**1**5), or specify another delay period within the range 1 to 60 seconds.

    Delayed transmission of BYE requests is enabled by default. Use the special value of 0 to disable delay, and transmit BYE requests immediately upon receipt.

```
ORACLE(sip-config)# msrp-delayed-bye-timer 20
ORACLE(sip-config)#
```

Configure tcp-media-profile

The tcp-media-profile defines Message Session Relay Protocol (MSRP) operations within a realm. You specify settings that are common to every tcp media profile, as well as optional settings that you use to customize a particular tcp media profile.

- If you want to set an allow list for allowed MSRP types, create the list before you perform this configuration.

Use the following procedure to build a TCP media profile that defines MSRP operations within a realm.

1. Access the **tcp-media-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(session-router)# tcp-media-profile
ORACLE(tcp-media-profile)#
```

2. Use the **name** parameter to provide a unique identifier for this TCP Media Profile instance.

```
ORACLE(tcp-media-profile)# name tlsMSRP
ORACLE(tcp-media-profile)#
```

3. Use the **media-type** parameter in conjunction with the **transport-protocol** parameter to identify the media-types and transport protocols (found in the SDP media description, m=, field as described in RFC 4566, SDP: Session Description Protocol) subject to this TCP Media Profile.

   media-type identifies the media subject to this TCP Media Profile. Retain the default value, **message**, for MSRP operations.

   transport-protocol identifies the transport layer protocols subject to this TCP Media Profile. Use either **TCP/MSRP** to specify unsecured TCP traffic or **TCP/TLS/MSRP** to specify secured, encrypted TLS traffic.

```
ORACLE(tcp-media-profile)# profile-list
ORACLE(profile-entry)#media-type message
ORACLE(profile-entry)#media-type


ORACLE(profile-entry)# transport-protocol TCP/TLS/MSRP
ORACLE(profile-entry)#
```

4. When the **transport-protocol** is **TCP/TLS/MSRP**, use the **tls-profile** parameter to identify the TLS profile that specifies the cryptographic resources available to support TLS operations.

   This parameter can be safely ignored if **transport-protocol** is **TCP/MSRP**.

```
ORACLE(profile-entry)# transport-protocol tcp/msrp
ORACLE(profile-entry)#
```

5. When the **transport-protocol** is **TCP/TLS/MSRP**, use the **require-fingerprint** parameter to enable or disable endpoint authentication using the certificate fingerprint methodology defined in RFC 4572, Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP).

By default, mutual authentication is **disabled**.

This parameter can be safely ignored if **transport-protocol** is **TCP/MSRP**.

```
ORACLE(profile-entry)# require-fingerprint enabled
ORACLE(profile-entry)#
```

6. Use the **listen-port** parameter to identify the TCP port monitored by the B2BUA for incoming MSRP connections. The 0 default value indicates that the B2BUA will choose the listening port from the steering pool of the realm (which the tcp-media-profile belongs to). Valid values: 0-65535. Default: 0.

```
ORACLE(profile-entry)# listen-port 43000
ORACLE(profile-entry)#
```

7. Use the **preferred-setup-role** parameter to specify the value the B2BUA uses for the a=setup attribute when negotiating the setup up role, regardless of the role (offerer or answerer) assumed by the B2BUA in the SDP offer- answer exchange.

```
ORACLE(profile-entry)# preferred-setup-role passive
ORACLE(profile-entry)#
```

The value of **preferred-setup-role** is used for the value of the a=setup attribute when the SBC makes an offer SDP and when the SBC replies to an offer SDP that has a=setup:actpass. It is not used when the SBC is forced into a role by the offerer, that is, if the offerer sends a=setup:active, the SBC must answer with a=setup:passive (and vice versa). Valid values: passive | active. Default: passive.

- Passive—Recommended. Indicates that the B2BUA accepts an incoming connection. Indicates that the system can either initiate an outgoing connection to a remote UA or accept an incoming connection from a remote UA. The role performed by the system, however, is ultimately determined by the value of the “a=setup” attribute in the answer SDP received from the remote UA.
- Active—Indicates that the B2BUA creates an outgoing connection.

8. Use the **msrp-cema-support** parameter to specify whether or not the SBC negotiates support for the CEMA extension (RFC6714) for TCP or TLS connections to and from the realm associated with the current TCP media profile. Enable the CEMA extension to enable the SBC to exchange MSRP traffic through middleboxes that anchor the media, but do not touch the SDP a:path attribute. Valid values: enabled | disabled. Default: disabled.

```
ORACLE(profile-entry)# msrp-cema-support enabled
ORACLE(profile-entry)#
```

9. Use the **msrp-sessmatch** parameter to specify whether or not the SBC validates the MSRP To-Path header based only on the session-id field and MSRP transport type of the MSRP URI (and not also on the IP address and port number in the authority part of the MSRP URI). Sessmatch enables the SBC to exchange MSRP traffic through Middleboxes that anchor the media and also adjust the SDP a=path attribute. Valid values: enabled | disabled. Default: disabled.

```
ORACLE(profile-entry)# msrp-sessmatch enabled
ORACLE(profile-entry)#
```

10. Use the **msrp-message-size-enforce** parameter to specify one element in an allowlist of allowed MSRP media types. Media types not included on the allowlist will be removed from

the SDP a=accept-types attribute of the SDP offers. A "*" indicates that all MSRP media types are allowed. When left empty, it indicates that no media types filtering is performed. Valid value: MsrpMediaTypeList.

```
ORACLE(profile-entry)# msrp-message-size-enforce enabled
ORACLE(profile-entry)#
```

11. Use the **msrp-message-size** parameter to specify the maximum size (in bytes) that MSRP is allowed to negotiate for the messages. It represents the maximum limit for the SDP a=max-size attribute, for the "size" token of the SDP a=file-selector attribute and MSRP Byte-range header. A value of 0 indicates that no maximum limit is enforced. Valid values: 0-4,000. Default: 0.

```
ORACLE(profile-entry)# msrp-message-size 2000
ORACLE(profile-entry)#
```

12. Use the **msrp-message-size-file** parameter to specify whether MSRP messages exceeding the negotiated size are rejected, respectively whether MRSP file transfers will be aborted when the negotiated size is exceeded. A value of 0 indicates that no maximum limit is enforced. Valid values: 0-4G. Default: 0.

```
ORACLE(profile-entry)# msrp-message-size-file 4
ORACLE(profile-entry)#
```

13. Use the **msrp-types-allowlist** parameter to specify a list of registered MSRP media types (RFC4975) supported for the ingress realm.

```
ORACLE(profile-entry)# msrp-types-allowlist <listname>
ORACLE(profile-entry)#
```

14. Use **done**, **exit**, and **verify-config** to complete tcp-media-profile configuration.

- Repeat the procedure to configure each additional tcp-media-profile that you need.
- Apply the profile to a realm.

Assign a tcp-media-profile to a Realm

Use the following procedure to assign a single, specific tcp-media-profile to a target realm.

1. From superuser mode, use the following command sequence to access realm-config configuration mode. While in realm-config mode, you assign a tcp-media-profile to a realm.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Use the **select** command to identify the target realm.
3. Use the **tcp-media-profile** parameter to assign a specific, named tcp-media-profile to the target realm.

```
ORACLE(realm-config)# tcp-media-profile tlsMutualAuth
ORACLE(realm-config)#
```

4. Use **done**, **exit**, and **verify-config** to complete tcp-media-profile assignment.

tls-profile Configuration

Use the following procedure to create a tls-profile configuration object, which specifies cryptographic resources available in support of TLS operations.

> **Note:**
>
> The option *allow-self-signed-cert* is only available for MSRP connections.

1. Access the **tls-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tls-profile
ORACLE(tls-profile)#
```

2. Use the **name** parameter to provide a unique identifier for this TLS Profile instance.

```
ORACLE(tls-profile)# name tlsMutualAuth
ORACLE(tls-profile)#
```

3. If the require-fingerprint attribute of the tcp-media-profile is set to **enabled**, use the **mutual-authenticate** parameter to enable mutual authentication.

```
ORACLE(tls-profile)# mutual-authenticate enabled
ORACLE(tls-profile)#
```

4. Retain default values for other parameters.
5. Type **done** to save your configuration.
6. Repeat Steps 1 through 5 to configure additional tls-profiles as required.

MSRP Statistics

MSRP byte and packet counters are available at the end of each MSRP call.

You can access stop records through the following interfaces:

- ACLI output. See the *ACLI Configuration Guide* and the *Maintenance and Troubleshooting Guide*. You can configure the system to display additional ACLI statistics.
- RADIUS VSAs and Local CDR. See "Oracle RADIUS VSAs" and "AVP Definitions" in the *Accounting Guide* for details.
- Diameter Rf ACR messages. See "Acme-Packet-Specific-Extension-Rf AVP" in the *Accounting Guide* for details.
- MSRP Objects for SNMP. See "SIP MIB" (ap-sip.mib) in the *MIB Reference Guide* for details. You can configure the system to display additional SNMP statistics.

Transmitted and received counters are available in Acme-Extended-Attributes. See "Acme-Extended-Attributes Explanation" in the *Accounting Guide* for more information.

MSRP Management Monitoring

When you want to see MSRP flow, cumulative, and state counts statistics for the Oracle Communications Session Border Controller (SBC), use the **show mbcd statistics**, **show msrp statistics**, and **show mbcd msrp** commands.

The **show mbcd statistics** command displays MSRP flow counts for the current statistics window. For example:

```
ORACLE# show mbcd statistics
16:38:49-145
MBCD Status                -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Sessions        1       1       0          1       1       1
...
Flow-MSRP              2       2       0          2       2       2
...
Flow Rate = 0.0
Load Rate = 0.0
ORACLE#
```

The **show msrp statistics** command can display the cumulative count for dozens of statistics. The system always displays the default set of statistics, even when the value is zero, plus any other statistics that do have a value. For example:

```
ORACLE# show msrp statistics
                       FD Table Size: 0
                Session-ID Table Size: 0
                Total Active Sessions: 0
               Maximum Active Sessions: 0
            Total Established Sessions: 0
            Total Provisioned Sessions: 0
               Total Finished Sessions: 0
            Total Accepted Connections: 0
           Total Connected Connections: 0
            Total Released Connections: 0
               Total Requests Received: 0
                   Total Requests Sent: 0
              Total Responses Received: 0
                  Total Responses Sent: 0
              Total Global Buffer Data: 0
            Total MSRP Nat Flows Added: 0
          Total MSRP Nat Flows Deleted: 0
            Total Active CEMA Sessions: 0
Total Established Sessmatch Sessions: 0
Total Provisioned Sessmatch Sessions: 0
     Total Active Sessmatch Sessions: 0
```

The **show mbcd msrp** command displays MSRP session state counts. For example:

```
ORACLE# show mbcd msrp
14:35:15-194
MSRP Statistics                              ---- Lifetime ----
                                  Recent      Total  PerMax
```

```
Provision Sessions Unreleased          0          0       0
Listening Sessions Unreleased          0          0       0
Established Sessions Unreleased        0          0       0
Closed Sessions Unreleased             0          0       0
Finished Sessions Unreleased           0          0       0
PPM Messages Received                  0          0       0
PPM Messages Processed                 0          0       0
PPM Messages Sent                      0          0       0
PPM Messages Send Fail                 0          0       0

MSRP Flow States                        -- Period -- -------- Lifetime
--------
                              Active    High   Total      Total  PerMax
High
MSRP-FlowListen                    0       0       0          0       0
0
MSRP-FlowProvision                 0       0       0          0       0
0
MSRP-FlowProvisioned               0       0       0          0       0
0
MSRP-FlowEstablished               0       0       0          0       0
0
MSRP-FlowShared                    0       0       0          0       0
0
MSRP-FlowClose                     0       0       0          0       0
0
MSRP-FlowFinished                  0       0       0          0       0
0
MSRP-FlowReleased                  0       0       0          0       0
0
MSRP-FlowWaitDrop                  0       0       0          0       0
0
```

You can configure the system to extend the **show mbcd msrp** output. After configuration, the system provides additional arguments. The additional arguments provide a broad set of system-wide or realm-based statistics, and include SEND and REPORT statistics. See "Extended MSRP Statistics" in the *ACLI Configuration Guide* and "MSRP Protocol Performance" in the *Maintenance and Troubleshooting Guide* for more information, including configuration.

Extended MSRP Statistics

You can configure the Oracle Communications Session Border Controller (SBC) to display additional protocol statistics on MSRP methods using SNMP and CLI.

The SBC displays MSRP traffic statistics on a per-window, periodic basis (100 seconds) and on a lifetime basis. The SBC correlates statistics using MSRP method type, including SEND and REPORT, and on a per-realm and system-wide basis. You can display the statistics using the ALCI by specifying method and basis. You can also collect the statistics using SNMP walks. Use the **reset all** and **reset mbcd** commands to reset MSRP KPI statistics.

The SBC supports Extended MSRP statistics on the following physical platforms, only:

- Acme Packet 4600
- Acme Packet 4900

- Acme Packet 6100
- Acme Packet 6300
- Acme Packet 6350

The SBC uses SIP and SDP to signal MSRP media traffic. MSRP traffic includes:

- MSRP SEND requests, which present the following methods:
  - CHAT
  - IS-TYPING
  - RECEIPT
- REPORT traffic, which indicates delivery success and failure
- Transaction response status traffic, which indicates success and failure in response to a request

You can use extended MSRP statistics to segregate MSRP method traffic, track cause values and byte counts on a receive and transmit basis. This allows you to analyze sessions within the context of specific transaction types, across relays, and across MSRP chunks.

> **Note:**
>
> RADIUS, CDR, and DIAMETER do not support Extended MSRP statistics.

The MSRP complexities that extended KPIs can help you analyze include:

- The session-oriented nature of MSRP requires the use of Message-ID and byte ranges, if present, to correlate traffic.
- Cause codes can change over the course of a session, based on circumstances across the relay path or the endpoints.
- SEND requests deliver a complete message or a chunk, which is a portion of a complete message, which can be supported by monitoring byte ranges to correlate components of associated requests.
- REPORT requests report on the status of a previously sent message, or a range of bytes inside a message.

You enable extended MSRP statistics by enabling the **msrp-kpi** parameter in **msrp-config**.

```
ORACLE(msrp-config)# msrp-kpi
<enabled/disabled> To enable the MSRP KPI statistics
Default: disabled
```

When you enable these extended statistics, the SBC provides additional MSRP statistics commands.

```
show mbcd msrp
show mbcd msrp realm <identifier>
show mbcd msrp SEND
show mbcd msrp REPORT
show mbcd msrp REPORT failure
show mbcd msrp SEND responses
show mbcd msrp realm <identifier> SEND
```

```
show mbcd msrp realm <identifier> REPORT
show mbcd msrp realm <identifier> REPORT failure
show mbcd msrp realm <identifier> SEND responses
```

See the *MSRP Protocol Performance* section in the *Oracle® Communications Session Border Controller Maintenance and Troubleshooting Guide* for detail on these commands.

Statistic Calculation Formulas

The SBC implements MSRP KPI statistics for total, average, and rate using the formulas shown below. The examples below use SEND transactions for the average and total statistics, and REPORT failure transactions for rate statistics:

- System wide:
  - Average calculation - MSRP-AvgSENDTransTx = Sum of all Tx SEND transactions of peer and core/number of MSRP sessions.
  - Total calculation - MSRP-SENDMsgBytesTx = Sum of Message bytes of all Tx SEND transactions of peer and core
  - Rate calculation - MSRP-REPORTFailureRateTx = (sum of all Tx REPORT failure transactions of peer and core / sum of all Tx REPORT success and failure transactions of peer and core) * 100
- Per Realm:
  - Average calculation - MSRP-AvgSENDTransTx = Sum of all Tx SEND transactions per realm/number of MSRP sessions
  - Total calculation - MSRP-SENDMsgBytesTx = Sum of Message bytes of all Tx SEND transactions per realm
  - Rate calculation - MSRP-REPORTFailureRateTx = (sum of all Tx REPORT failure transactions per realm /sum of all Tx REPORT success and failure transactions per realm) * 100

SNMP

Extended MSRP KPI statistics are also available by way of SNMP. These are the same statistics available from the ACLI, and also require configuration. Extended SNMP statistics are realm and system-wide.

Realm—The MSRP KPI realm labels are contained in OID APAPPS-MIB. The apAppsMSRPKPIRealmEntry entry consists of apMSRPKPIRealmName, which is configured in ACLI and apMSRPKPIRealmIndex, which is the object ID of the ACLI config element. Syntax for complete OIDs follow this sequence:

1. The root label for the MSRP KPI realm Stats SNMP table is APAPPS-MIB:: ApAppsMSRPKPIRealmStatsEntry
   The OID of each realm is the same ID as the configuration object instance. This means the table starts from the config-object ID, not from 0.
2. Next is the Instance OID (msrp-realm).
3. Next is the type of stats: recent, high, total, ltotal, lpermax, lhigh.
   The system distinguishes between apMsrpKpiStatsCounterType using the following values:
   - Period window values:
     - recent (1)

  - high (2)
  - total (3)
- Lifetime window values:
  - ltotal (4)
  - lpermax (5)
  - lhigh (6)

4. The last label is the type of statistic, including msrp-AvgSENDTransTx, msrp-AvgChatSENDTransTx, and so forth.

SystemWide—The ApAppsMSRPKPISystemStatsEntry entry consists of apMsrpKpiStatsCounterType and apMSRPKPIStatsType. Syntax for complete OIDs follow this sequence:

1. The root label for the MSRP KPI system Stats SNMP table is APAPPS-MIB:: ApAppsMSRPKPISystemStatsEntry
2. The type of stats: recent, high, total, ltotal, lpermax, lhigh
   The system distinguishes between apMsrpKpiStatsCounterType using the following values:
   - Period window values:
     - recent (1)
     - high (2)
     - total (3)
   - Lifetime window values:
     - ltotal (4)
     - lpermax (5)
     - lhigh (6)
3. The type of statistic, including msrp-AvgSENDTransTx, msrp-AvgChatSENDTransTx, and so forth.

See the *Oracle® Communications Session Border Controller MIB Guide* for detail on these tables.

MSRP Session Limitations Based on Entitlements

This section provides some guidelines on how the SBC determines whether your configured MSRP entitlement does or does not allow new sessions when you are reaching your configured session limit.

Within the context of MSRP, session definitions include:

- Active MSRP sessions—The MSRP TCP/TLS connections that are in the connected state. Active sessions include those in the Listen, Provisioning, Provisioned, Established and Shared states.
- InActive MSRP sessions—The MSRP TCP/TLS connections which are in terminated state. Inactive sessions include those in states the Close, Finished, and WaitDrop states.

When evaluating your system's MSRP session capacity, the SBC uses the following guidelines to calculate Active MSRP sessions:

- Your configured entitlement minus the currently Active (connected) sessions, OR

- The total number of sessions shall be derived using the configured entitlement as a base value by keeping some additional buffer minus the current number of sessions (ACTIVE+INACTIVE).

> **Note:**
>
> MSRP Flow States shows the counts in terms of flows. One B2B MSRP session is equal to '2' MSRP flows.

The examples below explain how the SBC is allowing the new calls/sessions by considering configured entitlement and MSRP states.

Example 1

Assume for this example, you have set your MSRP B2BUA Sessions entitlement to 1.

1. The SBC has received the first call, which is in Provisioned state.

```
ORACLE# show mbcd msrp

MSRP Flow States                -------- Period  -------------- Lifetime
--------

                                Active    High   Total      Total
PerMax    High
MSRP-FlowInitial                   0       0       0          0
0       0
MSRP-FlowListen                    0       0       0          0
0       0
MSRP-FlowProvision                 0       0       0          0
0       0
MSRP-FlowProvisioned               2       2       2          0
0       0
MSRP-FlowEstablished               0       0       0          0
0       0
MSRP-FlowShared                    0       0       0          0
0       0
MSRP-FlowClose                     0       0       0          0
0       0
MSRP-FlowFinished                  0       0       0          0
0       0
MSRP-FlowReleased                  0       0       0          0
0       0
MSRP-FlowWaitDrop                  0       0       0          0
0       0
```

2. When the SBC receives a second call, it rejects that call with a 503 message because it can support a maximum of one call/session based on the configured entitlement.

Example 2

Assume for this example, you have set your MSRP B2BUA Sessions entitlement to 3.

1. The SBC has received three calls, which are in Provisioned, Established, Shared states.

```
ORACLE# show mbcd msrp

MSRP Flow States                -------- Period  -------------- Lifetime
--------
                                Active    High   Total      Total
PerMax    High
MSRP-FlowInitial                   0       0       0          0
0       0
MSRP-FlowListen                    0       0       0          0
0       0
MSRP-FlowProvision                 0       0       0          0
0       0
MSRP-FlowProvisioned               2       2       2          0
0       0
MSRP-FlowEstablished               2       2       2          0
0       0
MSRP-FlowShared                    2       2       2          0
0       0
MSRP-FlowClose                     0       0       0          0
0       0
MSRP-FlowFinished                  0       0       0          0
0       0
MSRP-FlowReleased                  0       0       0          0
0       0
MSRP-FlowWaitDrop                  0       0       0          0
0       0
```

2. When the SBC receives a fourth call, it rejects that call with a 503 message because it can support a maximum three call/session based on configured entitlement.
3. When the SBC receives a BYE for, for example, the first call and has closed the associated MSRP connections, the **show mbcd msrp** statistics appear as follows.

```
ORACLE# show mbcd msrp

MSRP Flow States                -------- Period  -------------- Lifetime
--------
                                Active    High   Total      Total
PerMax    High
MSRP-FlowInitial                   0       0       0          0
0       0
MSRP-FlowListen                    0       0       0          0
0       0
MSRP-FlowProvision                 0       0       0          0
0       0
MSRP-FlowProvisioned               2       2       2          0
0       0
MSRP-FlowEstablished               0       2       2          0
0       0
MSRP-FlowShared                    2       2       2          0
0       0
MSRP-FlowClose                     0       0       0          0
0       0
MSRP-FlowFinished                  0       0       0          0
```

```
0       0
MSRP-FlowReleased                  2       2       2          0
0       0
MSRP-FlowWaitDrop                  0       0       0          0
0       0
```

4. When the SBC receives a new call, it allows that call because it now has two only Active calls and can support one new call.

```
ORACLE# show mbcd msrp

MSRP Flow States                --------- Period  -------------- Lifetime
--------
                                Active    High    Total      Total
PerMax    High
MSRP-FlowInitial                   0       0       0          0
0       0
MSRP-FlowListen                    0       0       0          0
0       0
MSRP-FlowProvision                 0       0       0          0
0       0
MSRP-FlowProvisioned               2       2       2          0
0       0
MSRP-FlowEstablished               2       2       2          0
0       0
MSRP-FlowShared                    2       2       2          0
0       0
MSRP-FlowClose                     0       0       0          0
0       0
MSRP-FlowFinished                  0       0       0          0
0       0
MSRP-FlowReleased                  2       2       2          0
0       0
MSRP-FlowWaitDrop                  0       0       0          0
0       0
```

At any point in time, whatever the configured MSRP entitlements value, the SBC supports that many Active MSRP sessions within the context of the best case scenario (which does not include inactive state sessions).

RCSe TLS/TCP Re-Use Connections

In an RCSe environment the sip-interface reuse-connections option is used to make the Oracle Communications Session Border Controller retain the TCP/TLS connection established by the endpoint during the registration for all subsequent messages to that endpoint, essentially providing for a persistent connection between the Oracle Communications Session Border Controller and the user equipment (UE).

Field experience uncovered an implementation deficiency associated with these persistent connections particularly within RCSe deployments. The basic scenario is as follows:

1. The UE registers in a TLS realm on SBC1. SBC1 stores the IP:Port from VIA (and Contact) as alias of the currently established connection.

2. The UE transits to another realm/sip-port (same or different Oracle Communications Session Border Controller) without previously unregistering or closing the TCP connection with the TLS sip-port on SBC1.
3. UE goes back to the TLS realm in SBC1 and establishes a new connection — same source IP as in Step 1, but a different port as in Step 1.

The problem arises at Step 3. If the Oracle Communications Session Border Controller has not detected that the TLS connection established in Step 1 has been effectively terminated, it will not update the alias connection to that established in Step 3, but instead continue to attempt to use the Step 1 connection.

This means that the next message from the core side to the UE will fail, since the Oracle Communications Session Border Controller will attempt to send the message of the dead TLS connection — that is using the IP address:port pair passed in Step 1.

All communications to this UE will fail until it sends the next message to the Oracle Communications Session Border Controller, when the alias connection will be update to the TLS connection in Step 3.

To resolve this issue, the Oracle Communications Session Border Controller needs to always update the alias table when it receives a new inbound connection on the configured sip-interface.

Option Configuration Guidelines

The following table lists the full range of options that pertain to TLS/TCP Connection reuse with an emphasis on use in RCSe environments.

Option	Connection Behavior
reuse-connections	Use/retain first inbound connection until explicitly closed
reuse-connections=latest	Use the last inbound connection, update the alias for each new connection
reuse-connections=no	Establish new connection at each UE access
NOT CONFIGURED	Equivalent to reuse-connections=no

RCSE TLS/TCP Re-Use Connections Configuration

To configure the reuse-connections option:

1. From Superuser mode, use the following command sequence to navigate to the sip-interface configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)# sip-interface
ORACLE(sip-interface)# option reuse-connections=latest
ORACLE(sip-interface)#
```

2. Use **done**, **exit**, and **verify-config** to complete configuration.

23
Local Media Playback

Commonly, ringback is the media playback of a certain tone informing callers that their calls are in progress. In typical deployments, remote endpoints or media servers handle ringback generation, leaving the Oracle Communications Session Border Controller (SBC) to proxy RTP. You can configure the SBC becomes the producer for environments where endpoints or media servers do not support ringback generation, or if different ringback is required

When you configure the SBC to generate ringback locally, you enable it to produce RTP media on a media flow called ringback tone (RBT) from a **session-agent**, **sip-interface** or **realm-config**, using typical precedence when determining which configuration to use if there are overlapping configurations. There are other criteria to determine which configuration to use described herein. These are generally characterized as "closest to the target". Because you can also use this capability for music-on-hold, announcements, interrupting media for notifications and so forth, this SBC capability is referred to as local medial playback.

The SBC can produce Local Media Playback using one of two methods:

- Transcoding based ringback—The SBC uses local transcoding resources to produce RTP.
- SPL based ringback—The SBC generates RTP using SPL resources.

You can only use one method on an SBC at a time.

There are two components of RBT configuration for both the SPL and transcoding-based methods:

- Triggers—Specifies when the SBC plays the media file:
  - Transcoding-based—Configured with the ACLI **ringback-trigger** parameter
  - SPL—Set as parameters to the **spl-option**
- Media—Specifies the local media file that the SBC plays:
  - Transcoding-based—Configured with the ACLI **ringback-file** parameter
  - SPL—Configured with ACLI **playback-config** elements and applied as a parameter to the **spl-option**

> **Note:**
>
> Transcoding based RBT requires transcoding resources. For example, be sure you have a transcoding core configured on your vSBCs to use this feature.

Additional Configuration Consideration

There are issues you may encounter with RBT generation with respect to compliance with RFC 6337, such as 1-way audio after the 200 OK. If the SDP presented in final 200 OK is not consistent with final 200 OK answer from the end station, this departure from RFC compliance may trigger a rejection of the SDP by the endstation. In such cases, the RBT may continue, but the call will fail.

To alleviate this compliance issue, set the **unique-sdp-id** option in the **media-manager**.

```
ACMEPACKET(media-manager)# options +unique-sdp-id
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

Local Media Playback Operation

You configure the Oracle Communications Session Border Controller (SBC) to generate media locally by specifying the triggers and media files you need. The SBC monitors signaling traffic for playback triggers. For the most part, operation of the local media playback feature is the same regardless of the RTP generation method. Differences are explained herein.

The SBC allows for playback configurations on a session agent, realm and sip interface. It plays back media to a caller using the configuration 'closest' to that endpoint. The term 'closest' refers to the hierarchy by which the SBC selects the playback configuration to use for a given call. The hierarchy the SBC uses is session-agent, followed by realm, followed by sip-interface. To complete the notion of 'closest', consider the element's proximity to the end station initiating the call, to which the SBC sends the RTP.

For example, if the initiating end station is a session agent that includes a playback configuration, then the SBC would use that configuration. But if the endpoint is not a session agent or is a session agent without a playback configuration, the SBC would check that endstation's realm, then the sip-interface configurations to identify which, if any, playback trigger and media it would use.

Key operational detail on the RTP stream generated by the SBC playback function includes:

- The SBC supports local media playback as RTP or SRTP streams over IPv4 and IPv6, using UDP transport and SIP.
- The SBC supports local media playback over VLANs.
- The generated RTP stream complies with all relevant RTP standards, including incrementing RTP timestamp and Sequence Number, and specifying a unique SSRC.
- If applicable, the playback RTP stream appropriately maintains the Payload Type, SSRC, RTP timestamp and Sequence Numbers of the original media stream.
- The SBC marks the first packet of the playback with the RTP marker bit.
- If the SBC receives a SIP request, such as an UPDATE or REINVITE, that includes a new SDP offer and the p-acme-playback header, it waits to play the RBT until it receives a corresponding successful answer. This resolves the issue wherein it is unclear which codec to use to play RBT (originator, terminator, or both) because the answer is incomplete and the request may still be rejected.
- For all triggers except the playback-on header trigger, the SBC does not create a playback stream if:
  - The initial INVITE or the SIP reply contains the proprietary SIP header "P-Acme-RBT: no".
  - The 18x response includes the "P-Early-Media:sendonly" or "P-Early-Media:sendrecv" parameter(s).

> **Note:**
>
> The SBC does play RBT if the 18x response has P-Early-Media set to "recvonly" or "inactive".

- If playback is operating on a hairpin stream, and the scenario fires multiple playback triggers, the SBC plays the stream based on the configuration 'closest' to the destination.
- Once playback is in progress, the SBC mutes the session in the playback direction so that only the playback media can be heard.

The SBC stops local media playback when:

- It receives the final SIP answer from the callee, or
- If you are using the 180-force or 180-no-sdp configuration, the callee has received a SIP UPDATE with SDP and has relayed it to the caller.

External signaling and other SBC configuration that impacts local media playback deployments include:

- The SBC disables local media playback when configured to release media.
- If both 2833 generation and playback are configured on a flow, the SBC gives precedence to the playback feature, disabling 2833 generation.
- If you are using the 180-no-sdp configuration, the SBC has not received SDP from the callee, and the processed response does not contain SDP, the SBC adds SDP:
  - The SBC chooses the codec based on the offer received, and after application of the ingress and egress codec policy.
  - If the SBC has not received an offer at that time from caller (delayed offer scenario), the SBC disables local media playback.
- When several early dialogs are received from the called party side, the SBC starts and stops local media playback based on the “active” early dialog. The active dialog is the last dialog for which the SBC received a provisional response.

Supported Local Media Triggers

The Oracle Communications Session Border Controller (SBC) can generate media locally based on end station signaling, local media playback configuration, and other SBC configuration.

Local media playback capabilities are dependent upon your choice of media generation method. Be aware of the few operational differences between these methods. Most new deployments use the transcoding-based method.

Deployments Configured for Transcoding Resources

The Oracle Communications Session Border Controller (SBC) supports the following playback triggers when configured for transcoding resources:

- Playback on 180 Ringing
- Playback on 180 Ringing when 180 does not include SDP
- Playback on 183 Session Progress
- Playback on REFER
- Playback upon both 183 and REFER

- Playback on header, where the header is P-Acme-Playback

You can also configure the SBC with HMR response code mapping on the SIP call egress side to map a 183 into a 180 response, thereby triggering playback on 180 Ringing. This requires that you map a 183 response into a 180 response, which then triggers playback.

Deployments Configured with SPL

The SBC supports the following playback triggers when configured with SPL:

- Playback on 183 Session Progress
- Playback on REFER
- Playback on header, where the header is P-Acme-Playback

The SBC does not support the following operations for local media playback when configured with SPL:

- SRTP
- Call recording
- SIPREC

Media Files

Media files of ringback tones are uploaded to /code/media to the SBC. This file differs based on your media generation method and must be raw media binary. For Transcoding based RBT, ensure that the files RAW PCM 16-bit MONO samples, sampled at 8-khz encapsulated with little-endian formatting and cannot exceed 4.8 MB.

When using the SPL method:

- A separate file is required for each different codec type, even if the media itself is the same.
- Your configuration must specify a playback rate in bytes per second, as this setting defines how many bytes of data per unit of ptime are needed.
- To preserve system memory resources, media files are limited to 2MB.

No media file can not exceed 5 minutes of playback data.

Media Setup and Playback

For each session requiring media playback, the SBC sets up media that supports standard RTP parameters. From the original media (if present), the SBC preserves the synchronization source (SSRC), timestamp, and sequence number.

Playback duration is continuous except when using the playback-on-header option. The playback duration for the playback on header scenario changes according to the settings in the P-Acme-Playback header. This header contains a duration=<ms-value|once|continuous> value, so you can customize the playback duration.

Playback timing options via playback on header include:

- Continuous—Media playback continues until the point at which the playback scenario defines its stop; media file loops if it fails to cover the entire playback duration.
- Once—The file plays until either the playback scenario defines its stop or until the media file runs out.

- Ms-value—Playback continues for a specific duration (in milliseconds) and will loop if necessary.

The P-Acme-Playback Header

You can configure the SBC to generate RTP media on a media flow when a request or response contains the P-Acme-Playback header after media setup is complete. You can configure this playback on a realm-config, sip-interface, or session-agent with either media generation method.

The P-Acme-Playback header can be included, along with the file format, by the endpoint in any request or response. When the SBC receives a header, it starts or stops the media, based on the header's parameters. By default, the SBC stops playback on any final response.

The syntax of the header, including all of its possible parameters, is:

```
P-Acme-Playback: <start|stop>[;media=<media file name>][;duration=<ms-value|
once|continuous>]
[;direction=<originator|terminator|both>][;stop-on-final-resp=<true|false>]
```

Header Element	Description
<start|stop>	Required Defines starting and stopping playback. • start: starts playback • stop: stops playback
[;media=<media-name>]	Optional Defines the name of the playback configuration to play. If unspecified, the playback configuration that was triggered by the header will play.
[;duration=<ms-value|once|continuous>]	Optional Defines the duration of playback. If unspecified, the value will be taken from the playback-config that was triggered. • ms-value: time value in milliseconds • once: plays playback media one time • continuous: loops the playback media
[;direction=<originator|terminator|both>]	Optional Defines the direction from which to play media. If unspecified, playback will begin in the realm, session agent, or SIP interface from which the header was received. • originator: plays in the west flow (original caller) • terminator: plays in the east flow (original callee) • both: plays in both directions

Header Element	Description
[;stop-on-final-resp=<true|false>]	Optional Defines whether or not to stop playing media upon the final response. If unspecified, this parameter is true. • true: stops playback automatically on a final response • false: stop only after a stop header is received or media terminated

> **Note:**
>
> Play back trigger playback-on-header does not work with other triggers like 180-force, 183.

Supported Playback Scenarios

This section lists and provides call flows for playback scenario triggers supported by the SBC. When more than one trigger applies to a call flow, the SBC acts on the trigger configuration closest to the playback endpoint.

Supported Playback Scenarios when using Transcoding-based media genearation

These scenarios are defined by the **ringback-trigger** parameter you configure for session agents, realms, and SIP interfaces:

- **none**—The SBC does not perform local media playback procedures for this configuration. Based on precedence, however, the SBC may issue playback based on other element configurations. Local media playback follows the precedence session-agent, realm, then sip-interface.
- **disabled**— The SBC does not perform media playback procedures on this flow, regardless of ensuing configurations.
- **180-no-sdp**—The SBC starts local media playback to the caller when a 180 is received without SDP, and a 18x with SDP has not yet been received. The SBC stops the playback:
  - On the final response, or
  - When the SBC receives an UPDATE with SDP from the callee.
- **180-force**—The SBC starts local media playback to the caller when a 180 is received regardless of whether it includes SDP. The SBC stops the playback:
  - On the final response, or
  - When the SBC receives an UPDATE with SDP from the callee.
- **183**—The SBC starts local media playback to caller when 183 is sent to call originator. The SBC stops the playback on the final response (either 2xx success or 4xx error). Configure this **183** value on the original INVITE ingress realm/sip-interface/session-agent.
- **refer**—The SBC starts local media playback to the referee when it receives a REFER. This trigger operates only if the SBC actually terminates and performs the refer operation. If the REFER is via proxy, playback is not a triggered. Playback stops when the refer operation is

complete with a final response (200-299 or 400-699). Configure this **refer** value on the ingress realm/sip-interface/session-agent of the transferred call.

- **183-and-refer**—The SBC starts local media playback when both 183 and refer triggers are activated.
- **playback-on-header**—The SBC starts or stops playback based on the presence of the P-Acme-Playback header and its definitions.

Playback triggers you configure when using the SPL method differ slightly from the transcoding method, but support the same types of event for triggers. You configure those triggers with SPL options.

Supported Playback Scenarios when using SPL

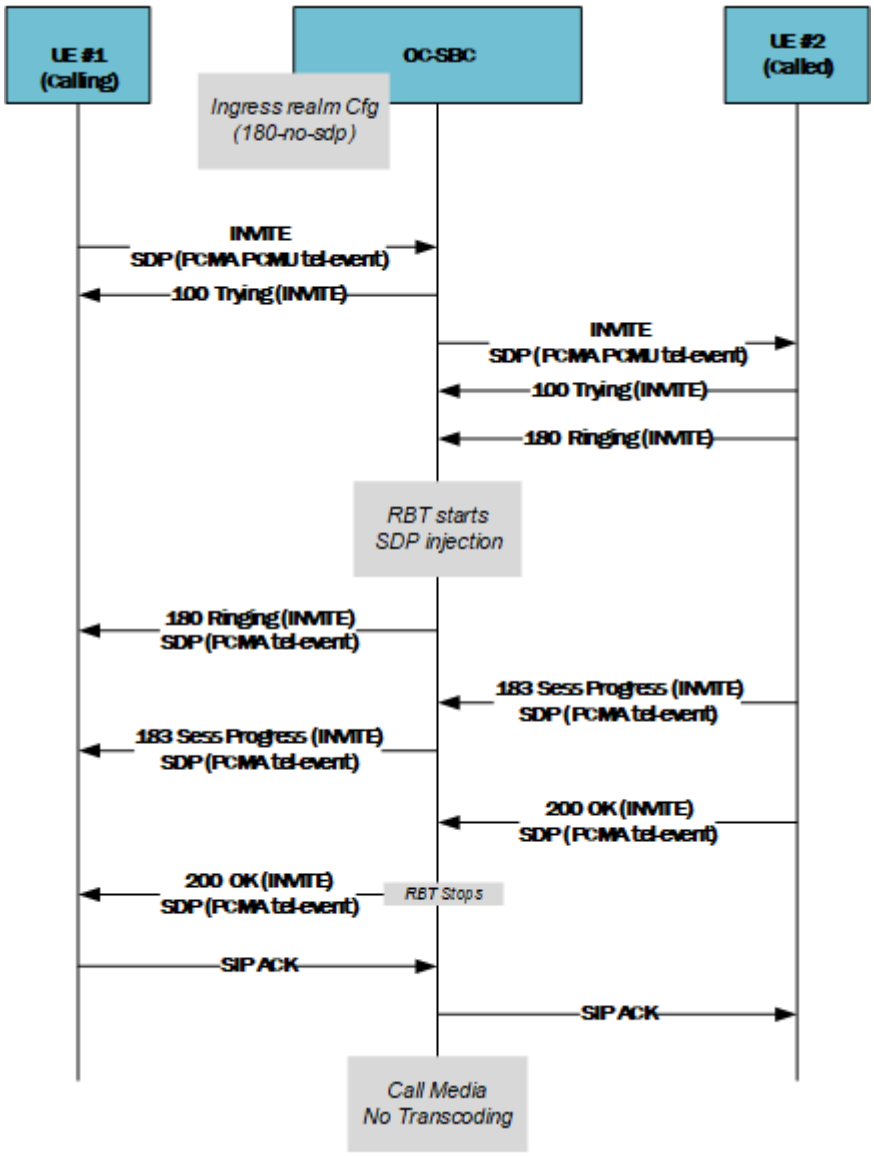
These scenarios are defined by the SPL options parameters you configure for realms, session agents, and SIP interfaces. For more information about configuring these options, see documentation on the Session Plug-in Language (SPL).

- **playback-on-183-to-originator**—Playback enabled upon the receipt of a 183 Session Progress destined for the originator and stops when a either a (200-299 or 400-699) final response is sent.
- **playback-on-183-from-terminator**—Playback enabled upon the receipt of a 183 Session Progress response is received from the terminator and stops when a (200-299 or 400-699) final response is received.
- **playback-on-refer**—Playback enabled for the caller being transferred when the SBC receives a REFER message that is locally terminated (i.e., processed on the SBC on REFER completion).
- **playback-on-header**—Starts or stops playback based on the presence of the P-Acme-Playback header and its definitions.

> **Note:**
>
> The SBC supports a maximum of 100 simultaneous playbacks when configured using the SPL method.

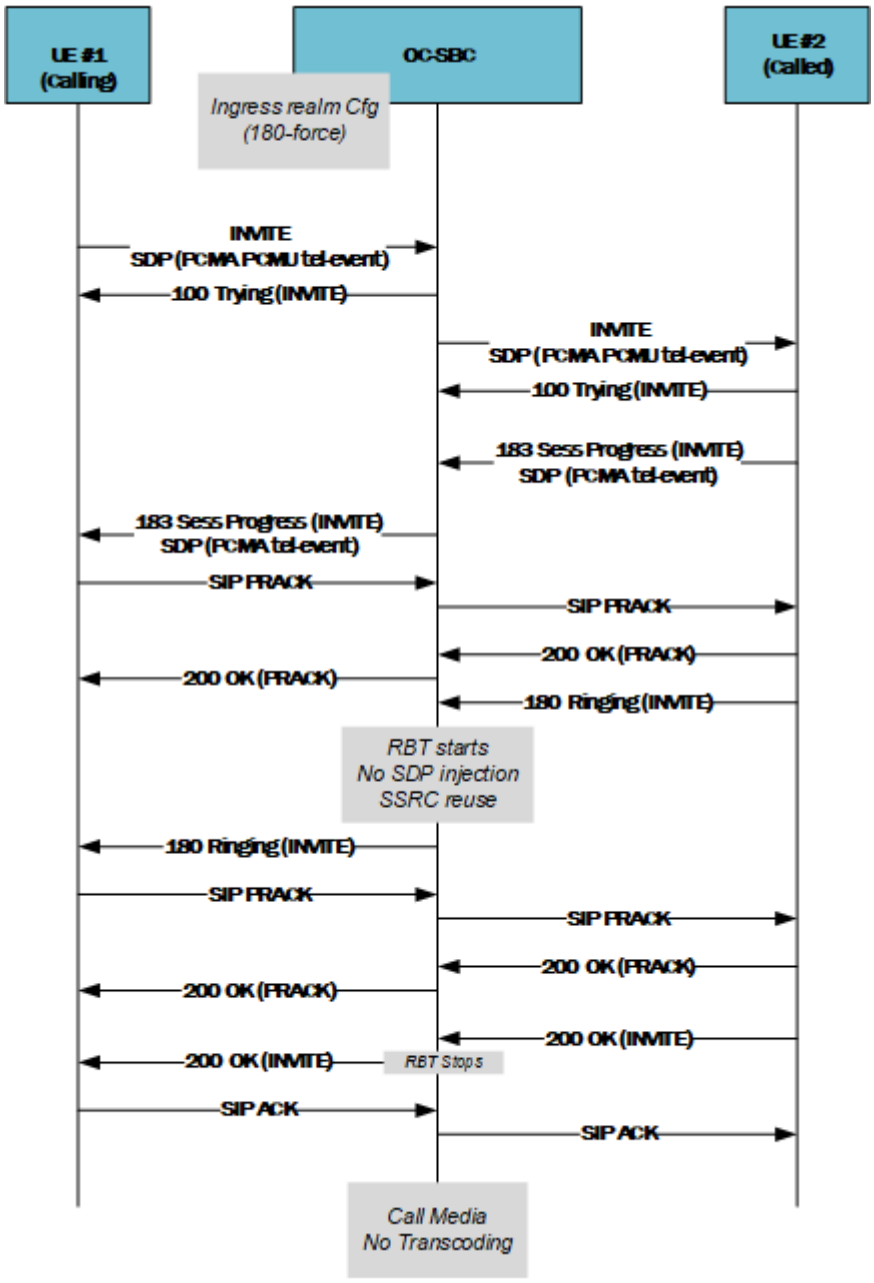
Playback on 180-no-sdp

The call flow below shows local media playback injecting RBT. The applicable configuration sets the ingress realm's **ringback-trigger** to use **180-no-sdp**. This scenario triggers only for 180 responses to initial INVITEs, not for re-INVITEs.



Playback on 180-force

The call flow below shows local media playback injecting RBT. The applicable configuration sets the ingress realm's **ringback-trigger** to use **180-force**. This scenario triggers only for 180 responses to initial INVITEs, not for re-INVITEs.

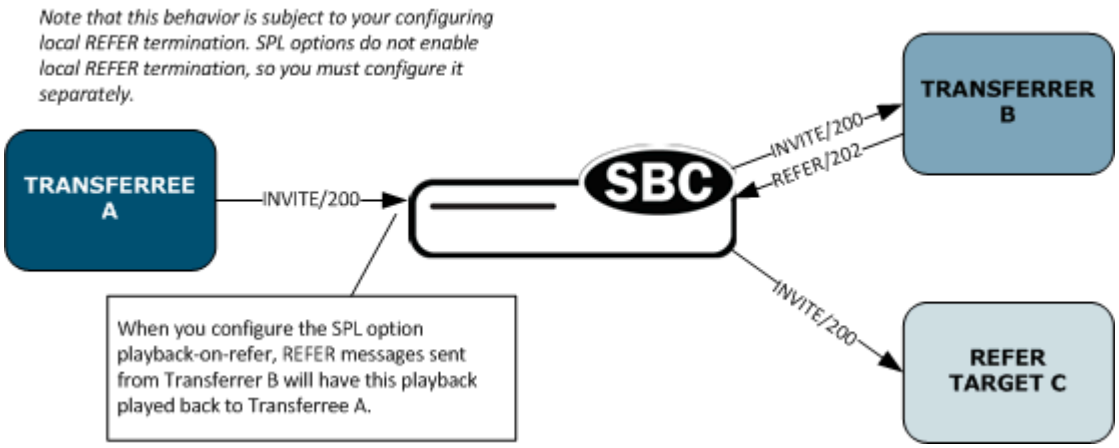


Playback on REFER

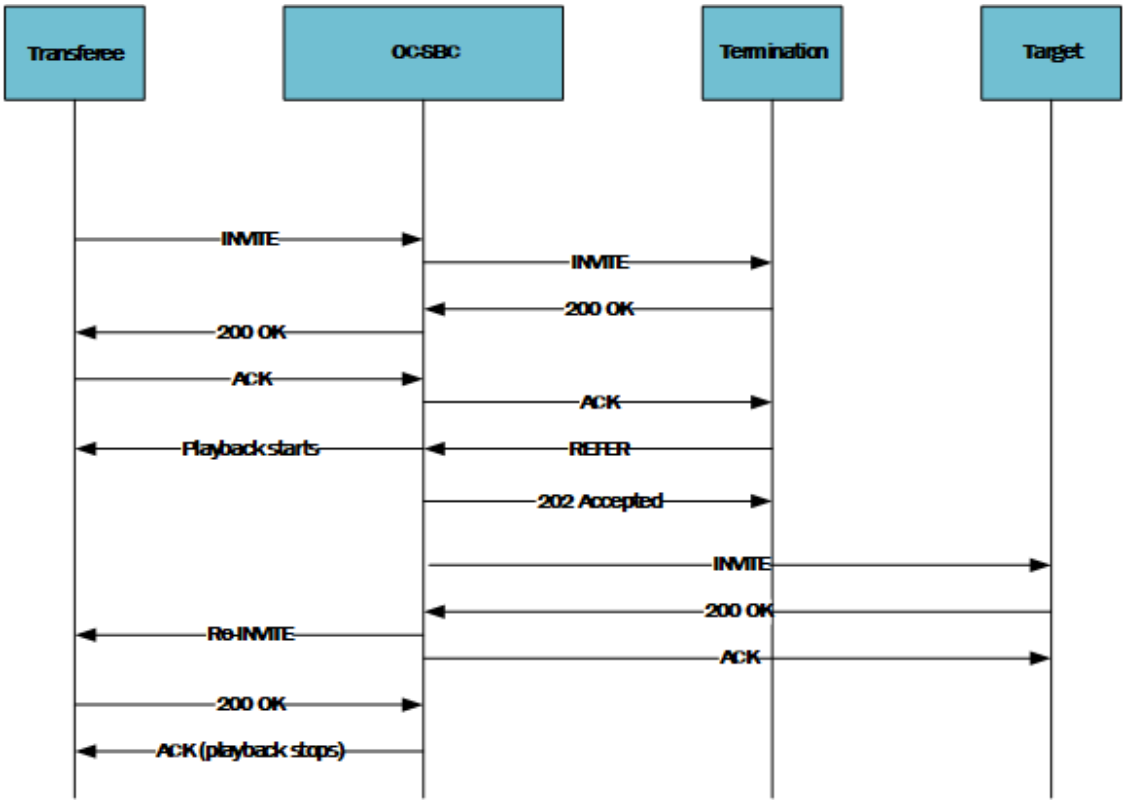
Setting the SPL options parameter to **playback-on-refer** or the ACLI **playback-trigger** parameter to **refer** (or **183-and-refer**) enables a REFER message to trigger playback. You configure this for the realm, session agent, or SIP interface for the transferrer, not for the transferee or the REFER target.

The REFER scenario requires that the Oracle Communications Session Border Controller performs local REFER termination, i.e., that it terminates the REFER locally. The SPL options you configure do not implement this behavior: You must configure local REFER termination separately. Proxying a REFER message is not a trigger.

Playback begins when the Oracle Communications Session Border Controller receives the REFER message, and stops when the REFER operation is deemed complete with a final response (200-299 or 400-699).



A call flow for the playback-on-refer scenario looks like this:

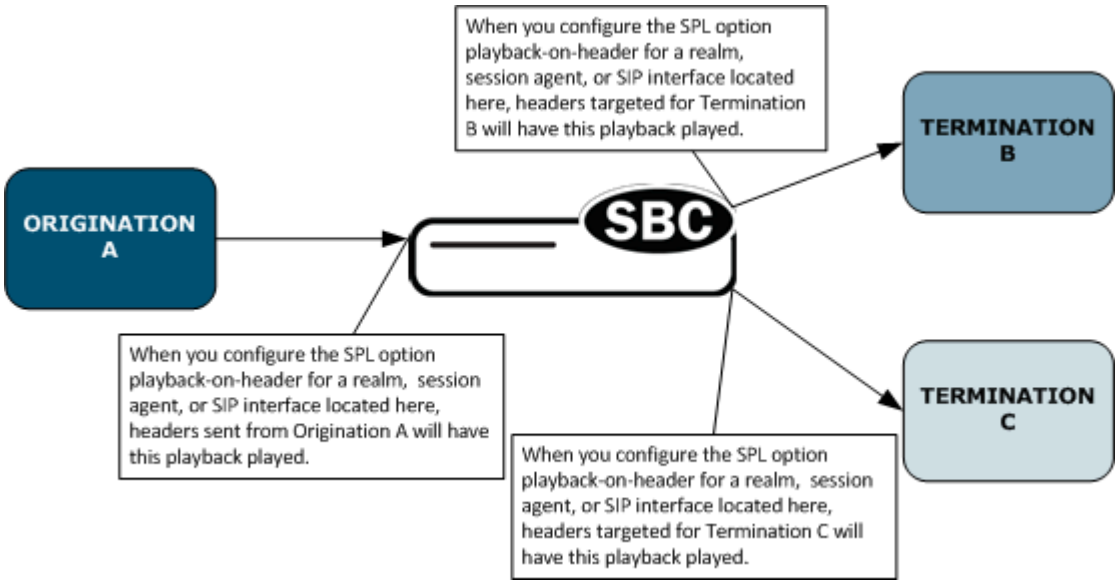


Playback Header

Setting the SPL options parameter to **playback-on-header** or the ACLI **playback-trigger** parameter to **playback-on-header** triggers in the presence of the P-Acme-Playback header. You can configure this on either the side receiving the header message or the side from which the message will be sent. If both trigger, then the configuration closest to the playback direction takes precedence.

This header can be part of any request or response, but playback can only start once media has been established. Playback stops automatically with a final response (200-299 or 400-699), unless explicitly turned off or another playback header requesting it to stop is received.

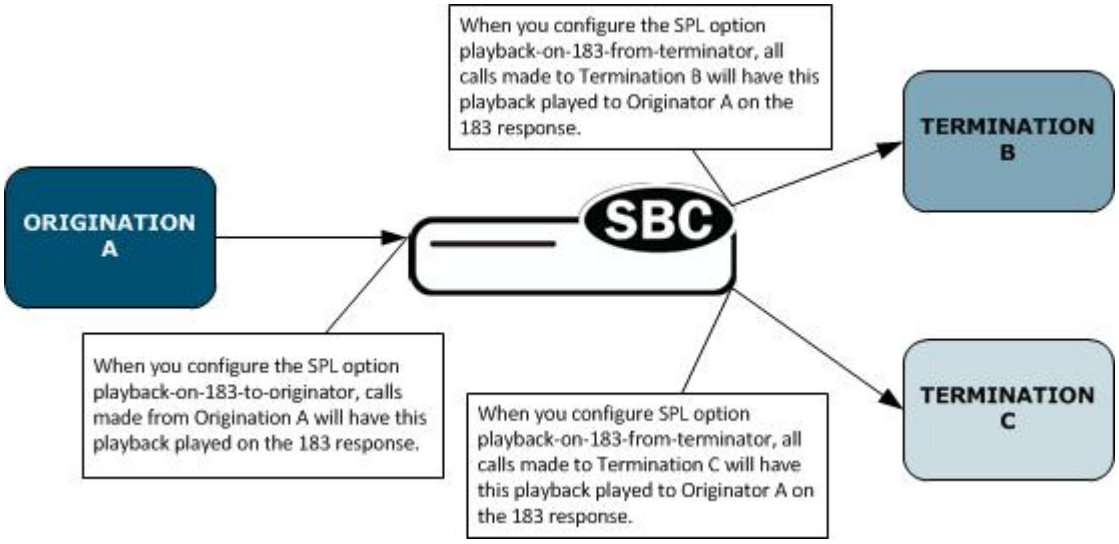
The Oracle Communications Session Border Controller deletes the P-Acme-Playback after processing if the SPL option is configured for the call (either incoming or outgoing).



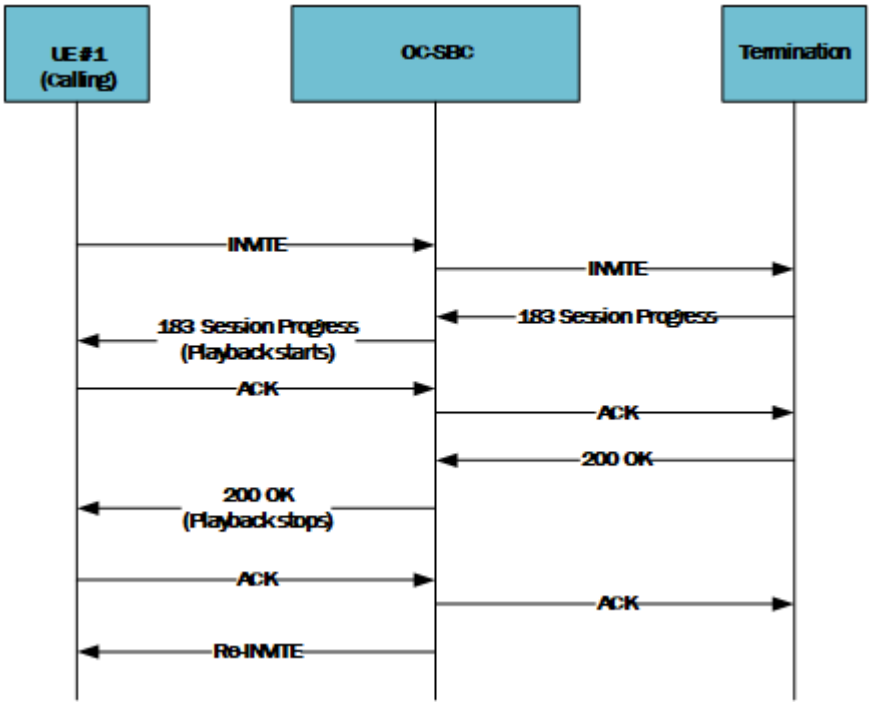
Playback on 183 Session Progress

This scenario is triggered by setting the SPL options parameter to either playback-on-183-to-originator or playback-on-183-from-terminator in realms, session agents, or SIP interfaces. When both options trigger, playback-on-183-to-originator takes precedence. This scenario triggers only for 183 Session Progress responses to initial INVITEs, not for re-INVITEs.

- playback-on-183-to-originator—Starts playback upon the receipt of a 183 Session Progress destined for the originator and stops when a either a (200-299 or 400-699) final response is sent. When you configure this option, every call sent from the originator triggers this playback.
- playback-on-183-from-terminator—Starts playback upon the receipt of a 183 Session Progress response is received from the terminator and stops when a (200-299 or 400-699) final response is received. When you configure this option, every call sent to the terminator triggers this playback.

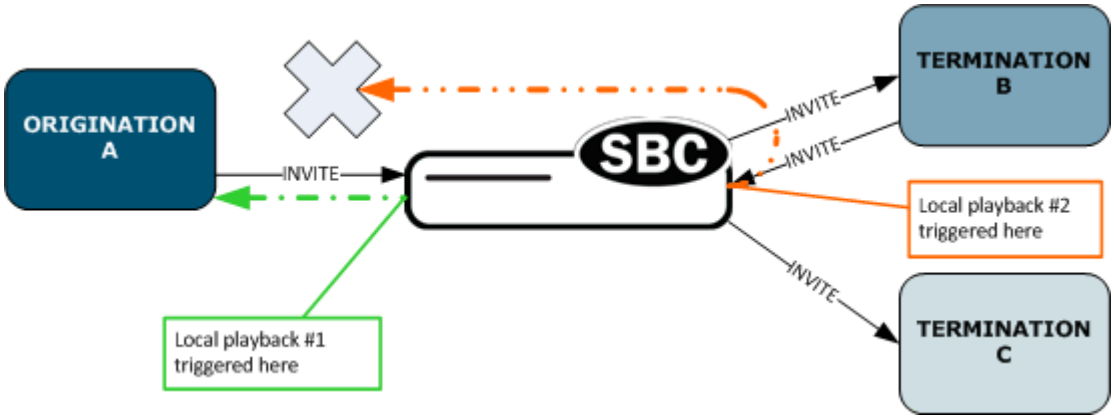


A call flow for the playback-on-183-from-terminator scenario looks like this:



Media Spirals

Certain call flows cause media to traverse the SBC multiple times, resulting in media spirals. For local playback, this means that multiple playback files can be triggered to play. In situations like this, the SBC uses the playback closest to the endpoint receiving the media playback. Origination A in the diagram below is played Local playback #1, even though the scenario also triggers Local playback #2.



Transcoding Free Operation for Local Media Playback

You can configure the SBC to avoid using transcoding resources within certain local media playback scenarios. After establishing a RBT call that includes transcoding, the SBC can trigger this Transcoding Free Operation (TrFO) feature if the P-Acme-TrFO header is present. Having determined that the call can proceed without transcoding, the SBC originates a reINVITE towards the calling party containing the called side codec. Once the reINVITE is completed, the call can continue without transcoding. In addition, the negotiated codec on the called party side must have been included in the calling party's original offer (after ingress codec-policy execution).

The SBC retains the codec list sent by the calling party, usually within the initial offer with all supported codecs. It also monitors the result of the ingress **codec-policy**. The SBC triggers the TrFO function for RBT when the following conditions are true:

- The P-Acme-TrFO header;
- One of the following are true:
  - The voice codecs are different on either side, or
  - The voice codecs are the same, but telephone event is missing from one side.

In this case, the SBC originates a reINVITE towards the calling party containing the called side codec. Once the reINVITE is completed, the call can continue without transcoding.

> **Note:**
>
> If the SBC detects fax tone, and you have enabled RTCP generation, the TrFO call flow can proceed, but the DSPs will still be in use afterwards so the feature can still be provided.

Navigate to the initial INVITE ingress realm's **realm-config** and configure the **feature-trfo** parameter using the syntax below. This is the same realm on which you have configured the **ringback-trigger**.

```
ORACLE(realm-config)# feature-trfo rbt
```

This 'manual' trigger by the SBC to force SDP re-negotiation prevents the call from requiring transcoding.

Related Configuration

You must also configure the following on the SBC for TrFO to work correctly with RBT:

- Transcoding enabled
- **realm-config**, **hide-egress-media-update** enabled
- **media-manager**, **options +unique-sdp-id**
- **media-manager**, **anonymous-sdp enabled**

TRFO reINVITE generation rules

The following table describes the general TrFO reINVITE generation rules when you set the **feature-trfo** parameter to **rbt** on the ingress realm and the P-Acme-TrFO header is present in either the provisional messages or the final 200OK.

Ringback-trigger Configuration	SDP injected	Transcoding	Codec is present in the initial INVITE	Is TRFO reINVITE generated?	Examples
When not set to any of these values: 180-no-sdp 180-force 183 183-no-sdp	N/A	N/A	N/A	No	TrFO is not triggered. TrFO reINVITE is not triggered because the negotiated callee codec was not present in the original INVITE offer.
180-no-sdp 180-force 183 183-no-sdp	No	N/A	N/A	No	TrFO is not triggered. TrFO reINVITE is not triggered because SDP was not inserted for RBT.
180-no-sdp 180-force 183 183-no-sdp	Yes	No	N/A	No	TrFO is not triggered. TrFO reINVITE is not triggered because there is not transcoding
180-no-sdp 180-force 183 183-no-sdp	Yes	Yes	No	No	TrFO is not triggered.

Ringback-trigger Configuration	SDP injected	Transcoding	Codec is present in the initial INVITE	Is TRFO reINVITE generated?	Examples
180-no-sdp 180-force 183 183-no-sdp	Yes	Yes - different caller/ callee audio codecs OR - same codec, telephone-event on one side	Yes	Yes	TrFO is Triggered.

RBT TrFO Flows

This section describes the TrFO flows when initiated by ringback tone. The flow descriptions use the Session Router as an example of an upstream or downstream service capable of HMR.

P-Acme-TrFO Call Flow

Data call identification is done via HMRs on both the SBC and the Session Router.

The need for TrFO results from HMRs execution (custom logic) and translates into a private header called P-Acme-TrFO, which is included in called party 200 OK or 18x messages ingress using HMR.

The SBC triggers TrFO feature whenever P-Acme-TrFO header is present, provided that:

- RBT triggers transcoding, based on the called party codec the SBC selects during call establishment.
- The negotiated codec on the called party side is included within the calling party side codecs supported in the original offer and after ingress the SBC executes the **codec-policy**.

TrFO for Data Calls

The need for TrFO results from HMRs execution (custom logic) and translates into a private header called P-Acme-TrFO (included in called party 200 OK ingress to SBC - using HMR).

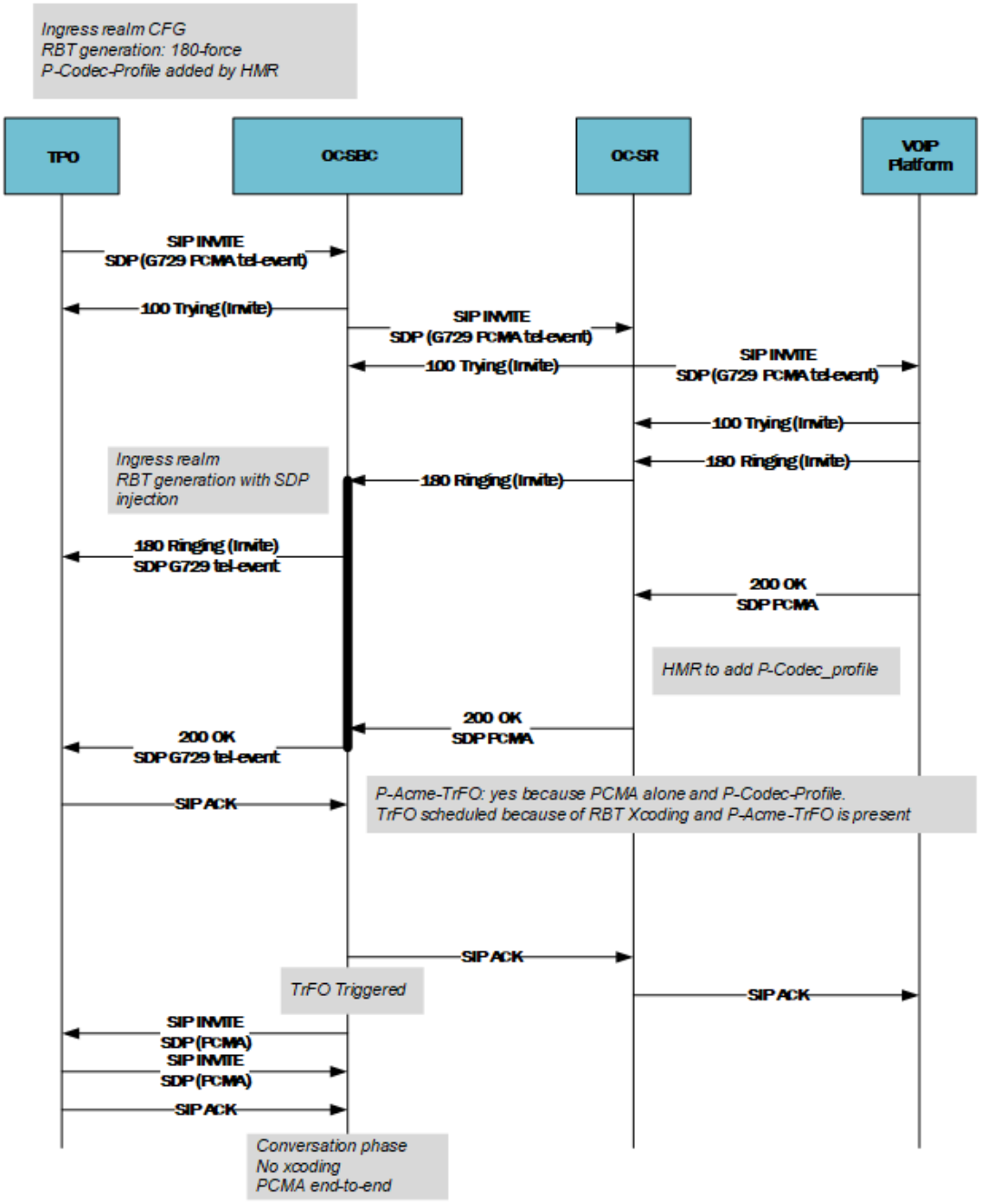
The SBC triggers the TrFO feature whenever the P-Acme-TrFO header is present, provided that:

- RBT induces transcoding due to called party codec selection by the SBC during call establishment.
- Negotiated codec on the called party side is included within the calling party side codecs supported in the original offer (after ingress codec-policy execution).

The SBC triggers the feature when the header is inserted by HMR or when it is present in the response. The HMR logic that triggers TrFO for data calls is as follows:

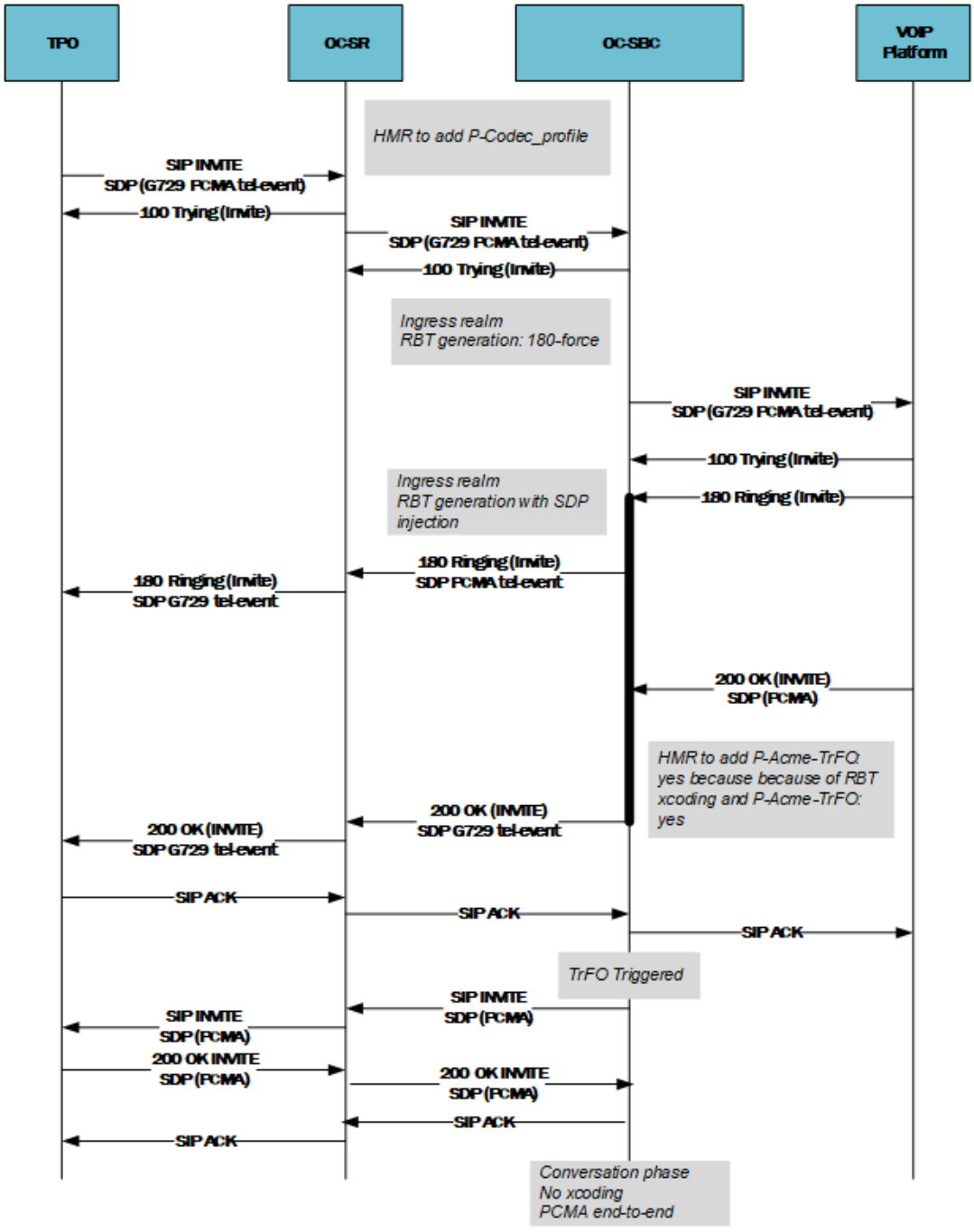
- For an Inbound call Flow:
  1. A known trunk that supports the data call has an Ingress HMR that adds the P-Codec-Profile Header.
  2. This HMR adds P-Codec-Profile: from the SBC to the Session Router.

3. An egress HMR on the Session Router adds the applicable flag based on the reception of P-Codec-Profile to the INVITE Via Header.
4. Topology hiding removes the P-Codec-Profile header.
5. On the reception of an answer from the distant UA, an Ingress HMR on the Session Router checks for the presence of the flag in the VIA, and for PCMA in the SDP.
6. If the previous condition is met, the Session Router sends the P-Acme-Trfo Header to the SBC, which then triggers TrFO.
7. If the 4 condition is not satisfied, the Session Router does nothing.



- For an Outbound Call Flow :
  1. A known trunk that supports the data call has an Ingress HMR that adds the P-Codec-Profile Header.

2. This HMR adds P-Codec-Profile: m2m from the Session Router to the SBC.
3. An egress HMR on the SBC adds the flag m2m based on the reception of P-Codec-Profile:m2m in the INVITE Via Header.
4. Topology hiding removes the P-Codec-Profile header.
5. On the reception of an answer from the distant UA, an Ingress HMR on the SBC checks for the presence of the m2m flag in the VIA, and for PCMA in the SDP.
6. If the previous condition is met, the SBC adds the P-Acme-Trfo Header to the Ingress Message of the SBC, which then triggers TrFO.
7. If condition 5 is not satisfied, the SBC does nothing.



This is an example and other logic can be used the same way to get to the same result.

RBT TrFO Reporting

The **show sipd rbt-trfo** command shows Ring Back Tone TrFO statistics:

- Initiated: number of initiated TRFO reINVITE transactions
- Success: number of successfully completed TRFO reINVITE transactions
- Failure: number of failed TRFO reINVITE transactions (either error returned or timed out)

Example:

```
ORACLE# show sipd rbt-trfo
 RBT TrFO statistics (2020-10-08 13:19:50.715)
             ---- Lifetime ----
            Recent Total PerMax
 Initiated    0      3      2
 Success      0      3      2
 Failure      0      0      0
```

You can reset this data back to zero using the **reset sipd** command.

Considerations for HA Nodes

On switchover, media set-up for playback is preserved, which requires negotiated codec and ptime for playback be transferred to the stand-by system in an HA node.

Regarding playback after switchover, any playback in progress does not continue on switchover.

While standard configuration replication handles transferring configuration information between the active and standby systems, media playback files (in /code/media) must be loaded onto the standby.

RTC Support

The playback configuration is supported by real-time configuration (RTC). Media files located in the /code/media directory and referenced by playback configuration entries are loaded at boot time and when you activate a configuration. The system does not reload any media being played to an endpoint. Playbacks that start after the boot or configuration activation use updated media files.

Alarms

These are the alarms for local playback. They are MAJOR in severity, and do not impact the system health score.

Alarm	Description
Playback media file not found or couldn't be loaded	Raised when a configuration is activated if the system cannot find a media file referenced configuration or if the system is unable to load the media file. This alarm clears automatically when a file is correctly referenced or when it is loaded properly. You might encounter this alarm if you have established playback configuration, but have not loaded the appropriate playback files to /code/media.
Playback could not be started due to capacity limit (Valid only when using the SPL configuration method.)	Raised at call time when system has reached its maximum number of playbacks (100). This alarm must be cleared manually.
Playback could not be started due to unsupported codec	Raised at call time when there is a mismatch of codecs between those in available files and one that must be played. This alarm must be cleared manually.

Monitoring

You can use the **show mbcd statistics** command to displays the number of media playbacks that are currently active, and for common period and lifetime totals.

```
ORACLE# show mbcd statistics
MBCD Status                      -- Period -- -------- Lifetime --------
                      Active    High   Total      Total  PerMax    High
Media Playback             0       5       5          6       0       6
```

You can use the **show mbcd errors** command to track the number of playback failures:

```
ORACLE# show mbcd errors
MBC Errors/Events              ---- Lifetime ----
                        Recent      Total  PerMax
Media Playback Fails         0          0       0
Playback Exh Resources       0          0       0
Playback Flow Inactive       0          0       0
Playback Mismatch            0          0       0
```

RBT TrFO Configuration

You configure the SBC to avoid using transcoding resources by setting the **feature-trfo** parameter in the applicable **realm-config** .

1. Access the media-manager configuration element

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
```

2. **feature-trfo**—Set the parameter to **ringback**.

```
ORACLE(realm-config)# feature-trfo ringback
```

3. Type **done** and save your configuration.

Configuring Local Media Playback with Transcoding Resources

You can configure local media playback with the transcoding-based method on a **session-agent**, **sip-interface** or **realm-config**, using this precedence when determining which configuration to use if there are overlapping configurations. This example procedure configures playback on a realm using the **180-force** trigger.

Although the procedure below sets parameters to a realm-config, you can apply the same parameters and values to a session-agent and a sip-interface.

1. In Superuser mode, use the following command sequence to access realm configuration:

```
ORACLE# configuration terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Set the ringback file.

```
ORACLE(realm-config)# ringback-file my-media.wav
```

3. Set the ringback trigger.

```
ORACLE(realm-config)# ringback-trigger 180-force
```

4. Use done and exit to complete the configuration.
5. Use save, verify-config and activate to apply the policy to the running configuration.

The **verify-config** command checks and reports on the following playback configuration issues:

- ringback-file refers to a file that exists under /code/media/
- ringback-file refers to a file that does not surpass the maximum size allowed 5 MB
- ringback-trigger and playbackConfig are not both configured

24

Advanced Media Termination Support

The Oracle Communications Session Border Controller (SBC) supports VoIP calls through the browser-based, real-time communication known as Advanced Media Termination. Using W3C and IETF standards, Advanced Media Termination supports cross-browser video calls and data transfers, such as browser-based VoIP telephony and video streaming. Advanced Media Termination allows users to make and receive calls from within a web browser, relieving the need to install a soft phone application. With Advanced Media Termination, the SBC can enable users to communicate concurrently with one or more peers through various browsers and devices to stream voice and data communications in real-time through a variety of web applications. Advanced Media Termination also supports communications through end-user clients such as mobile phones and SIP User Agents.

Advanced Media Termination supports clients

- connected to networks with different throughput capabilities.
- on variable media quality networks (wireless).
- on fire-walled networks that don't allow UDP.
- on networks with NAT or IPv4 translation devices using any type of mapping and filtering behaviors (RFC 4787).

Supported Advanced Media Termination Services

The SBC supports the following services and functions for Advanced Media Termination:

- ICE-STUN (Lite mode) - Interactive Connectivity Establishment - Session Traversal Utility for NAT (ICE-STUN) enables an Advanced Media Termination client to perform connectivity checks. Use ICE to provide several STUN servers to the browser by way of the application. ICE processing chooses which candidate to address. Other benefits include support for IPv4, load balancing, and redundancy. ICE STUN support requires configuring an **ICE Profile** and specifying the profile in **Realm Config**. See "Configure ICE Profile" and "Configure Advanced Media Termination in Realm Config."
- RTP-RTCP multiplexing - Enables Real-Time Protocol (RTP) and Real-Time Control Protocol (RTCP) packets to use the same media port numbers. RTP is used for real-time multimedia applications, such as internet audio and video streaming, VoIP, and video conferencing. RTCP is used to monitor data transmission statistics and QoS, and helps to synchronize multiple streams. RTP-RTCP support requires enabling **RTCP Mux** in **Realm Config**. See "Configure Advanced Media Termination in Realm Config."
- SIP services including codec renegotiation, late media, early media, PACK interworking, attended and unattended call transfer, call forking, music on hold, transcoding, and High Availability.

Supported Protocols

The SBC supports the following protocols for Advanced Media Termination.

- IPv4 for signaling and media
- UDP-RTP and UDP-RTCP on media

Supported Codecs

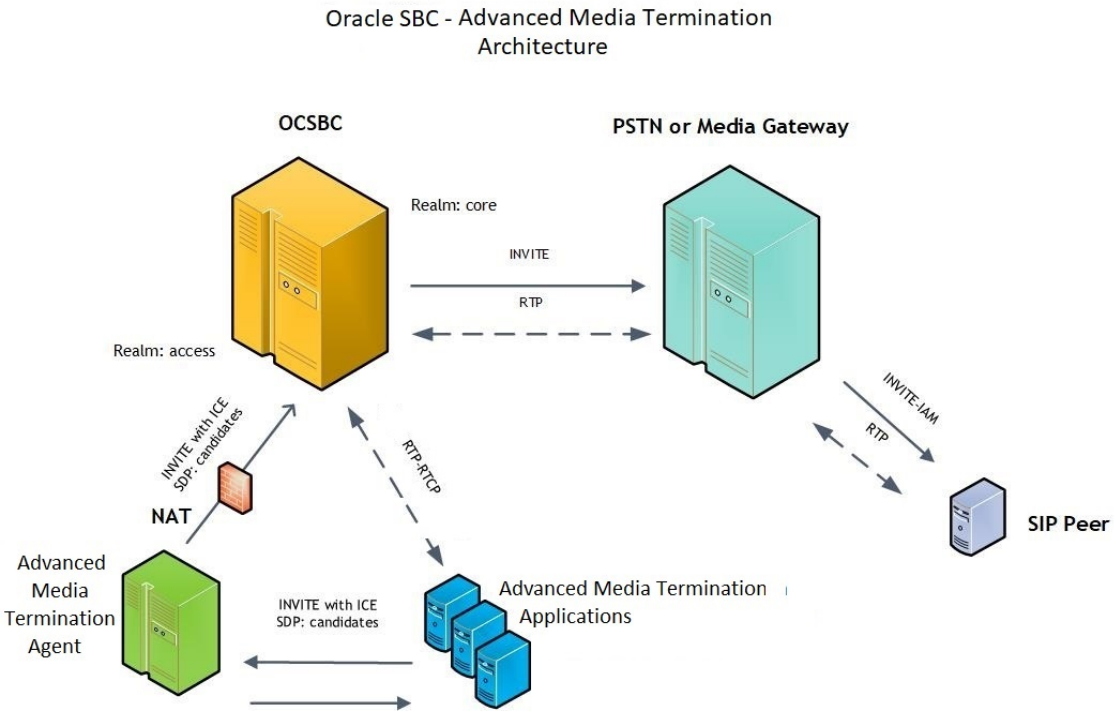
The SBC supports the following codecs for Advanced Media Termination.

- Silk, OPUS, G.729, and G.711

Advanced Media Termination Operations in the Network

The Oracle Communications Session Border Controller (SBC) can interconnect a Advanced Media Termination domain with a PSTN domain or another Advanced Media Termination domain.

The following illustration shows how the SBC supports Advanced Media Termination operations.



In the preceding illustration:

- The Advanced Media Termination Agent and the Advanced Media Termination Applications can communicate on a proprietary interface on a web socket.
- The Advanced Media Termination Agent can be behind a NAT interface with the SBC on a Advanced Media Termination configuration specific realm using SIP.
- The SBC can connect with a PSTN or Media Gateway on a different realm, for example "core," having RTP or SRTP media.
- In a call flow initiated by a Advanced Media Termination Client, the Advanced Media Termination Agent receives call setup from the Advanced Media Termination Client on a proprietary interface and the Advanced Media Termination Agent initiates a SIP session towards the SBC.
- The SBC can interwork with Advanced Media Termination and specific ICE-STUN and RTCP-Mux features towards the Advanced Media Termination Agent and a normal SIP, RTP-SRTP interface towards a PSTN-Media Gateway.

- SDES-SRTP media can flow directly from the Advanced Media Termination application to the SBC.

Additional STUN Candidate for RTCP

You can configure the SBC to establish a collapsed flow between itself and any STUN endpoint by enabling the **rtcp-stun** parameter in the applicable **ice-profile**. The system uses this flow for both RTP and RTCP, collapsing this traffic from two ports. As such, this configuration only applies when you have a realm supporting STUN with **rtcp-mux** disabled.

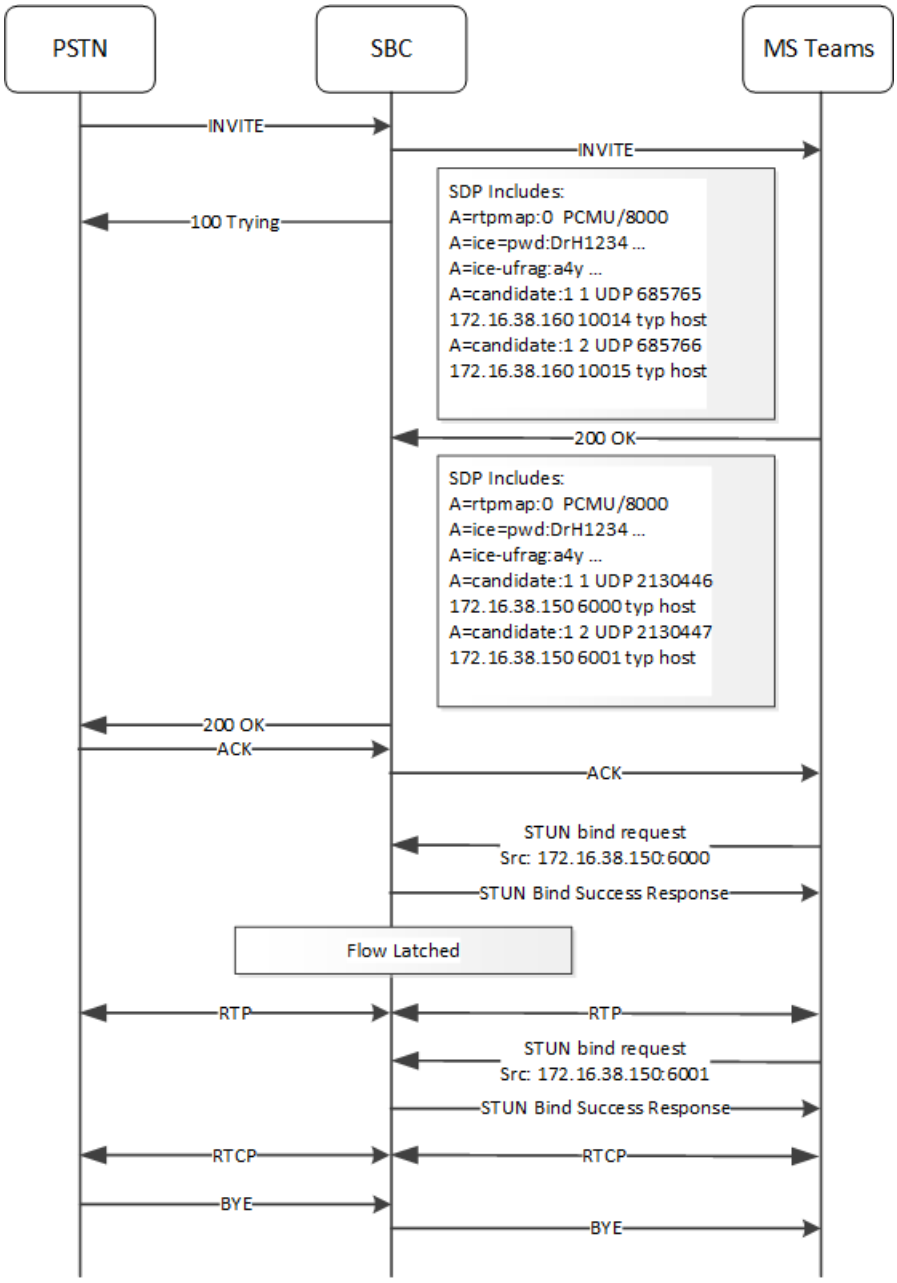
Although RTCP multiplexing can simplify advanced media deployments that use NAT, it does not adhere to the RFC 5245 requirements for ICE LITE. These requirements include using separate ports for RTP and RTCP. You use this configuration for ICE LITE deployments within which other components of the infrastructure require separate RTP and RTCP ports. When you enable **rtcp-stun** and leave **rtcp-mux** at its default of disabled, SBC behavior when it receives an INVITE that includes ICE includes:

- The source IP of RTCP STUN request is the same as the source IP of the RTP STUN request
- The source port of RTCP STUN request is (RTP STUN request port + 1)
  Your RTP port should always be an EVEN port number, and RTP+1 is always ODD.
- The destination IP of RTCP STUN request is the same as the destination IP of the RTP STUN request
- The destination port of RTCP STUN request is (RTP STUN request port + 1)
  Your RTP port should always be an EVEN port number, and RTP+1 is always ODD.

ICE deployment configuration includes enabling **rtcp-stun** on the **ice-profile**, applying that **ice-profile** to the applicable realms, and disabling **rtcp-mux** on both of your ICE call flow realms.

When you have configured this feature, applicable call signaling proceeds as follows:

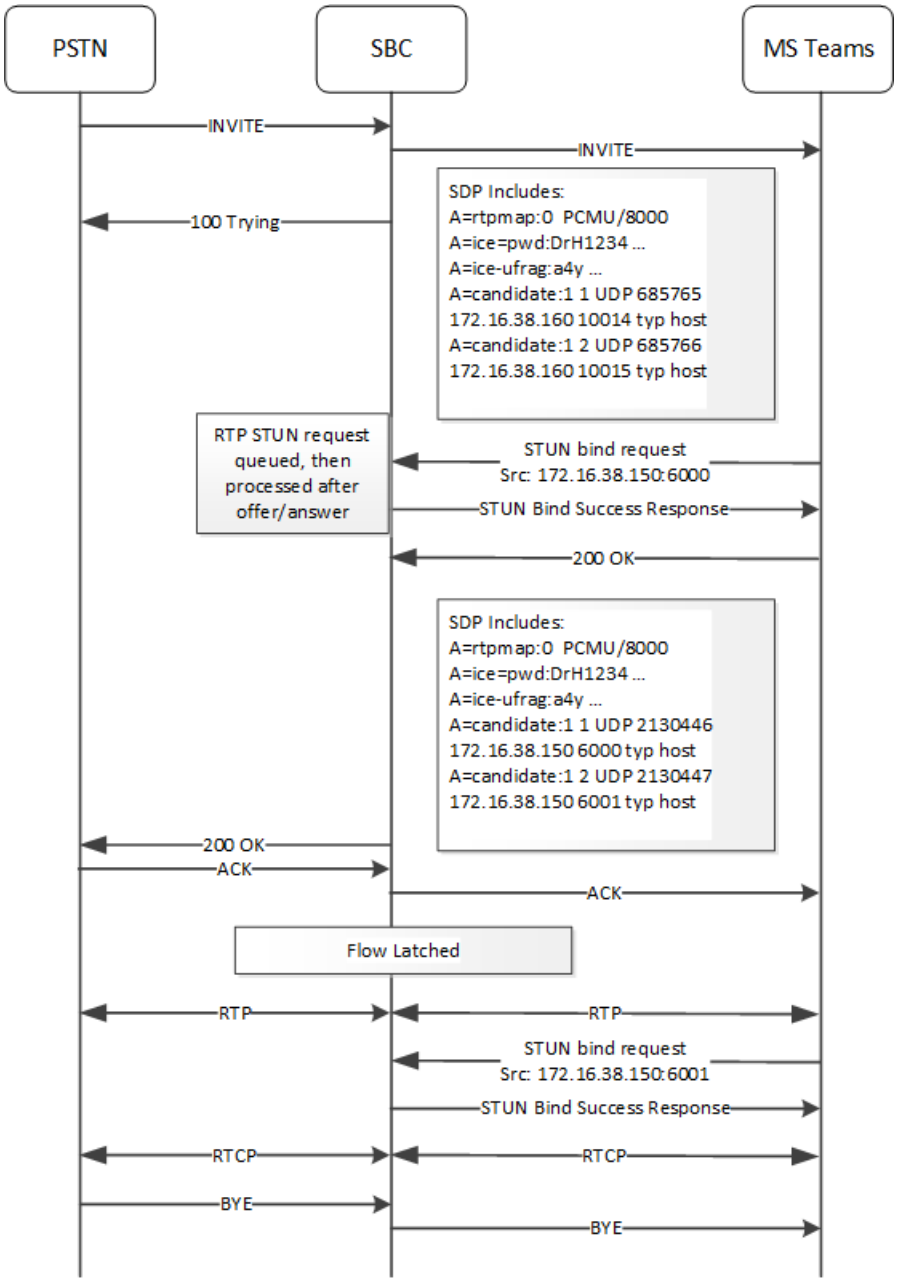
1. The SBC initiates ICE operation when it receives an INVITE that includes SDP with ICE attributes, effectively asking for an ICE call setup. This INVITE would also include two media candidates.
2. The SBC sends its own INVITE towards the callee, requesting ICE and providing two media candidates.
3. If the callee can support ICE, it responds by accepting the call.
4. After accepting, the callee issues an RTP STUN request to the RTP port advertised by the SBC.
5. Having received this request, the SBC replies with a STUN Bind Success Response and instantiates the RTP and RTCP flows. The system can then begin transmitting RTP and RTCP between the negotiated ports.
6. The callee issues an RTCP STUN request to the RTCP port.
7. Having received this request, the SBC replies with a STUN Bind Success Response.
8. When the callee sends periodic RTCP STUN requests, the SBC replies with success responses. These exchanges can be understood as a keepalive function.



Early STUN Bind Requests

In some call flows, the ICE end station may issue a STUN bind request prior to the call's 200 OK. In these cases, the SBC behavior is dependent on the type of bind request.

The call flow below presents this scenario with an RTP bind request. In this case, the SBC is able to latch the flow and start sending/receiving RTP media as soon as the 200 OK arrives. The call flow below also has the RTCP bind request arriving after the RTP flows are operational. The SBC responds by setting up the RTCP flow upon its successful bind reply.



If, however, the early bind request was for RTCP, the SBC would not queue the RTCP channel for setup after the 200 OK. In this case, the SBC needs to wait for the RTP bind request and issue a successful reply. As soon as the RTP bind request is successful, the SBC starts passing both RTP and RTCP media.

Feature Limitations

This section presents limitations to the Additional STUN Candidate for RTCP feature.

1. This feature works only when **rtcp-mux** is disabled on both the incoming and outgoing realms.
2. **hnt-rtcp** and **rtcp-stun** (Ice-lite feature flag) both are mutually exclusive. Make sure you disable your **hnt-rtcp** configuration when the Ice lite feature is enabled, and vice-versa.
3. Since the SBC does not support Teams to Teams calls, the Ice-lite feature does not support Teams to Teams scenarios.

4. The **skip-ice** option is not supported by this feature.
5. For MPO scenarios that include Proxy and Downstream SBCs in the path:
   - Make sure that the **ice-profile** configuration on each SBC is the same. This feature does not work properly if you have enabled **rtcp-stun** on one SBC but not on the other.
   - Disable the ICE configuration on the realm of the proxy SBC that faces the downstream SBC.
6. The following two scenarios are not supported by this feature because they are not supported on existing MS Teams implementation.
   - When handling an incoming call put on hold from the MS-Teams side, RTP and RTCP should flow towards the PSTN before the hold, but only RTCP should flow towards MS-Teams after the hold. After MS-Teams resumes the call, RTP and RTCP should flow both sides.
   - When handling an outgoing call put on hold from the MS-Teams side, RTP and RTCP should flow towards the PSTN before the hold, but only RTCP should flow towards MS-Teams after the hold. After MS-Teams resumes the call, RTP and RTCP should flow both sides.
7. This feature is not supported, if the endpoint on whose realm this feature flag is enabled sends an ODD media port for the RTP exchange in the offer/answer SDP. In this scenario, the SBC returns a 503 error towards the endpoint.

Advanced Media Termination Configuration Process

To configure Advanced Media Termination for theOracle Communications Session Border Controller, access the Media Manager configuration objects to create the necessary profiles and associations. For RTCP Multiplexing support, you need only to enable it in the target realm. Advanced Media Termination is configurable in real-time. The system does not require a reboot.

- Confirm that the realm you want to configure for Advanced Media Termination exists.

The process for configuring Advanced Media Termination includes the following tasks:

- In Media Manger:
  1. Configure **ICE Profile**, where you define STUN behavior. See "Configure ice-profile."
  2. Configure **Realm Config**, where you specify the **ICE Profile** and enable **RTCP Mux**. See "Configure Advanced Media Termination in Realm Config." (This assumes you have not enabled the **rtcp-stun** parameter.)

Configure ICE Profile

Interactive Connectivity Establishment - Session Traversal Utility for NAT (ICE STUN lite mode) enables a Advanced Media Termination client to perform connectivity checks, and can provide several STUN servers to the browser. ICE STUN support requires configuring an **ICE Profile** under **Realm Config**, where you define the STUN behavior.

- Confirm that the realm to which you want to apply this profile exists.

Use the following steps to create an **ICE Profile**.

1. Access the **realm-config** configuration object.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select **ice-profile**.
3. Do the following:

name	Set a unique name for this ice profile. Default: Empty.
stun-conntimeout	Set the maximum time interval, in seconds, between the first STUN binding request received in a media session and the time when a valid STUN binding request containing the USE-CANDIDATE attribute is received. Default: 10. Range: 0-9999.
stun-keepalive-interval	Set the interval, in seconds, since the last media packet or STUN binding request response after which a STUN keep alive message is sent. Default: 15. Range: 0-300. Zero means do not send keep-alive messages. The value must be less than the value set for subsq-guard-timer.
stun-rate-limit	Set the number of STUN binding requests that you want the SBC to process per minute. Default: 100. Range: 0-99999. Zero means impose no limit.
RTCP-STUN	Enable or disable the use of a STUN candidate for RTCP in the SDP of an INVITE that includes ICE in addition to RTP.

4. Type **done** to save your configuration.

- Set the **ICE Profile** parameter in **Realm Config**. See "Configure Advanced Media Termination in realm-config."

Configure Advanced Media Termination in Realm Config

To support Advanced Media Termination functionality, the Oracle Communications Session Border Controller (SBC) requires setting the parameters for **RTCP-Mux**, and **ICE Profile** in **Realm Config**.

- Confirm that the realm exists that you want to configure for Advanced Media Termination operations.
- Confirm that the **ICE Profile** exists.

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
```

```
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. Do the following:
   - RTCP-MUX: Specify "enable" to turn on RTCP multiplexing support. Default: Disable.

     > **Note:**
     >
     > This assumes you have not enabled the **rtcp-stun** parameter.

   - ICE Profile: Specify the ice-profile to associate with this realm. Default: Empty.
4. Type **done** to save your configuration.

Advanced Media Termination Troubleshooting

The Oracle Communications Session Border Controller (SBC) provides Session Traversal Utility for NAT (STUN) tracing.

To set STUN tracing, go to **Media Manager**, **Media Manager** and set **Options** to "stun-trace". The SBC stores the STUN traces in the Advanced Media Termination.log file.

Debug logs: log.sipd, log.mbcd, sipmsg.log, Advanced Media Termination.log

25
STIR/SHAKEN Client

STIR/SHAKEN is a framework of interconnected standards the SBC can use for authenticating calling parties in VoIP calls. To support STIR/SHAKEN, the SBC implements a STIR/SHAKEN REST Client, which, upon receiving an initial out-of-dialog SIP INVITE, sends a REST request to a STIR server for attestation or verification of the calling party identification. When you enable the STIR/SHAKEN entitlement, you can configure the SBC to perform the associated functions within ATIS or 3GPP environments. The SBC performs the functions of its STIR/SHAKEN client when triggered by the SIP INVITE method only.

> **Note:**
>
> The STIR/SHAKEN acronym combines the Secure Telephony Identity Revisited (STIR) and Secure Handling of Asserted information using Tokens (SHAKEN) standards acronyms to create this solution framework name.

You configure the STIR/SHAKEN REST Client functionality by creating server objects with the **sti-server** and the **sti-server-group** parameters and applying them to **session-agent**, **sip-interface** and **realm-config** elements, using this order to determine server selection precedence. Each **sti-server** provides you with the ability to configure it for either ATIS or 3GPP environments. In addition, the **session-agent**, **sip-interface** and **realm-config** elements provide parameters that allow some customization.

You configure and obtain operational statistics via the ACLI, ACP (and Oracle's session control management tools) and REST. You can also access client operational information via the HDR functionality, logs, and alarms on the SBC, as well as from SNMP walks, SNMP traps and Syslog configurations. Finally, the SBC also provides calling party authentication in CDRs.

> **Note:**
>
> This release of the Session Router supports version S-Cz9.1.0p2 STIR/SHAKEN functionality. Please refer to the S-Cz9.1.0 *ACLI Configuration Guide*, *MIB Guide*, and *HDR Guide* for STIR/SHAKEN documentation on the S-Cz9.2.0 Session Router.

HTTP Client Operating Modes

The SBC supports STIR/SHAKEN deployments that follow either the 3rd Generation Partnership Project (3GPP) or the Alliance for Telecommunications Industry Solutions (ATIS) specification. The ATIS group is a North American standards development group that is one of several 3GPP Organizational Partners. The STIR/SHAKEN Client Operation section details both authentication and verification behavior when configured for STIR/SHAKEN. All of those behaviors apply when operating in ATIS mode. When operating in 3GPP mode, however the SBC uses some different behaviors, especially with regards to verification.

ATIS architecture supports verifying SHAKEN passports. 3GPP supports verifying DIV passports in addition to SHAKEN passports. The DIV category refers to passports generated

for diverted calls. To prevent against intercepted calls, a SHAKEN passport includes both called and calling numbers, which can both be verified. But a diverted call includes a change to the calling number, which SHAKEN cannot support.

To alleviate this, the DIV passport can be used in addition to the SHAKEN passport. A DIV passport does not include an attestation level, with full attestation assumed. In addition to call diversion, the DIV standard also adds value to flows that need simultaneous ringing and emergency URN support. The SBC supports either architecture to provide you with flexibility within your deployments.

To define deployment type, the SBC provides the **http-rest-type** parameter under the **sti-server** element. You configure **http-rest-type** on a per-server basis to either **ATIS** (default) or **3GPP**. This sets the operating mode of that server.

Operating Under the ATIS Mode

By default, the The SBC operates its STIR/SHAKEN clients in ATIS mode. The behaviors described in the STIR/SHAKEN Client Operation section describe operation in ATIS mode.

When set to **ATIS** mode, the SBC sends and receives version ATIS-1000082 format SHAKEN PASSporT messages with STI-VS servers. In contrast with 3GPP mode, if the STIR server is configured in ATIS mode then the SBC does not generate DIV passports during authentication regardless of whether it detects a diversion in the call flow. In addition, the SBC only performs SHAKEN verification during ATIS verification. This is true even if the received INVITE has:

- SHAKEN passport
- SHAKEN and DIV passports
- only DIV passport
- other passports

Because the system is performing SHAKEN verification in these cases, it increments the "STI-VS received INVITEs with shaken PASSporT" counter for all of them.

STIR/SHAKEN Client Operation

The SBC can use STIR/SHAKEN to perform key security functions. These include authentication, which attempts to obtain identify information that was not provided, and verification, which attempts to verify identity information that has been provided. Authentication processes use AS servers; verification processes use VS servers. In addition to these, the SBC participates in the STIR/SHAKEN architecture to signing, or attestation, functions, within which the system distributes the results of these security checks.

Triggers to authentication and verification procedures include:

- If the SBC receives an out-of-dialog INVITE that includes a TN in either the PAI or FROM headers, it sends:
  - A signing request with the TN in the orig parameter if the "Identity" header is absent in the Invite and you have set the mandatory configuration on the SBC for initiating signing requests.
  - A verification request with the TN in the From header if the "Identity" header is present in the Invite and you have set the mandatory configuration on the SBC for initiating verification requests.
- If a TN is not present, meaning the SIP URI has no digits and only letters for the username in either the PAI or From header, the SBC does not initiate a signing request because there

is no valid TN for the orig parameter. Also, it does not initiate a verification request because there is no valid TN for the from parameter.

- If the PAI and From have different TNs, and the signing/verification fails with the PAI TN, the system does not try with the TN in the From.
  For example, a call requesting privacy may not contain a PAI and the From may contain anonymous@anonymous.invalid. In this case, the call would not be signed.

> **Note:**
>
> By default, the system applies precedence to incoming PAI headers when determining how to populate outgoing verstat and orig values. You can change this behavior globally or on a per-server basis, causing the system to consider information in FROM headers first, as described below.

Authentication

From a high level, authentication includes the following functions:

- Upon receiving an initial out-of-dialog SIP INVITE request that does not have a SIP identity header, the SBC sends a STIR AS request using REST. If the STIR server replies with a success response, the SBC adds the signed signature of the calling party to the INVITE request in a SIP identity header before routing the INVITE to the next hop.
- If preferred, you can configure the SBC to also use attestation level and origination ID headers from the ingress SIP INVITE in the REST query to the STI-AS. You do this by enabling the **sti-signaling-attest** on the applicable **realm**, **sip-interface** or **session-agent**, with the SBC using that configuration precedence to determine whether to perform the function.

```
(session-agent)# sti-signaling-attest enable
```

When this is configured, the Attestation-Info and Origination-ID headers override the configured values, if present. If one of the two requested headers is present, the other value is obtained from configured parameters.

> **Note:**
>
> If there are multiple instances of ‘Attestation-Info’ or ‘Origination-ID’, the SBC only inspects the first occurrence of these headers.

You can configure the SBC to generate an origination-ID or specify a server-associated origination-ID using the **orig-id** parameter in the applicable **sti-server** configuration. This parameter is required, but it accepts a full origination-ID or empty quotes (""). The empty quotes allow you to create the **sti-server** object without an actual origination-ID. When configured with empty quotes (""), the SBC generates its own origination-ID that identifies the SBC.

```
ORACLE(sti-server)# orig-id ""
```

Authentication Procedure

When the SBC receives an initial out-of-dialog SIP INVITE that does not have a SIP identity header, it determines the route for sending the INVITE and then sends an AS request to a STIR server if STIR/SHAKEN is enabled on the egress side. The SBC determines if STIR/SHAKEN is enabled by checking the following:

1. Is **sti-as** configured in the outbound **session-agent**
2. If not, is **sti-as** configured in the outbound **sip-interface**
3. If not, is **sti-as** is configured in the outbound **realm-config**

If **sti-as** is configured in one of the above steps, the SBC stops checking. The SBC uses the found **sti-as** to find the **sti-server** and uses information from the **sti-server** configuration to create and send the AS request.

1. The authority of the **as-server-root** specifies the address of the STIR server. If it is a FQDN then the SBC performs a DNS query to resolve the FQDN to an IP address. If the authority has a port, the AS request is sent to that port. Otherwise, port 80 is used if it is HTTP and port 443 is used if it is HTTPS.
2. The **sti-orig-id** and **sti-attest** from the inbound **session-agent**, **sip-interface** or **realm-config** are used in the AS request. The first values found in that order are used.
   The exception to this is when the INVITE message already has origId and attest values, and the parameter **sti-signaling-attest** is enabled in the inbound **session-agent**, **sip-interface** or **realm-config**. In that case, the system uses the origId and attest from the INVITE message. If there is no value in the INVITE, the SBC uses the applicable configured value.

   Additional detail includes:

   - If the **sti-orig-id** is not configured on any object, the SBC uses the value from the **sti-server**.
   - If the **sti-attest** is not configured on any object, the SBC uses the value from the **sti-server**.
   - The decision to send AS request is based on the sti-as parameter on the outbound side, but the values for origId and attest are obtained from the inbound side.
3. The AS request is sent over the network interface specified by the **http-client** that is configured in the **sti-server** configuration. If it is HTTPS, the **tls-profile** from the same **http-client** is used. Additional detail includes:

   - The SBC negotiates HTTP/2 by setting the ALPN parameter in the TLS negotiation, if HTTPS is used. If HTTP is used, only HTTP/1.1 is used. When HTTP/1.1 is used, the SBC maintains multiple TCP or TLS connections towards the STI server to avoid head-of-line issues with HTTP/1.1. If HTTP/2 is negotiated over TLS, only one HTTPS connection is established and HTTP/2 pipelining is used to send REST requests.
   - The SBC determines the interface using the realm you configure on the **sti-server**. If the **realm** field is empty, then the **ip-address** also has to be empty. In this case, the SBC uses the management interface to send the STI request
4. The SBC “parks” the call (i.e. pause processing the INVITE) and waits for a response from the STIR server for the amount of time configured in the **sti-server** timeout parameters. The default timeout is 2000 ms.
5. If the SBC receives a success response for the AS request, it adds a SIP Identity header the INVITE message using the identity returned in the AS response.

6. If the SBC does not receive a success response, the SBC does not add a SIP Identify header to the INVITE message. AS request timeout and failure to send due to internal error are both considered a failure to receive a success response.
7. In either case, the SBC “continues” the call by resuming the processing of the INVITE message to be sent to the next hop.

Verification

The SBC performs its verification process dependent on your configuration and the contents of the applicable INVITEs. Applicable configuration parameters includes the **sti-signaling-attest-info-mandatory** and **anonymous-uri-add-verstat-to-hostpart**, which are within the **sti-config** and disabled by default.

> **Note:**
>
> By default, the SBC sends signing requests to the server even if the INVITE does not include attestation and origination information.

Specifically, the SBC ensures that the TN is in either the PAI header, the FROM header or both before sending out any signing and verification request. When you enable **sti-signaling-attest-info-mandatory**, the SBC also refrains from sending the signing request if the INVITE does not contain attestation and origination information. In this case, the received INVITE must contain a P-Attestation-Info or Attestation-Info, and a P-Origination-ID header or Origination-ID header.

When you enable the **anonymous-uri-add-verstat-to-hostpart** parameter, and the received INVITE:

- Contains a Privacy header
- Contains a From header with an anonymous URI (For e.g. sip:anonymous@anonymous.invalid)
- Does not contain any PAI headers

The SBC adds the verstat parameter after the hostpart of the anonymous From URI. The SBC indicates privacy by placing this verstat in the hostpart of the FROM, as shown below.

```
From: ""Anonymous""<sip:anonymous@anonymous.invalid;verstat=TN-Validation-
Passed>;tag=9802748 " (V#12).
If privacy is requested then , then verstat=”No-TN-Validation”.
```

Finally, the SBC adds a reason header to 18x, 19x, 2xx, 4xx and 5xx responses to the INVITE if the verification fails.

> **Note:**
>
> PAI information may be presented in a single or in multiple headers. The behaviors below are the same if there are two separate PAI headers, or if the SIP URI PAI and TEL URI PAI are in the same header.

From a high level, verification includes the following tasks performed by the SBC:

- Upon receiving an initial out-of-dialog SIP INVITE request that has a SIP identity header, the SBC sends a STIR VS request to a STIR verification server for verification of the SIP identity. If the STIR server replies with a success response, the SBC adds the header parameter ‘verstat=TN-Validation-Passed’ to the “From:” header of the INVITE before it routes the INVITE to the next hop.
- If the SBC receives an INVITE without a PAI header, it adds the verstat parameter to the FROM header.
- The SBC adds this same header parameter to the “P-Asserted-Identity” header if it is present in the INVITE.
- If the SBC receives an INVITE with one SIP PAI header without a TN and a FROM header that has a TN, it adds the verstat parameter to the FROM header only.
- If the TN value from TEL URI PAI header is the same as the TN value from the FROM header. the SBC inserts a verstat parameter to both the FROM and the TEL URI PAI.
- If the SBC receives an INVITE with one PAI header, either a Tel PAI or a SIP PAI with a TN and:
  - A FROM with no TN, it adds the verstat parameter to the PAI header and not to the FROM Header.
  - A FROM with a TN, and if the values of the TNs match, it adds the verstat parameter to the PAI and FROM Headers.
- If the SBC receives an INVITE with two PAI headers, one of which is a TEL PAI and the other a SIP PAI and:
  - The SIP PAI has a TN, the SBC:
    - Compares each TN and, if all three TN's match, adds a verstat to all three headers.
    - Checks whether the FROM has a TN. If these TNs are not the same, it adds a verstat to the TEL PAI only.
    - If there is no FROM TN, the SBC compares the TNs in the SIP PAI and the TEL PAI, and:
      - If the TN's match, adds a verstat to both headers.
      - If the TNs do not match, add a verstat to the TEL PAI only.
  - If the SIP PAI has no TN, the SBC checks whether the FROM has a TN and whether that TN is the same as the TEL PAI TN:
    - If so, it adds a verstat to the TEL PAI and the FROM
    - If not, it adds a verstat to the TEL PAI only
- If the SBC does not receive a success response from the STIR server or times out while waiting for a response, you can configure the SBC to add the header parameter “verstat=No-TN-Validation” instead. This action is disabled by default.
- If the SBC receives a SIP INVITE without a SIP identity header on an ingress realm that is configured to trigger a STIR request to the STI-VS, the SBC adds the verstat parameter to the “From” and “P-Asserted-Identity” headers, when present, with a value of “No-TN-Validation”.

Verification Steps

When the SBC receives an initial out-of-dialog SIP INVITE that contains a SIP Identity header and STIR/SHAKEN is enabled on the ingress side, it sends a VS request to the STIR server

configured for the ingress side. The SBC determines if STIR/SHAKEN is enabled by checking the following:

1. Is **sti-vs** configured in the inbound **session-agent**
2. If not, is **sti-vs** configured in the inbound **sip-interface**
3. If not, is **sti-vs** is configured in the inbound **realm-config**

If **sti-vs** is found configured in one of the above steps, the SBC stops checking. The SBC uses the found **sti-vs** to find the **sti-server** and uses information from the **sti-server** configuration to create and send the VS request.

1. The authority of the **vs-server-root** specifies the address of the STIR server. If it is an FQDN then the SBC performs a DNS query to resolve the FQDN to an IP address. If the authority has a port, the VS request is sent to that port. Otherwise, port 80 is used if it is HTTP and port 443 is used if it is HTTPS.
2. The X-Instance-Id HTTP header on the REST request is set to the **sti-orig-id** from the inbound **session-agent**, **sip-interface** or **realm-config**. The first **sti-orig-id** found in that order is used. Additional detail includes:
   - If the **sti-orig-id** is not configured on any object, the SBC takes the value from the sti-server.
   - The decision to send AS request is based on the sti-as parameter on the outbound side, but the values for origId and attest are obtained from the inbound side.

   > **Note:**
   >
   > The parameters the SBC uses to decide on a query are always from the inbound side configuration, while the parameters the SBC uses to decide whether to make the query are dependent on the query type, as follows:
   >
   > - sti-as: inbound side configuration
   > - sti-vs: outbound side configuration

3. The VS request is sent over the network interface specified by the **http-client** that is configured in the **sti-server** configuration. If it is HTTPS, the **tls-profile** from the same **http-client** is used. Additional detail includes:
   - If an **auth-profile** is defined on the **http-client**, it refers to **security**, **authentication-profile**. An HTTP authorization header is sent with a Bearer token defined as the **preshared-key** in the **authentication-profile**.
   - The SBC negotiates HTTP/2 by setting the ALPN parameter in the TLS negotiation, if HTTPS is used. If HTTP is used, only HTTP/1.1 is used. When HTTP/1.1 is used, the SBC maintains multiple TCP or TLS connections towards the STI server to avoid head-of-line issues with HTTP/1.1. If HTTP/2 is negotiated over TLS, only one HTTPS connection is established and HTTP/2 pipelining is used to send REST requests.
   - The SBC determines the interface using the realm you configure on the **sti-server**. If the **realm** field is empty, then the **ip-address** also has to be empty. In this case, the SBC uses the management interface to send the STI request
4. The SBC “parks” the call (pauses the processing of the INVITE) and waits for a response from the STIR server for the amount of time configured in the **sti-server** timeout parameters. The default timeout is 2000 ms.

> **Note:**
>
> Upon receiving the INVITE, the inbound SIP manipulations are applied first. Then the caller information is obtained from the P-Asserted-Identity header, if it exists, or from the From: header. The destination number is extracted from the To: header. If the numbers are in e.164 format, the leading + sign is not sent on the STI request, but the numbers on the INVITE are not modified.

5. The SBC refers to the high-level verification rules above to determine how to proceed with header population and forwarding.
6. The SBC “continues” the call by resuming the processing of the INVITE message.

FQDN Server Root

The **as-server-root** and **vs-server-root** in the **sti-server** configuration allows a URL that uses IPv4, IPv6 or FQDN for the authority part. For FQDN, the SBC performs a DNS query for an A record (or AAAA record if you are using IPv6) to resolve the FQDN. The SBC sends AS/VS requests to the first IP address returned by the DNS server. Additional IP addresses are not used.

Adding Reason Headers

The SBC adds reason headers with reasoncode and reasontext information extracted from the JSON body of the response from STI-VS. The SBC adds these reason headers to all 18X, 19x and final responses, which include 200, 4xx, 5xx and 6xx messages, toward the calling party if verification fails. The system uses syntax defined by ATIS-1000074 version 17, as the example shows below.

```
Reason: SIP ;cause=436 ;text="Bad Identity Info"
```

No additional configuration is required to generate these reason headers.

Operating Under the 3GPP Mode

As discussed, the STIR/SHAKEN Client Operation section applies in full to ATIS mode operation. To a great extent, those behaviors also apply to 3GPP mode. But there are several differences between 3GPP and ATIS mode. Use this section as a compliment to STIR/SHAKEN Client Operation when reviewing 3GPP mode.

The SBC creates and processes SHAKEN as well as DIV PASSporT(s) when configured as an authentication entity, and you set the **http-rest-type** parameter within the applicable **sti-server** to **3GPP**. DIV PassporTs apply to call flows with diversion. When configured for **3GPP** and STIR verification, the SBC ignores and passes any PASSporT type that it cannot identify as SHAKEN or DIV.

Categories of operational differences between 3GPP and ATIS include:

- Available configuration parameters
- Identifying SHAKEN and DIV passports

> **Note:**
>
> The PASSporT type is specified by the “ppt=…” tag". The SBC allows tag formatting that uses either no quotes or double quotes.

- Behavior when presented with both SHAKEN and DIV passports
- Behavior when processing DIV passports

When you set the **http-rest-type** to **3GPP** within a **sti-server**, the SBC:

- Implements behavior supporting both Shaken and DIV verification requests per 3GPP TS 24.229.
- Implements behavior supporting both Shaken and DIV authentication requests per 3GPP TS 24.229.
- Sends and receives version TS 24.229 format SHAKEN and DIV PASSporT messages, including:
  - Authentication Requests to the STI-AS
  - Authentication Responses from the STI-AS
    - During authentication of an INVITE that includes call diversion, the SBC creates SHAKEN as well as DIV PASSporT(s)
  - Verification Requests to the STI-VS
  - Verification Responses from the STI-VS
- Uses the following configuration parameters in the **sti-config** to support 3GPP servers:
  - **tn-retargeting**—Values include disable (Default) or enable
  - **verstat-comparison**—Values include **TN-Validation-Passed** (Default), **No-TN-Validation**, **TN-Validation-Failed**, empty, or a list of values.
  - **dest-comparison**—Values include **Request-URI** (Default), and**To**
  - **check-duplicate-passports**—Values include disable (Default) or enable

Global 3GPP Behaviors

Behavior when you set **http-rest-type** to **3GPP** on a **sti-server** includes:

- When a received INVITE has a PASSporT that is not a SHAKEN or DIV PASSporT, the SBC ignores them and passes them through to the subsequent INVITE.
- When a received INVITE has a single PASSporT/Identity header with no ppt parameter, the SBC assumes it is a SHAKEN PASSporT and does not decode it.
- When a received INVITE has a SHAKEN and/or DIV PASSporT in addition to other PASSporTs that are not SHAKEN or DIV, the SBC attempts to verify the SHAKEN and DIV PASSporT(s), ignoring the others.
- When a received INVITE has multiple PASSporTs/Identity headers, and has no ppt parameter, the SBC performs a base64URL decode of each PASSporT to determine their PASSporT type.
- If a received INVITE has multiple verstat parameters that are not the same value, the SBC uses the following priority order for selecting the verstat value:
  1. Verstat value from the “P-Asserted-Identity” header with TEL-URI

2. Verstat value from the “P-Asserted-Identity” header with SIP-URI
3. Verstat value from the “From” header

- When configured for STIR/SHAKEN verification on the ingress interface and upon receiving an INVITE with a SHAKEN PASSporT and one or more DIV PASSporT(s), the SBC makes a SHAKEN and a DIV verification request to the STI-VS.
  When the STI-VS verifies PASSporTs, it sends the SBC a verstat response for the SHAKEN PASSporT and one or more verstat responses for the DIV PASSporT(s), referred to as verstatValue.
  - If any returned verstat values = “TN-Validation-Failed” then the SBC sets the outbound verstat in INVITE to “TN-Validation-Failed”.

    > **Note:**
    >
    > The reason for this behavior is that that “TN-Validation-Failed” is a more severe error that “No-TN-Validation”, and if “TN-Validation-Failed” appears for any verstat response, the system uses that for the outgoing verstat value.

  - If any returned verstat value = “No-TN-Validation” then the SBC sets the outbound verstat in the egress INVITE to “No-TN-Validation”.
  - Under all other conditions, the received verstat values would be “TN-Validation-Passed”. In these cases, the SBC sets the outbound verstat in the egress INVITE to “TN-Validation-Passed” and:
    * Retains the existing SHAKEN PASSporT in the outgoing INVITE
    * Retains the existing DIV PASSporT(s) in the outgoing INVITE
- When configured for STIR/SHAKEN verification on the ingress interface, and the received INVITE has one or more DIV PASSporT(s) but no SHAKEN PASSporT, the SBC behaves the same as it does when it receives no PASSporTs. In this situation, the SBC:
  1. Increments the STI-VS with DIV PASSporT(s) counter.
  2. Does not send a DIV PASSporT verification request to the STI-VS.
  3. Sets outgoing verstat value to “No-TN-Validation”.
  4. Retains the existing DIV PASSporT(s) in the outgoing INVITE
  5. Includes the SIP REASON header with the code of 436 and the text “Bad Identity Info” in provisional and final responses it sends in the reverse direction.
     The reason for this behavior is that this is an error scenario. The SBC, therefore, sends this reason in the reverse direction regardless of the **use-identity-header** parameter setting.

CDR Behaviors

The SBC uses information specific to STIR/SHAKE traffic and results to populate CDR fields, including:

- Stir-TN-Used-For-AS-VS-Request—During verification procedures, the SBC captures the TN number within the Stir-TN-Used-For-AS-VS-Request CDR field. the SBC uses for STIR-AS/STIR-VS, as a CDR attribute for STIR/SKAKEN only. The SBC uses the following priority order for selecting the TN it uses in the CDRs:
  1. From the Tel PAI, if present
  2. From the SIP PAI, if present and Tel PAI is not present

3. From the “From” header if both Tel and SIP PAI are not present.

- Stir-Div-Signed-Request—After the SBC receives a 200 OK to a DIV signing request from the STIR-AS, the SBC populates this CDR field with:
  - 0, indicating the STIR-AS is not signed
  - 1, indicating the STIR-AS is signed
  - 2, indicating signing is not applicable
- Stir-Div-Verified-Request—After the SBC receives a 200 OK to a DIV verification request from the STIR-VS, the SBC populates this CDR field with:
  - 0, indicating the STIR-VS did not verify the DIV
  - 1, indicating the STIR-VS verified the DIV
  - 2, indicating verification is not applicable

The SBC inserts these fields within START, INTERIM and STOP CDR records for STI-VS requests, and within INTERIM and STOP CDR records for STI-AS requests. When you configure the **generate-start** parameter to **OK** in the **account-config**, the SBC generates CDRs that also include the STI-AS fields for START events.

3GPP Behaviors Related to Configuration

The SBC includes configuration parameters that apply to 3GPP STIR/SHAKEN operation only. Your settings for these parameters generate important behaviors to consider.

The scenarios below assume you have configured the SBC for STIR/SHAKEN authentication on the egress interface or for verification on the ingress interface. In addition, you must have set the associated STI-AS or STI-VS server's **http-rest-type** to **3GPP**.

Behaviors Based on verstat-comparison Configuration

You can set the **verstat-comparison** parameter, within the **sti-config**, to **TN-Validation-Passed** (default), **empty**, **No-TN-Validation** or **TN-Validation-Failed**. The **verstat-comparison** parameter accepts a single or multiple values:

- When the received INVITE has a SHAKEN PASSporT, a DIV PASSporT, and the **verstat-comparison** value is empty, the SBC:
  1. Does a base64URL decode of the last (outermost) DIV PASSporT. If this decode fails, the SBC does not perform any SHAKEN or DIV authentication, instead passing the existing passport to the egress INVITE. If this decode succeeds, the SBC proceeds to step 2.
  2. Checks to see that the {dest} value from the last DIV PASSporT is NOT equal to the Request-URI TN value
  3. If so, the SBC:
     a. Makes div authentication request to the STI-AS
     b. Adds div PASSporT to the outgoing INVITE as a SIP Identity header
     c. Passes the existing DIV PASSporT(s) to the outgoing INVITE
     d. Passes the existing SHAKEN PASSporT to the outgoing INVITE
- Consider the scenario above, changing it such that the {dest} value from the last DIV PASSporT is equal to the Request-URI TN value. In this case, the SBC:
  1. Passes the existing DIV PASSporT(s) to the outgoing INVITE

2. Passes the existing SHAKEN PASSporT to the outgoing INVITE

The SBC recognizes that no diversion has occurred.

- When configured for STIR/SHAKEN authentication on the egress interface, the received INVITE has SHAKEN and DIV PASSporTs, and the **verstat-comparison** value is not empty, the SBC:

  1. Does a base64URL decode of the received div PASSporT
  2. Determines the last (outermost) div PASSporT.
  3. Checks to see that:
     - The {dest} value from the last div PASSporT is not equal to the Request-URI TN value, and
     - The verstat value from the received INVITE does matches a value from **verstat-comparison**
  4. If so, the SBC:
     a. Makes DIV authentication request to the STI-AS
     b. Adds the DIV PASSporT to the outgoing INVITE as a SIP Identity header
     c. Passes the existing DIV PASSporT(s) to the outgoing INVITE
     d. Passes the existing SHAKEN PASSporT to the outgoing INVITE

- Consider the scenario above, changing it such that the verstat value from the received INVITE does NOT match a value from **verstat-comparison**. In this case, the SBC:

  1. Passes the existing DIV PASSporT(s) to the outgoing INVITE
  2. Passes the existing SHAKEN PASSporT to the outgoing INVITE

  In this case, the SBC recognizes that a diversion has occurred.

- When the received INVITE has a SHAKEN PASSporT, no DIV PASSporT, and the **verstat-comparison** value is not empty, the SBC checks to see that:

  - The {dest} value from the SHAKEN PASSporT is not equal to the **dest-comparison** AND
  - The verstat value from the received INVITE matches a value from **verstat-comparison**. If so, the SBC:

  1. Makes DIV authentication request to the STI-AS
  2. Adds DIV PASSporT to the outgoing INVITE as a SIP Identity header
  3. Passes the existing SHAKEN PASSporT to the outgoing INVITE

- Consider the scenario above, changing it such that the verstat value in the received INVITE does NOT match a value in the **verstat-comparison**. In this case the SBC passes the existing SHAKEN PASSporT to the outgoing INVITE. In addition, the SBC does not perform DIV authentication.

- When the received INVITE has a SHAKEN PASSporT, no DIV PASSporT, and the **verstat-comparison** value is empty, the SBC:

  1. Does a base64URL decode of the SHAKEN PASSporT. If this decode fails, the SBC does not perform any SHAKEN or DIV authentication, instead passing the existing passport to the egress INVITE. If this decode succeeds, the SBC proceeds to step 2.
  2. Checks to see that the {dest} value from the SHAKEN PASSporT is NOT equal to the **dest-comparison** value.
  3. If so, the SBC:

a. Makes DIV authentication request to the STI-AS
b. Adds DIV PASSporT to the outgoing INVITE as a SIP Identity header
c. Passes the existing SHAKEN PASSporT to the outgoing INVITE

- Consider the scenario above, changing it such that the {dest} value from the SHAKEN PASSporT is or is not equal to the **dest-comparison** value. In this case, the SBC passes the existing SHAKEN PASSporT to the outgoing INVITE.
  The SBC recognizes that no diversion has occurred. In addition, the SBC does not perform DIV authentication.

Behaviors Based on dest-comparison Configuration

When presented with a received INVITE that has a SHAKEN PASSporT, but does not have DIV PASSporT, and the **verstat-comparison** value is empty, the SBC:

1. Does a base64URL decode of the received SHAKEN PASSporT
2. Checks to see that the {dest} value from the last SHAKEN PASSporT is not equal to the **dest-comparison**. If not, the SBC:
   - If the {dest} value is not the same as the **dest-comparison**, the SBC:
     a. Makes a DIV authentication request to the STI-AS
     b. Adds the DIV PASSporT to the outgoing INVITE as a SIP Identity header
     c. Passes the existing SHAKEN PASSporT to the outgoing INVITE
   - If the {dest} value is the same as the **dest-comparison**, the SBC passes the existing SHAKEN PASSporT to the outgoing INVITE

Behaviors Based on tn-retargeting Configuration

You can set the **tn-retargeting** parameter to **enabled** (default) or **disabled**.

- When the egress interface is configured for STIR/SHAKEN authentication, the received INVITE does not have a SHAKEN PASSporT, and you have set the **tn-retargeting** parameter set to:
  – disabled—The SBC performs SHAKEN PASSporT authentication by sending a SHAKEN authentication request to the STI-AS. Additionally, after receiving a successful response, the SBC adds the SHAKEN PASSporT to the outgoing INVITE as a SIP Identity header
  – enabled—The SBC checks to see if “To” header TN value is not the same as the Request-URI TN value.
    * If so, the SBC makes separate SHAKEN and DIV authentication requests to the STI-AS. Additionally, the SBC adds both the SHAKEN and DIV PASSporTs to the outgoing INVITE as two separate SIP Identity headers in the outgoing INVITE.
    * If the values are the same, the SBC only makes a SHAKEN authentication request to the STI-AS. The system also add a SHAKEN PASSporT to the outgoing INVITE as a SIP Identity header in the outgoing INVITE.

Behaviors Based on check-duplicate-passports Configuration

You can enable or disable the **check-duplicate-passports** parameter.

- When presented with a request that has multiple PASSporTs and you have enabled **check-duplicate-passports**, the SBC checks whether they are all the same. If so, the SBC removes the duplicate passports and handles the remaining passport normally.

When presented with a request that has multiple PASSporTs and you have **check-duplicate-passports** is disabled, the SBC checks whether they are all the same. If so, the SBC does not perform STIR/SHAKEN verification, instead passing all the PASSporTs through, and sets the verstat to “No-TN-Validation” in the outgoing INVITE). In addition, the SBC includes a SIP REASON header with the code of 436 and the text “Bad Identity Info” in the reverse direction, in both provisional and final responses.

Behaviors Based on use-identity-header Configuration

When you enable **use-identity-header**, if the received INVITE doesn't contain an Identity header and the SBC is configured to perform STI verification, the system adds a Reason header and sends it in the reverse direction in 18x, 2xx as well as final responses (3xx, 4xx, 5xx, 6xx) to the originator with a cause value of “428” and the text “Use Identity Header”.

If a SIP Reason header is already present, the system appends the existing Reason header with the new data at the end after a semi-colon.

3GPP Behaviors Related to Errors

When you use 3GPP mode, the SBC introduces specific error condition behaviors that apply to 3GPP STIR/SHAKEN operation only.

The SBC can respond to the same errors conditions during both authentication and verification procedures. The errors are divided in two categories:

1. Service Error - The server provides a service error when the server is unable to process the request
2. Policy Error - The server provides a policy error when the server is able to process the request, but not able to complete the service execution due to a policy restriction.

The following are the different types of Service errors that the SBC could receive:

Exception text	HTTP status code	Description
Exception text	HTTP status code	Description
Error: Missing request body.	400	The request could not be processed due to missing request body.
Error: Missing mandatory parameter.	400	The request could not be processed due to missing parameters.
Error: Requested response body type is not supported.	406	The request could not be processed due to a not supported message body format.
Error: Requested resource not found.	404	The request could not be processed due to no resource available related to the Request-URI
Error: Unsupported request body type.	415	The request could not be processed due to not supported message body.
Error: Invalid parameter value.	400	The request could not be processed due to invalid parameter value.
Error: Failed to parse message body.	400	The request could not be processed due to failure to parse the message body.

Exception text	HTTP status code	Description
Error: Missing mandatory Content-Length headers	411	The request could not be processed due to a missing Content-Length header.

The following are the different types of Policy errors that the SBC could receive:

Exception text	HTTP status code	Description
Method not allowed	405	The resource was invoked with unsupported operation
Internal server error.	500	The request failed due to internal error

Important error behaviors include:

- If the SBC receives an error (4xx or 5xx) HTTP response from the STI-AS server for a SHAKEN or DIV request, the system continues processing the call without adding any SHAKEN or DIV passport.
- The SBC increments the following counters associated with the response when it receives a Policy error:
  - STI-AS Unsuccessful Responses
  - STI-AS Policy Exception
- The SBC increments the following counters associated with the response when it receives a Service error:
  - STI-AS Unsuccessful Responses
  - STI-AS Service Exception
- If it receives an INVITE that does not contain an identity header during verification procedures, the SBC includes a SIP REASON header with the code 436 and the text of “Bad Identity Info” in provisional and final responses it sends in the reverse direction.
- If it receives an error (4xx or 5xx) response from the STI-VS server for a SHAKEN or DIV verification request, the SBC adds the vertstat="No-TN-Validation" parameter to the appropriate PAI and FROM and continue processing the call.
- If it receives an error (4xx or 5xx) response from the STI-VS server for a SHAKEN or DIV verification request, the SBC adds the SIP Reason header to the responses (1xx, 2xx, etc.) and sends it in the reverse direction. It also places the cause information taken from the HTTP status-code and text value from the HTTP reason-phrase (if present) in the SIP header.
  The SBC gets the cause value it puts in the SIP header from the HTTP status-code (e.g. 400). If the HTTP\REST response from the STI-VS has a reason-phrase, such as “Missing request body”, the SBC includes it. If the HTTP\REST response from the STI-VS does not have a reason-phrase, the SBC does not include a text value in the SIP Reason header.

  The SBC sends the following Reason header in responses sent to the originator: Reason: SIP ;cause=400 ;text="Missing mandatory parameter"

  When there is no reason phrase in the response, the SBC sends the Reason header without text. If there is reason header in the SIP response already, the SBC appends the new reason parameters into the header at the end after a semi-colon.

The example below shows a new reason added by the SBC after an existing 580/ Precondition Failure reason.

```
-Reason: SIP ;cause=580 ;text="Precondition Failure";
cause=400 ;text="Missing mandatory parameter"
```

Extract verifyResults Content for DIV Passports for 3GPP

In the case of 3GPP, if the response from the STI server includes multiple reason values, the SBC includes two reason values in the SIP Reason header:

- cause & text specific to shaken PASSporT verification
- cause & text specific to first div PASSporT verification

By default, the SBC does not extract and parse the content in the verifyResults header during 3GPP verification response processing. When you enable the **man-compliance** option, however, the SBC does perform this function. This allows the SBC to fetch the reasonCode and reasonText based on PPT type SHAKEN and the first index of PPT type DIV.

The SBC is required to fetch reasonText and reasonCode based on PPT type and order of DIV PPTs received within the context of its current implementation.

Extracting and parsing this header allows the SBC to use this reasonText and reasonCode information to better identify a verstat of, for example TN-Validation-Failed. The SBC ultimately uses the corresponding DIV ppt received in the VerifyResults to populate reasonText and reasonCode.

> **Note:**
>
> The SBC identifies the first index of a DIV PPT based on how it is received in the incoming INVITE.

An example of an applicable 3GPP verification response is presented below.

```
{"verificationResponse":
{
"verifyResults": [
{"verifyResult": {"status": "fail", "ppt": "shaken", "reasonCode": 438,
"reasonText": "Identity signature payload does not match the re-constructed
one", "passport": "eyJhbGciOiJFUzI1NiIsInBwdCI6InNoYWtlbiIsIn....."}},
{"verifyResult": {"status": "fail", "ppt": "div", "reasonCode": 438,
"reasonText": "Identity signature payload does not match the re-constructed
one", "passport": "eyJhbGciOiJFUzI1NiIsInBwdCI6uPuDqJ8ua..........."}}
].
 "verstatValue": "TN-Validation-Failed"
}
}
```

Number Authentication Mechanism Standards Compliance Features

You can configure the SBC to include compliance behaviors for the Number Authentication Mechanism 2.0 operations standards by enabling the **man-compliance** option in the **sti-config**. The Number Authentication Mechanism 2.0 standard, often referenced as MAN 2, is a French standard for CLI anti-spoofing. It was developed via a collaborative effort between the French Telecoms Federation (that unifies telco operators in France) along with ARCEP (Regulatory Authority) and the APNF (French association of standardization platforms for inter-operator flows).

When you enable the **man-compliance** option in the **sti-config**, you change default behaviors to comply with the Number Authentication Mechanism 2.0 standard by implementing the following non-default behaviors:

- STI-VS Trigger processing
- Provide enhanced support for SIP Reason Header added to egress INVITE
- Provide enhanced CDR support with parameters used in STI-AS/VS Requests

Enable the **man-compliance** option in the **sti-config** using the syntax below.

```
ORACLE(sti-config)# options +man-compliance=enabled
```

Changing the STI-VS Trigger

To optimize processing and reduce unnecessary traffic, enabling the **man-compliance** option allows the SBC to trigger STI-VS processing after it completes HMR and DoS processing, which includes constraint checks. When configured in this way, if a call attempt is below session-constraints set for the corresponding ingress Session Agent, sip-interface or realm, the system makes its request to the STI-VS server, and continues processing the STI-VS response regardless of constraints status.

By default, the SBC triggers STI-VS processing after ingress Header Manipulation Rules (HMR) processing. Ingress HMR processing occurs immediately after software Denial of Service (DoS) handling, which includes CPU load monitoring, message parsing, access control, and constraints checking. This sequence of operations leads to the system triggering STI-VS exchanges before the session constraints are evaluated, resulting in unnecessary processing on both the SBC and the STI server, and generating additional, often avoidable traffic processing.

Default call processing stages:

1. SIP message parsing
2. Denial of Service (DoS) handling (including CPU load monitoring, message parsing, access control, and constraints)
3. Inbound HMR (Header Manipulation Rules)
4. Load Index (LI) processing
5. STIR-VS processing
6. Check session constraints

Call processing stages after enabling the **man-compliance** option:

1. SIP message parsing
2. Denial of Service (DoS) handling (including CPU load monitoring, message parsing, access control, and constraints)
3. Inbound HMR (Header Manipulation Rules)
4. Load Index (LI) processing
   (Steps 5 and 6 are reversed.)
5. Check session constraints
6. STIR-VS processing

By moving the STI-VS trigger to occur after the system completes software DoS processing, the system can streamline call processing logic and improve efficiency, ultimately reducing unnecessary traffic processing.

Including STI-VS Failure Reasons in INVITEs

When you enable the **man-compliance** option in the **sti-config**, the SBC adds STI-VS failure reasons information to SIP reason headers for applicable egress INVITE messages. This improves your ability to identify failure scenarios.

Having enabled **man-compliance**, the SBC populates SIP Reason headers in egress INVITEs with STI-VS failure information after receiving a 4xx/5xx answer from an STI-VS request, including:

- HTTP response code
- JSON reason phrase/text as described in ATIS 1000082 (8.2.4.6 HTTP Response Codes) and in 3GPP TS 124 229 (V.2.4.3.2 Service errors)

Additional behaviors include:

- If the reason code is empty, eg, reasonCode: "", the system uses the HTTP response code.
- If the reason code is a string within the double quotes, for example reasonCode: "438", then the system treats reasoncode value as empty.

The SBC behaves differently when there is a single or multiple reason values presented. This section proceeds with descriptions of behaviors following single reason values first, and then multiple values. Examples of each are included.

When there is a single reason value in the HTTP response in either ATIS or 3GPP mode, the SBC populates the SIP Reason Header as follows:

- cause:
  - reasonCode (included in JSON body)
  - OR HTTP response code if reasonCode is not available in JSON body
- text:
  - reasonText (included in JSON body)
  - OR error (included in JSON body) if reasonText is not available in JSON body
  - OR HTTP reason phrase if reasonText or error is available in JSON body

Single Reason Value - Example 1

In this case, the reasonCode and reasonText are included in the JSON Body. The HTTP response may appear as follows.

```
HTTP/1.1 430 Bad Request
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
"verificationResponse": {
"verifyResults": [
 {
"verifyResult": {
"status": "fail",
"ppt": "shaken",
"reasonCode": 438,
"reasonText": "Bad Identity Info",
 "passport": "eyJhbGciOiJFUzI1NiIsInBwdCI6InNoYWtlbiIsIn....."
 }
 }
 ],
 "verstatValue": "TN-Validation-Failed"
 }
}
```

In this case, the Expected Reason header is as follows.

```
Reason: SIP ; cause=438 ;text="Bad Identity Info"
```

Single Reason Value - Example 2

In this case, the reasonCode and error are not included in the JSON Body, but error text is included in the JSON Body. The HTTP response may appear as follows.

```
HTTP/1.1 400 Bad Request
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
 “requestError”:
      {“serviceException”: {“messageId”: “SVC4001”
 “text”:
      “Error: Missing mandatory parameter ‘%1’”,“variables”:
      [“iat”]}         }}
Expected Reason header:
Reason: SIP ; cause=400 ;text=" Error: Missing mandatory parameter ‘%1’"
```

In this case, the Expected Reason header is as follows.

```
Reason: SIP ; cause=400 ;text=" Error: Missing mandatory parameter iat
```

Single Reason Value - Example 3

In this case, the reasonCode is included in the JSON Body, but no reasonText or error text is included. The HTTP response may appear as follows.

```
HTTP/1.1 430 Bad Request
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
 "verificationResponse": {
 "verifyResults": [
 {
 "verifyResult": {
"status": "fail",
 "ppt": "shaken",
"reasonCode": 438,
 "passport": "eyJhbGciOiJFUzI1NiIsInBwdCI6InNoYWtlbiIsIn....."
 }
 }
 ],
 "verstatValue": "TN-Validation-Failed"
 }
 }
 }
```

In this case, the Expected Reason header is as follows.

```
Reason: SIP;cause=438;text="Bad Request"
```

Single Reason Value - Example 4

In this case, the Response is a 200OK, and neither the reasonCode, reasonText or error text is included in the JSON Body. The HTTP response may appear as follows.

```
HTTP/1.1 200 OK
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
 "verificationResponse": {
 “verstat”: “TN-Validation-Passed”
 }
}
```

In this case, the Expected Reason header is None. The Reason header is not populated in this scenario.

Multiple Reason Values in Response

In the case of 3GPP, if the response from STI server includes multiple reason values, the SBC includes two reason values in the SIP Reason header:

- cause and text specific to shaken PASSporT verification

- cause and text specific to first div PASSporT verification

It is assumed that the STI-VS always returns a SHAKEN PASSporT verification on top of the JSON body, followed by any DIV PASSporT verification(s). For example, consider the scenario wherein there are multiple reasonCode and reasonText objects included in the JSON Body. In this case, the HTTP Response may appear as follows.

```
{"verificationResponse":
{"verifyResults": [
 {"verifyResult": {"status": "fail", "ppt": "shaken", "reasonCode": 438,
"reasonText": "Identity signature payload does not match the re-constructed
one", "passport": "eyJhbGciOiJFUzI1NiIsInBwdCI6InNoYWtlbiIsIn....."}},
{"verifyResult": {"status": "fail", "ppt": "div", "reasonCode": 438,
"reasonText": "Identity signature payload does not match the re-constructed
one", "passport": "eyJhbGciOiJFUzI1NiIsInBwdCI6uPuDqJ8ua..........."}}
 ],
"verstatValue": "TN-Validation-Failed"
 }
 }
```

In this case, the Expected Reason header is as follows.

```
Reason: SIP ; cause=438 ;text="Identity signature payload does not match the
re-constructed one"; cause=438 ;text=" Identity signature payload does not
match the re-constructed one"
```

CDR Compliance for Number Authentication Mechanism

Compliance changes targeting the Number Authentication Mechanism 2.0 operations standards by the FFT (French Telecom Federation) include data and behavior changes related to supporting CDRs. All of these changes require that you enable the **man-compliance** option. Some changes require that you enable the **man-compliance** option and the **reason-json-sip-translation** parameter in the **sti-config** . The changes implemented by this configuration apply to CDRs delivered within RADIUS, DIAMETER and Local CDR deployments.

With respect to data changes, this implementation introduces three new objects to the set of RADIUS VSAs and DIAMETER AVPs supporting STIR/SHAKEN. The system includes all of these objects within its processes for verification processes including ACA/ACR exchanges with STIR/SHAKEN VS servers and the resulting CDRs. Each of these objects are documented below.

There are also significant and varied CDR behavioral changes associated with this compliance. Each behavior change is explained below.

CDR Behavior Changes

As with the other behaviors associated with Number Authentication Mechanism 2.0 operations standards, you enable the **man-compliance** option and the **reason-json-sip-translation** parameter in the **sti-config** of your SBC configuration to enable these CDR behaviors. The changes implemented by this configuration apply to CDRs delivered within RADIUS, DIAMETER and local CSV deployments.

The CDR-related behavioral changes implemented by this compliance option are addressed individually below.

Stir-VS-Verstat Attribute

This compliance feature enhances the behavior for populating the Stir-VS-Verstat attribute based on STI-VS HTTP responses. These behaviors apply to RADIUS, DIAMETER and CSV CDR deployments, and are applicable to both ATIS and 3GPP SHAKEN passports.

Based on STI-VS HTTP response, the SBC populates the "Stir-VS-Verstat" field as follows:

- Use the full string returned by the STI-VS, including No-TN-Validation, TN-Validation-Passed, TN-Validation-Failed, or anything custom up to 30 bytes
- If there is no verstat returned by STI-VS (whatever the reason: timeout, unavailable, missing body, etc.), leave the Stir-VS-Verstat empty
- If there is no request sent to the STI-VS, leave the Sti-VS-Verstat empty

Stir-VS-Reason Attribute

This compliance feature enhances the behavior for populating the Stir-VS-Reason attribute based on STI-VS HTTP Responses. These behaviors apply to both RADIUS, DIAMETER and CSV CDR deployments, and are applicable to both ATIS and 3GPP SHAKEN passports.

Based on STI-VS HTTP response, the SBC populates the "Stir-VS-Reason" field as follows:

- If the HTTP response status is 200 OK, the SBC:
  - Populates the "Sti-VS-Reason" field with the concatenation of the "reasoncode" and "reasontext" obtained from the JSON body.
  - If there is no reason returned within the JSON body, sets the "Sti-VS-Reason" field to be empty.
- If the HTTP response status is 4xx/5xx, the SBC:
  - ATIS—If the HTTP response status falls within the 4xx or 5xx range (client or server errors), populates the "Sti-VS-Reason" field with the HTTP response code and its associated response phrase.
  - 3GPP—If the HTTP response status falls within the 4xx or 5xx range (client or server errors), populates the "Sti-VS-Reason" field with the HTTP response code and its associated response phrase.
- No Answer from STI-VS—If there is no response received from the STI-VS service, sets the "Sti-VS-Reason" field to be empty.
- No Request to STI-VS—If no request has been made to the STI-VS service, meaning there was no interaction with the service, sets the "Sti-VS-Reason" field to be empty.

Enhancement for the Stir-Verified-Request-Exception-Id and Stir-Signed-Request-Exception-Id Attributes

The Case list and Exception table for these attributes are the same, with equivalent functions using either service (SVC) or policy (POL) exceptions. These behaviors are applicable to both ATIS and 3GPP deployments and are applicable to RADIUS, Diameter and CSV CDRs.

When the SBC fails to verify a request, the system populates the "Stir-Verified-Request-Exception-Id" and "Stir-Signed-Request-Exception-Id" attributes with the SVC or POL Exception IDs that are hard coded values mapped to each case. Based on event case, the SBC:

- For 4xx/5xx responses returned by STI-VS, populate with "SVC" or "POL" exceptions
- In case of a timeout, populate with "sti server timeout"

- When the STI-VS response is meaningless (JSON missing or malformed), populate with "invalid sti response"
- If the SBC does not send the request because the Identity header is missing or empty, populate with "Identity missing"

> **Note:**
>
> The case where the SBC does not send the request because the Identity header is missing or empty does not apply to the "Stir-Signed-Request-Exception-Id" attribute.

- If the SBC doesn't send the request due to the absence of a valid caller TN (From or PAI), populate with "TN missing"
- If the SBC refrains from sending the request due to STI admission control constraints, such as max-burst-rate, max-sustain-rate, burst-rate-window, or sustain-rate-window exceeded, populate with "sti constraints exceeded"

> **Note:**
>
> If no server is available from the sti-server-group, the system checks the STI admission control constraints of the last selected STI server.

- When the SBC is unable to send the request to the STI server (e.g., due to overloaded SIPD threads or overloaded CURLD), populate the field with "internal client error"
- If the SBC cannot send the request because the STI server is unreachable (circuit-breaker = open): populate with "sti server unreachable"
  - If no server in a **sti-server-group** is available, the system checks for the server unreachable status of the last selected STI server, and marks the CDR with the exception ID mapped to "sti server unreachable"
  - If you configured the server with an FQDN and the DNS IP list is empty, the system marks the CDR with the exception ID mapped to "sti server unreachable"
  - If the server is in Half Open State, the system does not send a request for that server. For this case also, the system marks the CDR with the exception ID mapped to "sti server unreachable"

The following table maps exception ID information based on the reason for the exception.

Exception Reason	Exception ID	Exception Text	Note
If SBC is not sending the request due to Identity header missing or empty	SVC4501	"Identity missing"	Input is missing hence this is case of service exception
If the STI server answer is meaningless (JSON missing or malformed)	SVC4502	“invalid sti response”	Invalid response, hence this is case of service exception
timeout	SVC4504	“sti server timeout”	Timeout is service exception
If SBC not sending the request due to no valid TN	SVC4505	"TN missing"	Input missing is case of service exception

If SBC failing to send the request to STI server (e.g. sipd threads overloaded or curld overloaded)	POL5500	"internal client error"	The request failed due to internal error is policy exception
If the SBC is not sending the request due to sti-server unreachable (circuit-breaker = open)	POL5503	"sti server unreachable"	sti-server unreachable is policy exception
If the SBC is not sending the request due to STI admission control (max-burst-rate or max-sustain-rate or burst-rate-window or sustain-rate-window reached)	POL5506	"sti constraints exceeded"	Constraint exceeded is policy exception
If the SBC bypasses the STIR request and the configured header finds in the incoming INVITE	SVC4506	“sti server bypass"	Bypassing the STIR request, hence it is case of service exception

Additional AS case considerations, when configured to 3GPP mode, include:

- Both SHAKEN and DIV servers are running and sending the invalid response.
  In this case, the system fills the "Stir-Signed-Request-Exception-Id” CDR field with SVC4502, which signifies an “invalid sti response”
- When the SHAKEN server is up and the DIV server is down:
  - If the response received from the SHAKEN server is invalid response, the system fills the "Stir-Signed-Request-Exception-Id” CDR field with SVC4502, which signifies an “invalid sti response”
  - If the response received from the SHAKEN server is valid, the system fills the "Stir-Signed-Request-Exception-Id” CDR field with SVC4504, which signifies a “sti server timeout”
- When the SHAKEN server is down and the DIV server is up:
  - If the response from the DIV server is invalid, the system fills the "Stir-Signed-Request-Exception-Id” CDR field with SVC4502, which signifies an “invalid sti response”
  - If the response from the DIV server is valid, the system fills the "Stir-Signed-Request-Exception-Id” CDR field with SVC4504, which signifies a “sti server timeout”
- When both the SHAKEN and DIV servers are down, the system fills the "Stir-Signed-Request-Exception-Id” CDR field with SVC4504, which signifies a “sti server timeout”

Additional CDR Data

When you enable the **man-compliance** option and the **reason-json-sip-translation** parameter, the SBC can populate three additional data objects within its validation process for 3GPP deployments. The system uses these objects to collect verification detail, from which it can determine how to populate CDRs. The changes implemented by this configuration apply to all CDR types.

The SBC packs these objects in RADIUS, DIAMETER and Local CSV CDRs in compliance with the MAN 2.0 requirements. When you enable **cdr-output-inclusive** and the man-compliance option, the system populates CDRs in this section with the applicable data.

- Stir-VS-Attest—Populate the attest claim value from the STI-VS verification response within the JSON body for 3GPP deployments. If there is no attest claim in provided by the verification response, the system populates the Sti-VS-Attest field with empty values in all CDRs.
- Stir-Div-VS-Verstat—This object collects the values, enclosed in double quotes, for each verified div PASSporT, separated by commas. In addition:
  - If both verstatValue and divResult verstatValue are present:
    - Populate verstatValue with “stir-vs-verstat”
    - Populate divResult verstatValue with “stir-div-vs-verstat”
  - If only verstatValue is present with no divResult, but ppt=div is there:
    - Populate verstatValue with “stir-vs-verstat”
    - Populate verstatValue with “stir-div-vs-verstat”
  - If a verstatValue is present with no divResult, and ppt=div is not there, populate verstatValue with “stir-vs-verstat” only.
- Stir-Div-VS-Reason—This object collects the reason value with the concatenation of the "reasoncode" and "reasontext" obtained from each Div Verify result of JSON body. The system collects the reasoncode and reasontext for Div Verify result, even if they are same. The SBC also maintains its existing behavior when reasoncode or reasontext is missing in the JSON body. (Compare the current implementation of the Sti-VS-Reason CDR attribute with ATIS mode).

RADIUS Objects

The table below specifies additional Oracle VSAs used for RADIUS verification in compliance with the MAN 2.0 requirements. When you enable **cdr-output-inclusive** and the man-compliance option, the system populates CDRs in this section with the applicable data.

VSA Name	Code	Type	Values
Stir-VS-Attest	178	String	A or B or C
Stir-Div-VS-Reason	179	String	Examples: SIP ; cause=438 ;text="Identitysignaturepayloaddoesnotmatchthere-constructedone", ;cause=410 ;text="Application inactive""
Stir-Div-VS-Verstat	180	String	Examples "TN-Validation-Passed, TN-Validation-Failed, TN-Validation-Passed"

The following table contains additional VSAs applicable to MAN 2.0 compliance that may be present in the generated ACR message.

RADIUS Attribute	Start	Interim-Update	Stop	Message Type = INVITE	Attribute value Type
Stir-VS-Attest	Yes	Yes	Yes	Yes	String
Stir-Div-VS-Reason	Yes	Yes	Yes	Yes	String

RADIUS Attribute	Start	Interim-Update	Stop	Message Type = INVITE	Attribute value Type
Stir-Div-VS-Verstat	Yes	Yes	Yes	Yes	String

Object Location in Local CDRs when protocol is RADIUS (or protocol=default)

When you enable **cdr-output-inclusive** and MAN 2.0 compliance, the system populates the following attributes in CDRs in your RADIUS CSV files.

STIR/SHAKEN attributes in CSV file for Call when the generate-event = START

CSV Placement	Attribute Name	
148	Stir-VS-Attest	ACME
149	Stir-Div-VS-Reason	ACME
150	Stir-Div-VS-Verstat	ACME

STIR/SHAKEN attributes in CSV file for Call when the generate-event = INTERIM

CSV Placement	Attribute Name	
194	Stir-VS-Attest	ACME
195	Stir-Div-VS-Reason	ACME
196	Stir-Div-VS-Verstat	ACME

STIR/SHAKEN attributes in CSV file for Call when the generate-event = STOP

CSV Placement	Attribute Name	
217	Stir-VS-Attest	ACME
218	Stir-Div-VS-Reason	ACME
219	Stir-Div-VS-Verstat	ACME

DIAMETER Objects

The table below specifies additional Oracle AVPs used for DIAMETER verification in compliance with the MAN 2.0 requirements.

AVP Name	Code	Is grouped?	Type	Values
Stir-VS-Attest	119	N	String	A or B or C
Stir-Div-VS-Reason	120	N	String	Examples: SIP ; cause=438 ;text="Identitysignaturepayloaddoesnotmatchthere-constructedone", ;cause=410 ;text="Application inactive"
Stir-Div-VS-Verstat	121	N	String	Examples "TN-Validation-Passed, TN-Validation-Failed, TN-Validation-Passed"

Object Location in Local CDRs when protocol is DIAMETER

When you enable **cdr-output-inclusive** and MAN 2.0 compliance, the system includes the following attributes in CDRs in your Diameter CSV files.

STIR/SHAKEN attributes in CSV file for Call when the generate-event = START

CSV Placement	Attribute Name	
176	Stir-VS-Attest	ACME
177	Stir-Div-VS-Reason	ACME
178	Stir-Div-VS-Verstat	ACME

STIR/SHAKEN attributes in CSV file for Call when the generate-event = INTERIM

CSV Placement	Attribute Name	
219	Stir-VS-Attest	ACME
220	Stir-Div-VS-Reason	ACME
221	Stir-Div-VS-Verstat	ACME

STIR/SHAKEN attributes in CSV file for Call when the generate-event = STOP

CSV Placement	Attribute Name	
243	Stir-VS-Attest	ACME
244	Stir-Div-VS-Reason	ACME
245	Stir-Div-VS-Verstat	ACME

STIR/SHAKEN Servers

Server Names as FQDNs

You can configure the SBC to use FQDNs for STI-AS and STI-VS server to establish STIR/SHAKEN server pools using DNS.

The primary objective for establishing server pools for STI servers using FQDNs is to load balance the servers, assuming they resolve to more than one STI server address. When the SBC cycles through an FQDN-resolved pool, it attempts to reach each of the servers in the list using round robin. The system tracks these servers' availability using its circuit breaker configuration and the DNS TTL, the latter of which confirms each IP address' availability upon expiry.

Your configured FQDN may reside within a **sti-group**, or be present as one of the four sti-servers configured on your source object (session-agent, sip-interface or realm). The system performs its resolution process on configuration load or TTL expiry. Each time the system picks an FQDN, the system refers to its resolution list and uses round-robin to select the next available IP address.

You setup the SBC to use DNS services for STIR/SHAKEN deployments by configuring the "root" configurations in the **sti-server** using an FQDN as prefix instead of an IP address:

- **as-server-root**—Applies to both modes
- **vs-server-root**—Applies to both modes
- **div-as-server-root**—Applies to 3GPP mode

- **div-vs-server-root**—Applies to 3GPP mode

When configured as an FQDN, a root setting syntax specifying all detail looks like the following:

```
ORACLE (sti-server)#as-server-root MyAsServer1/StirShakenWeb/resources/v1/
signing
```

> **Note:**
>
> If you do not specify a port for root configurations, the SBC uses port 80 for HTTP and port 443 for HTTPS

When the SBC uses FQDNs, it queries DNS for an A record (or AAAA record if you are using IPv6) to resolve/update the FQDNs on system bootup, configuration load, configuration activation, and TTL expiration. Upon resolution, the SBC:

- Enforces the applicable constraints, including realm max-burst-rate and max-sustain-rate, for each **sti-server** regardless of the number of IP addresses resolved
- Uses the round-robin strategy to load balance requests to each **sti-server**
- Establishes and maintains a **circuit-breaker** for each resolved address
- Uses TTL to trigger new queries to DNS to resolve/update the FQDNs

> **Note:**
>
> An FQDN entry operates the same whether it resolves to a single or multiple IP addresses.

Monitoring FQDN-Resolved Server Status

The SBC maintains a circuit breaker for each of the IP address returned by the DNS Resolver as separate entities, creating their circuit breakers when the address gets used. If the system chooses an address that already has a circuit breaker associated, it re-uses that circuit breaker. Each circuit breaker monitors the status of the corresponding server instance and triggers the system to raise major and minor traps and alarms. The correlation of server object status and separate IP address availability depends on the number of addresses resolved to each server object:

- If there is a single address associated with a server—The system generates a major alarm and trap when this **circuit-breaker** goes open (address unavailable).
- If there are multiple addresses associated with a server—The system generates a minor alarm and trap when one **circuit-breaker** goes open (address unavailable), and there is at least one other **circuit-breaker** closed (address available).
  Whenever every **circuit-breaker** associated with a resolved FQDN is open, the system generates a major alarm and trap.

Server status also depends on the **role** configuration within your **sti-server**. You can configure a server to be an AS server, a VS server or both (default). Oracle recommends you configure your DIV root with the same IP or FQDN prefix used for the AS or VS root on the same device. If not:

- If the configured **role** is **STI-AS**, and the **as-server-root** and **div-as-server-root** are configured with different IPs, then the system marks the **sti-server** as down if any one of the IP's configured against as-server-root or div-as-server-root is down.
- If the configured **role** is **STI-VS**, and the **vs-server-root** and **div-vs-server-root** are configured with different IPs, then the system marks the **sti-server** as down if any one of the IP's configured against vs-server-root or div-vs-server-root is down.
- If the configured **role** is **both**, and all root configurations have different IPs, the behavior includes:
  - If any of the IPs configured against **as-server-root** and **div-as-server-root** are down, the system marks the authentication service as down.
  - **sti-server vs-server-root** and **div-vs-server-root** are down, the system marks the verification service as down.

Load Balancing STIR/SHAKEN Servers

Load balancing STIR/SHAKEN servers allows you to take better advantage of server pools, establishing preferences for given servers based on their deployments.

You can configure the **sti-as** and **sti-vs** in the **session-agent**, **sip-interface** and **realm-config** with an **sti-server-group** or a list of **sti-server** names (up to 4). If you configure the **sti-server-group**, the selection of a server to send the AS/VS requests to is based on the strategy parameter configured in the **sti-server-group**. If you have configured multiple entries for **sti-server**, the SBC treats the list as a group with round robin selection strategy.

Strategies include:

- **Hunt**—Selects servers in the order in which they are listed. For example, if the first server is online, working, and has not exceeded defined constraints, all traffic is sent to the first server. If the first server is offline or if it exceeds a defined constraint, the second server is selected. If the first and second server s are offline or exceed defined constraints, the third server is selected. And so on through the list of servers.
- **RoundRobin**—Selects each server in the order in which they are listed in the destination list, selecting each server in turn, one per transaction.
- **LeastBusy**—Selects the server that has the fewest number of transactions. If multiple servers have the same number of transactions, the foremost server in the group will be used.
- **PropDist**—Proportionally distributes the transactions among all of the available servers.
- **LowSusRate**—The Lowest Sustained Rate strategy routes to the server with the lowest sustained rate of transaction (based on observed sustained rate).

You can load limit servers in a group by configuring **max-burst-rate**, **max-sustain-rate**, **burst-rate-window** and **sustain-rate-window** in the **sti-server** configuration. When the rate of AS/VS requests sent to a STIR server exceeds either the **max-burst-rate** or **max-sustain-rate**, the SBC does not select the server, reducing its ovehead. If **max-burst-rate**, **max-sustain-rate**, **burst-rate-window** and **sustain-rate-window** are not configured (with their values equal to 0) there is no load limiting.

STI Server Heartbeat

You can configure the SBC to extend upon its STIR server availability functions using the **sti-heartbeat-config** element. This heartbeat feature allows the system to check the state of the server without using live signaling traffic. This feature works for both ATIS and 3GPP deployments and is disabled by default.

A STIR server supports multiple addresses, and establishes circuit breaker state for each one. This is also true for server groups. The SBC refers to the circuit breaker state to determine when to use this heartbeat function. When you enable this function, either globally or on a given server, the SBC performs this heartbeat check for all circuit breakers in either the HALF_OPEN or OPEN state.

When a STIR server is in OPEN state, the SBC does not send any live traffic to it, preferring to send that traffic to a server known to be available. For servers with connectivity issues, you can use this heartbeat function to test availability and bring that server back into the service rotation faster. When a connection responds to this heartbeat test, the SBC marks that circuit breaker as CLOSED and starts using that server in its rotation.

This heartbeat function sends JSON requests to applicable servers made up of dummy data. These requests use the correct format for authentication and verification requests. This allows the test to verify service availability in addition to connectivity without using live call data. If the server sends a response, the SBC recognizes its dummy data and the source of the response. The SBC recognizes the response is to the heartbeat feature and, therefore, makes the server available for service requests for real calls.

Additional operational detail includes:

- The SBC does not apply this **sti-heartbeat-config** function to any server that is currently in the closed state.
- For ATIS deployments, if all IP addresses for an AS root on a STIR server are down, meaning each circuit breaker is in the OPEN state, the SBC does not perform authentication service with that server. The means the SBC does not send an authentication request for any INVITE, instead sending only heartbeat requests to all of that server's IP addresses that have their circuit breaker state as OPEN or HALF_OPEN. Within this context, the SBC sends heartbeat requests periodically. If it receives a heartbeat response from any STIR server address, it changes the circuit breaker state of that IP from OPEN to CLOSED and performs authentication service with that server.
- For 3GPP deployments, the SBC uses the same behavior as the bullet above, referring to the circuit breaker state of both AS and DIV AS root addresses.

  > **Note:**
  >
  > The applicable configuration fields are **as-server-root** and **div-as-server-root**.

- For ATIS and 3GPP deployments, the SBC uses the same behavior as the two bullets above for verification services.

  > **Note:**
  >
  > The applicable configuration fields are **vs-server-root** and **div-vs-server-root**.

- When accessing service using a **sti-server-group**, the SBC uses the behavior described in the three bullets above based on the circuit breaker status for all addresses in that **sti-server-group**.

You can configure an individual **sti-server** to ignore the **sti-heartbeat-profile** by setting the **sti-heartbeat-state** option under that **sti-server** to disabled. This **sti-heartbeat-state** option under **sti-server** takes precedence over the **sti-heartbeat-config** setting.

```
ORACLE(sti-server)#options +sti-heartbeat-state=disabled
```

By default, this option is not set for any STI server. You can however, enable this option, resulting in the system applying the feature to this STI server even if you have left the **sti-heartbeat-config** disabled. This allows you to apply the feature to a single or a small group of servers only. In this case, the transaction would include the default or configured values of the other **sti-heartbeat-config** parameters. Conversely, if you set this feature to disabled and enable the **sti-heartbeat-config**, the system applies the feature to all servers except those with the option disabled. This flexibility allows you to easily apply the feature to a majority or a minority of your servers.

This feature works for all STI servers whether configured as single **stir-server** elements or configured within a **sti-server-group**.

Configuration

The **sti-heartbeat-config** includes a parameter to enable the function, a timing parameter that specifies how often it tries to reach each server, and parameters to specify dummy orig, dest and div TNs for inclusion in the request. Specific profile parameters include:

- **sti-heartbeat-state**—Enables/disables the heartbeat functionality for all STIR/SHAKEN servers at the system level.
- **sti-heartbeat-msg-interval-time**—Specifies, in seconds, the time-interval to send the heartbeat message to STIR/SHAKEN servers. The default is 5, and the range is 0 - 3600 seconds.
- **sti-orig-tn-number**—Specifies the calling party TN number when sending heartbeat messages to STIR/SHAKEN servers. The value is a string, and the default is 9999999999.
- **sti-dest-tn-number**—Specifies the called party TN number when sending heartbeat messages to STIR/SHAKEN servers. The value is a string, and the default is 7777777777.
- **sti-div-tn-number**—Specifies the called party diverted TN number when sending heartbeat messages to STIR/SHAKEN servers. The value is a string, and the default is 3333333333.

Reporting on Heartbeat Statistics

The **show stir** command supports the **heartbeat** argument to display heartbeat feature statistics. Without further argument, the command displays statistics on all STI servers. You can further refine this command with specific server names to narrow the command output to your specific server.

```
ORACLE#show stir heartbeat MyStiServer
06:36:02-44
Server: MyStiServer
STIR/SHAKEN Heartbeat Statistics        ------------LIFETIME----------
                                          Recent     Total      Permax
HB Request sent to AS                       0          0          0
HB Response recevieved from AS              0          0          0
HB Request sent to VS                       0          29         6
HB Response received from AS                0          3          1
```

Explanations for each heartbeat output line includes:

- HB Request sent to AS—Displays the count of STIR AS/Div-AS heartbeat request sent to check the server availability.
- HB Response received from AS—Displays the count of STIR AS/Div-AS heartbeat response received from STIR server.

- HB Request sent to VS—Displays the count of STIR VS/Div-VS heartbeat request sent to check the server availability.
- HB Response received from VS—Displays the count of STIR VS/Div-VS heartbeat response received from STIR server.

In addition, you can use the **reset** command to set all statistic values to zero, again for all servers or for your specific server.

Applicable ACLI syntax for these commands includes:

- **show stir heartbeat all**
- **show stir heartbeat -all**
- **show stir heartbeat**
- **reset stir heartbeat <sti-server-name>**

Example STI Server Heartbeat Requests

This section provides examples of ATIS and 3GPP STI server heartbeat requests.

ATIS Requests

For SHAKEN authentication heartbeat messages, the system gets the values for the 'orig TN' and dest 'TN taken' from your **sti-heartbeat-config** configuration. The system gets the values for the 'origid' and 'attest' from your **sti-server** configuration.

The following is an example ATIS authentication heartbeat message.

```
POST /StirShakenWeb/resources/v1/signing HTTP/1.1
Host: stirshaken.oracle.com
Content-Type: application/json
X-RequestID: 4c7a2de8-1d8f-4b56-4ef5-0585fee32d15
Accept: application/json
Content-Length: 169

{"signingRequest":{"iat":1683545181,"attest":"B","ppt":"shaken","dest":{"tn":
["7777777777"]},"orig":
{"tn":"9999999999"},"origid":"00000000-0000-0000-0000-0000000f0abc"}}
```

For SHAKEN verification heartbeat messages, the system gets the values for the 'from TN' and the 'to TN' taken from your **sti-heartbeat-config** configuration, and uses the current time.

The following is an example ATIS verification heartbeat message.

```
POST /StirShakenWeb/resources/v1/verification HTTP/1.1
Host: stirshaken.oracle.com
Content-Type: application/json
X-RequestID: 67188cde-f7ea-434e-55dd-53b57af06333
Accept: application/json
Content-Length: 97

{"verificationRequest":{"time":1683538513,"to":{"tn":["7777777777"]},"from":
{"tn":"9999999999"}}}
```

3GPP Requests

For SHAKEN and DIV authentication heartbeat messages, the system gets the values for the 'orig TN' and dest 'TN taken' from your **sti-heartbeat-config** configuration. The system gets the values for the 'origid' and 'attest' from your **sti-server** configuration. The 'iat' parameter is the current time.

The following is an example 3GPP SHAKEN Authentication heartbeat message.

```
POST /StirShakenWeb/resources/v1/signing HTTP/1.1
Host: stirshaken.oracle.com
Accept: application/json
Content-Type: application/json
Content-Length: 169

{"signingRequest":{"iat":1683544351,"attest":"B","ppt":"shaken","dest":{"tn":
["7777777777"]},"orig":
{"tn":"9999999999"},"origid":"00000000-0000-0000-0000-0000000f0abc"}}
```

The following is an example 3GPP DIV Authentication heartbeat message.

```
POST /StirShakenWeb/resources/v1/divSigning HTTP/1.1
Host: stirshaken.oracle.com
Accept: application/json
Content-Type: application/json
Content-Length: 121

{"signingRequest":{"iat":1683544351,"ppt":"div","dest":{"tn":
["3333333333"]},"div":{"tn":"7777777777"},"orig":{"tn":"9999999999"}}}
```

For SHAKEN and DIV verification heartbeat messages, the system gets the values for the 'orig TN' and dest 'TN taken' from your **sti-heartbeat-config** configuration. The 'iat' parameter is the current time.

The following is an example 3GPP SHAKEN verification heartbeat message.

```
POST /StirShakenWeb/resources/v1/verification HTTP/1.1
Host: stirshaken.oracle.com
Accept: application/json
Content-Type: application/json
Content-Length: 95

{"verificationRequest":{"to":{"tn":"7777777777"},"time":1683544122,"from":
{"tn":"9999999999"}}}
```

The following is an example 3GPP DIV verification heartbeat message.

```
POST /StirShakenWeb/resources/v1/divVerification HTTP/1.1
Host: stirshaken.oracle.com
Accept: application/json
Content-Type: application/json
Content-Length: 148
```

{"verificationRequest":{"time":1683543455,"from":{"tn":"9999999999"},"dest":
{"tn":"3333333333"},"identityHeaders":["xyz"],"to":{"tn":"7777777777"}}}

Per Call Availability of STI Servers

In addition to circuit breakers, which prevent the SBC from continually trying to communicate with a non-responsive server, you can configure maximum retries and SIP transaction timeout responses that determine how the SBC behaves with individual calls when there are problems reaching STI servers. This feature is applicable for both authentication as well as verification.

STI Servers can become unreachable due to connection issues and problems, meaning they may not respond or generate timeouts. The SBC includes processes for determining how to handle unresponsive servers on an ongoing basis as well as during a call.

- With respect to the ongoing process, the system does not attempt to access any server with a circuit breaker in the Open state.
- During a call, the SBC creates a server list and attempts to access servers in that list based on either your **sti-server** or **sti-server-group** configuration. Having generated a list of servers for a call, the SBC iterates through the list each time an attempt to reach a server times out. This process, which you enable by setting a **max-retry-attempts** parameter to a non-zero value, is used if a request sent to a STIR server does not get a response or the response times out. If the system does not send the request itself successfully, the system does not perform any retry.

When a call triggers a **sti-server-group** or a single server configured with an FQDN, and you have configured the **max-retry-attempts** parameter in the **sti-config**, the SBC adds your number of retries to its criteria for accessing servers. Failure to reach a server after reaching the **max-retry-attempts** is one of the reasons the system may determine that the STIR/SHAKEN infrastructure is not able to authenticate or verify that call.

A **sti-server-group** is basically a list of **sti-server**s. Whether you have configured a **stir-server** with an IP address or an FQDN, the system cycles through the server list established by those configurations:

- When all servers in the group use an IP address, the system cycles though servers using the group's **strategy**.
- When all servers in the group use an FQDN, the system creates a list using the **strategy** and cycles through each resolved address using round robin before proceeding with the original list.
- When cycling through a **sti-server-group** that has both addresses and FQDNs, the system cycles through the group list using your configured strategy, and cycles through all resolved addresses using round-robin when it reaches an FQDN in the group list.

For a single server configured with an FQDN, the system cycles through all resolved addresses using round-robin.

You consider each server's **timeout** value in conjunction with your **max-retry-attempts** value when applying this feature to your environment. Each server's **timeout** value determines when the system tries the next server in the list for individual calls. The **max-retry-attempts** defines the total number of servers the system attempts to reach for a call.

As a simple calculation for determining retry count, consider that a standard SIP transaction timeout is 32 seconds and you have configured the **timeout** for the applicable **sti-server** to its minimum of 100 milliseconds. This results in 10 retries in 1 second, and 30 retries in 3 seconds. Applicable values for **max-retry-attempts** are zero, which disables this feature, to 30, which would accommodate the scenario above.

Ultimately, the SBC stops trying to access servers recursively for this call when:

- The SIP transaction timeout expires.
- The system reaches your **max-retry-attempts** value.
- The system has tried to reach all the servers in the applicable **sti-server-group**. This remains true even if the SIP transaction timeout has not expired and the process has not reached your **max-retry-attempts** count.

The action the system takes after it stops trying to access servers depends on the reason for stopping:

- For verification processes, when process reaches your **max-retry-attempts** count, the system proceeds with the call without verification, adding the “verstat=No-TN-Validation” parameter and continuing with the SIP transaction.
- If the SIP transaction has timed out, SIP logic ends the call. The system performs these processes for each call, regardless of whether a previous call was not able to reach the STI server.

The SBC honors standard SIP operation and any other configuration you establish at any point within these recursions:

- If you have configured any of your **stir-servers** with verstat=No-TValidation-Timeout in its **sti-response-treatment-entry** and that server times out, the system rejects the call without any further retries.
- The SBC stops recursion if any server responds with a failure or exception. These responses are considered valid, ending the retry logic.
- The system rejects calls if the results of the overall call process meets the criteria for rejection.
- The **max-retry-attempts** feature interacts with the **sti-response-treatment-config** feature:
  - If max-retry-attempts = 0 (recursion disabled), then call rejection shall apply as soon as sti-vs answer
  - If max-retry-attempts > 0 (recursion enabled), then call rejection shall apply based on the verstat value configured for call rejection for each server. If you have configured any **stir-server** with verstat=No-TN-Validation-Timeout in the **sti-response-treatment-entry** and a timeout occurs for that server, the SBC rejects the call and does not make any further retry attempts.
  - If you have not configured any applicable **stir-server** with verstat = No-TN-Validation-Timeout in **sti-response-treatment-entry**, the SBC retries all the servers until it reaches either the **max-retry-attempt** count or a SIP transaction timeout occurs.
  - If you have configured the server with a **sti-response-treatment-entry** that has a verstat value other than No-TN-Validation-Timeout, and **max-retry-attempts** is greater than 0, the system applies call rejection logic based on the last attempted server that responds.

Consider the following example, wherein you have two servers configured in a **sti-server-group** using the round robin strategy. In addition, you have configured your **max-retry-attempts** to 2. These servers include:

- server1 – stirDemo1.tryfqdn.com:9001, resolving to two addresses, 10.199.240.155 and 10.198.240.157
- server2 – stirDemo2.tryfqdn.com:9002, resolving to two addresses, 10.146.134.180 and 10.196.136.185

> **Note:**
>
> Each Curl POST request includes the following:
>
> ```
> "{"verificationRequest":{"identity":"..rq3pjT1hoRwakEGjHCnWSwUnshd0-
> zJ6F1VOgFWSjHBr8Qj","time":1672646120,"to":{"tn":["1a23"]},"from":
> {"tn":"123"}}}"
> ```

The process proceeds with the SBC performing the following steps:

1. Resolves the FQDN for server1 and selects 10.199.240.155 using round robin on the DNS resolutions.
2. Sets its retry count to zero.
3. POSTs a Curl POST request to server1, address 10.199.240.155:
4. Times out this request, having not received a response from 10.199.240.155.
5. Initiates its first retry by selecting 10.198.240.157 from the list of untried IPs for server1.
6. Sets its retry count to one.
7. POSTs a Curl POST request to server1, address 10.199.240.157.
8. Times out this request, having not received a response from 10.199.240.157.
9. Resolves the FQDN for server2 and selects 10.146.134.180 using round robin on the DNS resolutions.
10. Initiates its second retry by selecting 10.146.134.180 from the list of untried IPs for server2.
11. Sets its retry count to two.
12. POSTs a Curl POST request to server2, address 10.146.134.180.
13. Times out this request, having not received a response from 10.146.134.180.
14. Terminates the retry process. The **max-retry-attempt** is now equal to the retry count.
15. Continues with the SIP Transaction.

Reporting

This feature affects the **show stir** command output by including the retries made to a server in its query count. When examining this output, you need to consider these retries:

- If you are not using this feature, you can expect one SIP INVITE to generate a single query to the STI servers.
- If you are using this feature and have set **max-retry-attempts** to three, an unsuccessful attempt to reach a single server would record four queries (1 request + 3 retries).

Applicable commands include:

- show stir stats
- show stir agents all
- show stir agents <identifier>
- show stir interface all
- show stir interface <identifier>

- show stir realm all
- show stir realm <identifier>
- show stir all

With respect to this feature, the system does not include a specific field for displaying retry count. Instead, you can derive retries from the commands above by referring to the STI-AS and STI-VS Queries, Success Responses, and Unsuccessful Responses rows. As such, the number of STI-AS or STI-VS queries is based on the total number of Retry to sti-servers and the number of response counts, accordingly.

Supporting HA with STIR SHAKEN over TCP

You can configure the SBC with the **exclusive-http-client-port-range** option within the **system-config** to support an HA Pair running STIR SHAKEN to use the different set of ports between Primary and Secondary machine for establishing TCP connection with HTTP server.

The SBC allocates connection ports using this feature when you have configured the option prior to operation. If you have an HA deployment that is operating with STIR SHAKEN over TCP that does not use this feature, you can use one of the following procedures to implement the feature.

The procedures are dependent on the current state of your deployment. The key differentiator with these procedures is whether the current Active machine is the Primary HA member or the Secondary HA member. For both conditions, an additional consideration is whether you are implementing this feature during an upgrade or to an HA deployment that already supports the **exclusive-http-client-port-range** option.

After all of these procedures, the SBC allocates a separate set of TCP ports for the Primary and Secondary machines to use for connections to STIR/SHAKEN HTTP clients.

The Current Active is also the Primary

Use these procedures if the current Active machine is the Primary and the current Standby machine is the Secondary:

- This first procedure applies to HA deployments that you are upgrading to a version that supports this feature:
  1. Enable the **exclusive-http-client-port-range** option within the **system-config** on the active SBC. HA processes synchronize this configuration with the standby node.
  2. Load the new software version onto the standby node.
  3. Reboot the standby node.
  4. Execute a manual switchover (**notify berpd force**) from the current active. At this point, the former active becomes the new standby.
  5. Load the new software version onto the new standby node.
  6. Reboot the new standby node.
- This second procedure applies if you have already upgraded your HA deployments to a version that supports this feature:
  1. Enable the **exclusive-http-client-port-range** option within the **system-config** on the active SBC. HA processes synchronize this configuration with the standby node.
  2. Reboot the standby node.
  3. Execute a manual switchover (**notify berpd force**) from the current active to the standby.

4. Reboot the new standby node.

The Current Active is also the Secondary

Use these procedures if the current Active machine is the Secondary and the current Standby machine is the Primary:

- This first procedure applies to HA deployments that you are upgrading to a version that supports this feature:
  1. Enable the **exclusive-http-client-port-range** option within the **system-config** on the Active SBC [Secondary]. HA processes synchronize this configuration with the standby machine [Primary].
  2. Load the new software version onto the standby machine [Primary].
  3. Reboot the standby machine.
  4. Execute a manual switchover (**notify berpd force**) from the current Active. At this point, the former active becomes the new standby.
  5. Load the new software version onto the new standby node.
  6. Reboot the new standby node.
  7. Execute a manual switchover (**notify berpd force**) from the current Active to the standby.
- This second procedure applies if you have already upgraded your HA deployments to a version that supports this feature:
  1. Enable the **exclusive-http-client-port-range** option within the **system-config** on the active SBC. HA processes synchronize this configuration with the standby node.
  2. Reboot the standby node.
  3. Execute a manual switchover (**notify berpd force**) from the current active to the standby.
  4. Reboot the new standby node.
  5. Execute a manual switchover (**notify berpd force**) from the current Active to the standby.

Configuring the STIR/SHAKEN Client

STIR/SHAKEN Client configuration procedures include:

- Configure your global **sti-config**
- Configure **sti-server** elements, specifying server operating mode and authentication methodology
- Create **sti-group** elements (if load balancing is desired)
- Configure STIR/SHAKEN parameters to **session-agent**, **sip-interface** and/or **realm-config**. These configurations use the standard precedence of agent, interface, realm to determine which configuration applies to traffic, and supersede the values you may have set on the server itself.

Configure the Authentication Profile

You configure Authentication Profile elements on the SBC for use, for example, within your REST configuration context. The client configuration specifies servers to which your realm can make HTTP requests.

Procedure

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)#
```

2. Type **security**and press Enter.

```
ACMEPACKET(configure)# security
ACMEPACKET(security)#
```

3. Type **auth-profile**and press Enter.

```
ACMEPACKET(security)# auth-profile
ACMEPACKET(auth-profile)#
```

4. **name**—Enter a name (unique identifier) for the server. Valid values are alpha-numeric characters. Default is blank.
5. **authentication-scheme**—Set the authentication scheme. The default is **Bearer**.
6. **preshared-key**—Set the encrypted password for this profile.
7. Type **done**.
8. Save and activate your configuration.

Configure the HTTP Client

You configure HTTP Client elements on the SBC for use, for example, within your REST configuration context. The client configuration specifies the interface and IP address from which the SBC makes HTTP requests.

Before you begin:

- If needed, create the TLS profile that you want to use for your HTTP client. See TLS Configuration Process.

> **Note:**
>
> Although the SBC acts as TLS client in this implementation, your TLS configuration must include generation of the SBC certificate and CSR. (The end entity certificates parameter under the **tls-profile** is mandatory .)

- Create the authentication profile that you want to use for your HTTP client.

Procedure

1. Navigate to the **http-client** element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# http-client
ORACLE(http-client)#
```

2. **name**—Enter a name (unique identifier) for the client. Valid values are alpha-numeric characters. Default is blank.
3. **state**—Set to enable. The default is enable.
4. **realm**—Set the name of the realm on which to send requests.

   You can have your **http-client** operate on wancom0 by leaving this **realm** field empty.
5. **ip-address**—Set the IP address of the HTTP client.

   This address must be an operational HIP address configured on the target **network-interface** in the applicable realm.
6. **tls-profile**—Set the TLS profile that want this HTTP client to use.
7. **auth-profile**—Set the name of the authentication profile that you want the HTTP client to use.
8. Type **done**.
9. Save and activate your configuration.

Configure the STI Config

To configure global STIR/SHAKEN configuration parameters on the SBC, use the **sti-config** object under **session-router**.

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)#
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)#
```

3. Type **sti-config**and press Enter.

```
ACMEPACKET(session-router)# sti-config
ACMEPACKET(sti-config)#
```

4. **circuit-breaker-window-duration**—Specify the time in seconds the system uses to establish the window it uses to establish the circuit breakers timing. The default is 10 seconds. The range is from 10 to 30.
5. **circuit-breaker-error-threshold** —Specify the number of errors the system counts before it marks the server as out of service. The default is 5 seconds. The range is from 3 to 10.

> **Note:**
>
> The number of unsuccessful statistics, displayed as STI-VS Unsuccessful Responses, can be greater than this configured error threshold. These unsuccessful stats are a cumulative count, not a count of the errors received in this window duration.

6. **circuit-breaker-retry-time**—Specify the time in seconds the system uses to retry connecting to the server. The default is 15 seconds. The range is from 5 to 900.
7. **circuit-breaker-half-open-frequency**—Specify the number of times the system skips this server while it is marked half open. The default is 6, which causes the system to re-use the server once every 6th retry. The range is from 5 to 100.
8. **sti-signaling-attest-info-mandatory**—Enables the system to require that the received INVITE contain the P-Attestation-Info and/or Attestation-Info header, and the P-Origination-Id and/or Origination-Id header, for the system to send a signing request to STI-AS. When disabled, the system sends a signing request to the STI-AS using either your configured attestation and orig-id values or, if **sti-signalling-attest** is **enabled**, using the information from the relevant SIP headers.
   - Disabled (Default)
   - Enabled
9. **anonymous-uri-add-verstat-to-hostpart**—Enables the system to place the verstat parameter after the hostpart when the received INVITE does not contain a P-Asserted-Identity header, but does contain a Privacy header and an anonymous URI in the FROM. When enabled, the system adds the verstat parameter after the hostpart of the anonymous From URI. When disabled, the system adds the verstat parameter after the user-part of the anonymous From URI.
   - Disabled (Default)
   - Enabled
10. **use-identity-header**—Enable, in conjunction with STI verification, to add a Reason header to 18x, 19x responses and 3xx, 4xx, 5xx, 6xx final responses to a callee with a cause value of “428” and the text “Use Identity Header” for all received INVITEs that did not contain an identity header.
11. **check-duplicate-passports**—Enable the system to check for duplicate SHAKEN or DIV passports in a received INVITE. If it finds duplicates, the system deletes one of the duplicates from the INVITE.
12. **TN-retargeting**—Enables to perform DIV authentication request, based on the received INVITE.
13. **verstat-comparison**—Specify how the system compares the verstat value present in FROM and PAI headers with the values present in this parameter. If a value matches, then the system accepts the validation and performs only DIV authentication processes. If the value is empty, the system does not perform the comparison.
   - Default: Empty
   - TN-Validation-Passed
   - No-TN-Validation
   - TN-Validation-Passed
   - TN-Validation-Failed

14. **dest-comparison**—Specify whether and on which header the system compares its stored TN with either the Request-URI or the To header in received INVITEs. If the value is empty, the system does not perform the comparison
    - Default: Empty
    - Request-URI
    - To
15. **sti-as-correlation-id** —Enables the system to add the SipCallId parameter to REST authentication requests to the STI-AS. This parameter contains the information from the corrID parameter in the P-NokiaSiemens.Session-Info SIP header.
16. **sti-header-mapping-ruleset-name**—Specifies the name of this STI Header Mapping Ruleset you want to use as default across all sti-servers. A ruleset name configured against a sti-server takes precedence for that server over this ruleset.
17. **reason-json-sip-translation** —Enables the system to create a Reason header from the parameters reasoncode and reasontext, if received from the STI-VS. The system also adds this Reason header to the egress INVITE.
18. **flip-tn-lookup-order**—Specifies whether the system applies precedence to PAI or FROM headers within incoming out-of-dialog requests when determining how to populate the orig shaken passport claim and to populate the verstat parameter received from STI-VS in outgoing requests.
    - Disabled (Default)—Apply precedence to PAI headers
    - Enabled—Apply precedence to FROM headers
19. Type **done**.
20. Save and activate your configuration.

Configure the STI Servers

To configure an STI server on the SBC, use the **sti-server** object under **session-router**. The SBC creates a UUID4 for each sti server you configure. Use the following procedure to configure a server on the SBC.

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)#
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)#
```

3. Type **sti-server**and press Enter.

```
ACMEPACKET(session-router)# sti-server
ACMEPACKET(sti-server)#
```

4. **name**—Enter the name (unique identifier) of the server. Valid values are alpha-numeric characters. Default is blank.
5. **description**—Enter a description of the server. Valid values are alpha-numeric characters. Default is blank.

6. **role**—Specify whether the system performs authentication, verification or both functions with this server.
   - Default: Both
   - STI-AS
   - STI-VS
7. **http-rest-type**—Specify the operating mode the system uses with this sti-server.
   - Default: ATIS
   - 3GPP
8. **as-server-root**—Enter the IP Address or FQDN of the STIR verification server, with optional port and base path.
9. **vs-server-root**—Enter the IP Address or FQDN of the STIR authentication server, with optional port and base path.
10. **div-as-server-root**—Specify the server that handles DIV authentication requests. Values include the IP Address or FQDN of the STIR DIV authentication server with optional port and base path.
11. **div-vs-server-root**—Specify the server that handles DIV verification requests. Values include the IP Address or FQDN of the STIR DIV verification server with optional port and base path.
12. **orig-id**—Specify a UUID v4 other than the one automatically created by the SBC. When configured, the SBC uses this value within STI-AS requests.
    - Empty (default)
    - Type in the desired origination-ID
    - ""—The system generates an origination-ID
13. **sti-attest**—Select an attestation value that is sent in AS request. You configure attestation using the ATIS-defined grading for attestation efficacy.
    - A—(Full-attestation)
    - B—(Partial-attestation)
    - C—(Gateway-attestation)
14. **timeout**— Enter the amount of time (in milliseconds) to wait for a REST response from a STIR server. The valid range is 100-30000.
15. **http-client**—Enter the name of a http-client configuration that has information for making connection to the sti-server. This name can be up to 128 characters and must not begin with a period (.) or a dash (-) character.
16. **max-burst-rate**—Enter the maximum sending burst rate for STIR requests (per second). The valid range is 0-999999999.
17. **max-sustain-rate**—Enter the maximum sending sustained rate for STIR requests (per second). The valid range is 0-999999999.
18. **burst-rate-window**—Enter the time period (in seconds) used to calculate burst rate. The valid range is 0-999999999.
19. **sustain-rate-window**—Enter the time period (in seconds) used to calculate sustain rate. The valid range is 0-999999999.
20. **sti-header-mapping-ruleset-name**—Specify the name of the mapping ruleset you created for HTTP to SIP INVITE header mapping to perform between this server and the system.

21. **options**—Configure your applicable options to this sti-server.
22. Type **done**.
23. Save and activate your configuration.

Configure STI Server Groups

To configure an STI server on the SBC, use the **sti-server-group** object under **session-router**. Use the following procedure to configure a server on the SBC.

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)#
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)#
```

3. Type **sti-server-group**and press Enter.

```
ACMEPACKET(session-router)# sti-server-group
ACMEPACKET(sti-server-group)#
```

4. **name**—Enter the name (unique identifier) of the HTTP proxy. Valid values are alpha-numeric characters. Default is blank.
5. **description**—Enter a description of the HTTP proxy. Valid values are alpha-numeric characters. Default is blank.
6. **strategy**—Select a strategy for sti-server selection.
   - hunt
   - round-robin
   - least-busy
   - prop-dist
   - low-sus-rate
7. **sti-servers**—Enter the sti-server names that are members of the group. The server names can be specified as a space-separated list enclosed in double quotes. Alternatively, individual sti-server name can be added or removed by prefix the name with a plus (+) or minus (-) character, respectively.
8. Type **done**.
9. Save and activate your configuration.

Configure STIR/SHAKEN REST Client Functionality on a Session Agent

To configure an STI server on the SBC, use the **sti-server** object under session-router. Use the following procedure to configure a server on the SBC.

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)#
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)#
```

3. Type **session-agent**and press Enter.

```
ACMEPACKET(session-router)# session-agent
ACMEPACKET(session-agent)#
```

4. **sti-as**—Enter the name of a sti-server-group name or a space-separated list of sti-servers (up to four allowed) to which the SBC shall send AS requests.

   When configuring a group-name, use the syntax below, which uses the prefix `stg:` followed by your group name. In this example, `myStiGroupName` is your server group name.

```
ACMEPACKET(session-router)# sti-as stg:myStiGroupNsme
ACMEPACKET(session-agent)#
```

5. **sti-vs**—Enter the name of a sti-server-group name or a space-separated list of sti-servers (up to four allowed) to which the SBC shall send VS requests.

   When configuring a group-name, use the syntax below, which uses the prefix `stg:` followed by your group name. In this example, `myStiGroupName` is your server group name.

```
ACMEPACKET(session-router)# sti-vs stg:myStiGroupNsme
ACMEPACKET(session-agent)#
```

6. **sti-orig-id**—Enter a UUID v4 to be added to STI-AS requests, if not already present.
7. **sti-attest**—Select an attestation value that is sent in AS request.
   - full-attestation
   - partial-attestation
   - gateway-attestation
8. **sti-signaling-attest**—Select whether to use attestation value and origId from SIP headers, when present.
   - enabled
   - disabled
9. Type **done**.
10. Save and activate your configuration.

<Enter Concept Title Here>

(Required) <Enter a short description here.>

(Required) <Enter introductory text here, including the definition and purpose of the concept.>

Call Rejection Enhancement

You can configure the SBC can provide more detailed reasons for rejecting a call towards the caller. You configure these reason-code and reason-text entries within your **sti-response-treatment-config** configurations. This feature applies to the messages sent by the SBC to the caller after it has completed its STI verification process.

By default, the SBC sends generalized reason detail to applicable stations for rejecting a call based on an STI-VS response. You can understand these responses as normalized verstats, presenting a standardized string for both ATIS and 3GPP deployments with the applicable SIP messages. As a result, the verstat information provided by the SBC may not be as specific as desired about the reason a call failed.

For example, without this feature the SBC tags the following call failure scenarios with the same verstat, No-TN-Validation:

- STI-VS no request
- STI-VS no answer
- Answer with no verstat

The SBC, however, allows you to configure your **sti-response-treatment-config** to more provide more specific reason information for specific call fail scenarios, including:

- No-TN-Validation-Timeout—STI-VS does not answer
- No-TN-Validation-StiInvalid—STI-VS answer is meaningless (JSON missing or malformed)
- No-TN-Validation-IdentityMissing—SIP Identity is missing (no request sent to STI-VS)
- No-TN-Validation-TNMissing—TN is missing (no request sent to STI-VS)
- No-TN-Validation-StiConstraints—STI constraints are exceeded (no request sent to STI-VS)
- No-TN-Validation-Unreachable—STI-VS circuit-breaker is open (no request sent to STI-VS)
- No-TN-Validation-ClientError—SBC internal failure (no request sent to STI-VS)

No-TN-Validation-Timeout—STI-VS does not answer

When you configure a **sti-response-treatment-entry** with a verstat of **No-TN-Validation-Timeout** and a **reason-code** that is “empty" or "0”, the SBC rejects applicable calls when there is a response timeout on the STI-VS server. However, the SBC adds "No-TN-Validation" (default) in the FROM.

When these requests to STI-VS servers time out, the SBC uses the verstat key “No-TN-Validation-Timeout” to find matches for rejecting these calls.

This first configuration example demonstrates the SBC providing a less accurate response.

```
ORACLE(sti-response-treatment-entry)show
verstat No-TN-Validation-Timeout
reason-code
role STI-VS
sip-reason-code
sip-reason-text
```

In this case, the SBC rejects the call with:

- A “403 Forbidden” Response
- STI-VS-REASON "" (Field left empty)
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—Empty - No reason header included

This second configuration example demonstrates the SBC providing a more accurate response. Here, the system adds "NoValidationTimeout" (your configuration) in the FROM.

```
ORACLE(sti-response-treatment-entry)show
verstat No-TN-Validation-Timeout
reason-code
role STI-VS
sip-reason-code 404
sip-reason-text
sip-reason-text NoValidationTimeout
```

In this case, the SBC rejects the call with:

- “404 NoValidationTimeout” Response
- STI-VS-REASON empty
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—Empty - No reason header included

No-TN-Validation-StiInvalid/No-TN-Validation-MalformedStiResponse—STI-VS answer is meaningless (JSON missing or malformed)

When you configure a **sti-response-treatment-entry** with a verstat of **No-TN-Validation-StiInvalid** and a **reason-code** that is “empty" or "0”, the SBC rejects applicable calls when there is a malformed response or missing JSON body from STI-VS server.

Again however, the SBC adds "No-TN-Validation" (default) in the FROM.

This first configuration example demonstrates the SBC providing a less accurate response.

```
ORACLE(sti-response-treatment-entry)show
sti-response-treatment-entry
verstat No-TN-Validation-MalformedStiResponse
reason-code
role STI-VS
sip-reason-code
sip-reason-text
```

In this case, the SBC rejects the call with:

- A “403 Forbidden” Response
- STI-VS-REASON, composed of the HTTP status code plus status text
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden

  - 3GPP SIP Reason header—SIP;cause=HTTP status code;text=HTTP status phrase

Unlike the other cases in this section, you enable the **man-compliance** option in the **sti-config** to establish this more accurate behavior.

This second configuration example demonstrates the SBC providing a more accurate response. Here, the system adds "No-TN-Validation" in the FROM.

```
(sti-response-treatment-entry)show
sti-response-treatment-entry
verstat No-TN-Validation- MalformedStiResponse
reason-code
role STI-VS
sip-reason-code 404
sip-reason-text NoValidationStiInvalid
```

In this case, the SBC rejects the call with:

- A “404 NoValidationStiInvalid” Response
- STI-VS-REASON includes http reason code + reason text
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—SIP;cause=403;text="4xx/5xx"

No-TN-Validation-IdentityMissing—SIP Identity is missing (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-IdentityMissing
sip-reason-text NoValidationIdentityMissing
```

Here, the system provides a “404 NoValidationIdentityMissing” Response, and a adds "No-TN-Validation" in the FROM.

No-TN-Validation-TNMissing—TN is missing (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-TNMissing
sip-reason-text NoValidationTNMissing
```

Here, the system provides a “404 NoValidationTNMissing” Response, and a adds "No-TN-Validation" in the FROM.

No-TN-Validation-StiConstraints—STI constraints are exceeded (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-StiConstraints
sip-reason-text NoValidationStiConstraints
```

Here, the system provides a “404 NoValidationStiConstraints” Response, and adds "No-TN-Validation".

No-TN-Validation-Unreachable—STI-VS circuit-breaker is open (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-Unreachable
sip-reason-text NoValidationStiUnreachable
```

Here, the system provides a “404 NoValidationStiConstraints” Response, and adds "No-TN-Validation".

No-TN-Validation-ClientError—SBC internal failure (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-ClientError
sip-reason-text NoValidationClientError
```

Here, the system provides a “404 NoValidationStiConstraints” Response, and adds "No-TN-Validation".

Changing the Precedence for Handling orig and verstat Values

By default, the SBC uses incoming PAI headers as a vehicle to retrieve information, create an orig value and to convey verstat values. If PAI headers are not present or sufficient, the system uses incoming FROM headers for these purposes. Some regions, however, require that the FROM header be the first source of this information. To accommodate these deployments, you can configure the SBC to use the FROM as the primary caller id source for information used to determine a SHAKEN orig claim and the verstat value. This behavior is applicable to both ATIS and 3GPP based implementations.

The SBC refers to the URIs in PAI and FROM headers for the presence of telephone numbers (TNs) as identifying information during authentication and verification procedures:

- To support authentication, when the SBC receives an out-of-dialog INVITE with no identification header or verstat value, it searches for an appropriate TN in the request and, if it finds one, creates an 'orig' key and initiates a signing request.

- To support verification procedures, meaning the initial request did have an identity header, the SBC :
  - Searches for an appropriate TN in the request and, if it finds one, creates an 'orig' key and initiates a verification request.
  - Obtains verification and populates the header from which it selects TNs with verstat values indicating verification success or failure.

The presence of multiple TNs as well as URIs that are not TNs in the initial request adds some complexity with respect to the use of the information and the population of the verstat result. In addition to changing the preferred source of TNs, this feature also results in the TN URI and SIP URI parsing and selection behaviors covered below.

You can enable this functionality on a global basis using the **flip-tn-lookup-order** parameter within the **sti-config**. The default is disabled.

```
ORACLE(sti-config)#flip-tn-lookup-order enabled
```

If preferred, you can enable this functionality on a per-server basis by enabling the **flip-tn-lookup-order** value as an option within the **sti-server**.

```
ORACLE(sti-server)#options +flip-tn-lookup-order=enabled
```

The **sti-server** option setting, when enabled, takes precedence over the global configuration. This allows you to, for example, use the default behavior across the device, while preferring the FROM header for the STI servers that you configure with the option.

When enabled, the SBC changes behaviors to the following:

- Prioritizes the selection of URIs in FROM headers to populate the “orig” shaken passport claim.
- Prioritizes the selection of URIs in FROM headers for population of the “verstat” received from Verification Server.
- Applies precedence to any **sti-server** option setting over the global setting.

When you enable the feature, detailed SBC authentication behavior includes checking the URI in the FROM and PAI headers for the presence of a TNs and creating an orig key using that TN. Regardless of the TN selection, the SBC always performs the signing request, and continues to process the call:

- If a TN is present in the FROM, the SBC uses that TN to create the "orig" key.
- If there is no TN in the FROM, and there is a tel PAI, the SBC uses the tel PAI to create the "orig" key.
- If there is no TN in the FROM, there is no tel PAI and there is a SIP PAI, the SBC uses the SIP PAI to create the "orig" key.
- If there is no TN in any header, the SBC does not send a validation request to the STI servers and sets the verstat to No-TN-Validation in the FROM and PAI headers.

Detailed SBC verification behavior also begins with checks for the presence of TNs. Regardless of the TN selection, the SBC performs the same procedure for selecting a TN and creating an "orig" key it performs for authentication, initiates the verification request, and continues processing the call. Results of the TN check includes:

- If there is a TN in the FROM, the SBC applies the verstat to the FROM header.

- If there is a TN in the FROM, and there is a PAI header with a TN (either SIP or Tel PAI), where the values of the TNs match, the SBC adds the verstat parameter to both the From and PAI headers.
- If there is not a TN in the From, and there is a PAI header with a TN (either SIP or Tel PAI), the SBC adds the verstat parameter to the PAI header.

Additional verstat population behavior when there are multiple PAI headers includes:

- If there is a TN in the From, and two PAI headers:
  - Tel URI and SIP URI with matching TNs, the system adds the verstat to all headers.
  - One PAI matches and the other does not, the system adds the verstat to the From and the matching PAI.
  - Neither PAI has a TN, the system adds the verstat to the From only.
- If there is not a TN in the From, but there are Tel and SIP PAIs that have TNs, the system adds the verstat to the Tel PAI.
  The system also compares the Tel PAI TN with the SIP PAI. If the TNs match, the system also adds the verstat to the SIP PAI.
- If there is a From Header with no TN, and the PAI headers also have no TNs, the system adds the verstat to all headers.

Creating a Reason Header During Verification

You can configure the SBC to create and insert SIP reason headers into applicable SIP INVITEs based on information received from an STI-VS during verification attempts. These headers provide insight into the reason the STI-VS could not or did not verify the request. You can use this feature to provide visibility into the reasoncode, reasontext and the verstat parameters downstream within the SIP INVITE and in CDRs. This feature applies to both ATIS and 3GPP modes.

For each verification scenario, if a server response indicates a failure or no validation with reasoncode and/or reasontext in the JSON body, the SBC looks up the associated **sti-server** for the presence of the string **reason-json-sip-translation enabled** in the options parameter. If enabled, the SBC creates a Reason header using the values received in the JSON claims, reasoncode and reasontext and adds the reason header to the egress INVITE before forwarding it to the next hop. The syntax follows RFC 3326.

This setting is also available as a parameter in the **sti-config**. If the **sti-server** parameter is not set, the SBC refers to the global parameter setting. The **sti-server** setting takes precedence.

If the INVITE already includes a reason header, the SBC appends the header with the parameters it retrieves from the STI-VS to the original reason parameters, as shown in the example below:

```
Reason: SIP ;cause=438 ;text="Invalid Identity
Header" ;cause=403 ;text="Stale Date"
```

If the STI-VS is unavailable due to timeout or because it is unreachable, the SBC does not provide static mapping between reason related json claims (reasoncode and reasontext) and the reason header. Although it does not add a new Reason header, it does, per existing behavior, add a verstat with No-TN-Validation to the From/PAI header(s).

To enable this functionality on a per-server basis, you set the **reason-json-sip-translation** value as an parameter within a **sti-server** element, as shown below. You should consider the

sti-server as having three settings, including enabled, disabled and empty. The default is empty.

```
ORACLE(sti-server)#options +reason-json-sip-translation=enabled
```

To enable this functionality on a global basis, you use the **reason-json-sip-translation** parameter within the **sti-config**, as shown below. The default is disabled.

```
ORACLE(sti-config)#reason-json-sip-translation enabled
```

The SBC only refers to the global parameter in the **sti-config** if the **sti-server** parameter is empty. This allows you to, for example, enable the **sti-config** parameter, and enable the feature across the SBC by leaving **sti-server** parameters empty. You could then explicitly disable the feature on a per-server basis by setting the server parameter to disabled.

```
ORACLE(sti-server)#options +reason-json-sip-translation=disabled
```

Additional feature behavior to consider includes:

- If you have enabled this feature, but the JSON body of the response from STI-VS does not contain reasoncode and reasontext claims, the SBC does not add a reason header to the egress INVITE.
- The ATIS specification defines JSON claims, reasontext and reasoncode as optional. If there is only a reasoncode or reasontext in the response from STI-VS and you have configured this feature, the system adds the reason object presented and does not address the missing object.
  For example, if there is no reasoncode but there is reasontext, the translation may be:

  ```
  Reason: SIP ;text="Stale Date"
  ```

- If there is a failure or timeout at the STI-VS, the SBC performs no action on the claims and adds the No-TN-Validation verstat, per standard behavior.
- The 3GPP specification does not include JSON claims reasoncode and reasontext. Therefore, the SBC creates the Reason header from the Reason Phrase and HTTP return code.
  For example, consider the sample HTTP Response Received below.

  ```
  HTTP/1.0 400 Missing request body
  Date: Tue, 28 Apr 2020 02:14:29 GMT
  Content-Type: application/json
  ```

  The SBC adds the below to the egress INVITE.

  ```
  Reason: SIP ;cause=400 ;text="Missing request body"
  ```

  Furthermore, if there is no reason phrase in the response, the SBC adds the following Reason header without text to the egress INVITE.

  ```
  Reason: SIP ;cause=400
  ```

- For 3GPP deployments, the SBC appends new data to an existing SIP Reason header at the end of the existing line after a semi-colon.

For example, consider the sample SIP header below as present.

```
Reason: SIP ;cause=580 ;text="Precondition Failure"
```

The SBC appends that header with the new cause and text as shown below.

```
Reason: SIP ;cause=580 ;text="Precondition Failure";
cause=400 ;text="Missing request body"
```

> **Note:**
>
> The SBC also does this when adding a Reason header to responses, including 2xx and 5xx.

Enhanced Reason Header Configuration

The Number Authentication Mechanism specification requires additional reason header capabilities that give the provider more specific reasons for call handling. Within the context of this specification, the key SBC function is to better identify call verification issues, allowing the system to conform with the specification's requirements for handling and rejecting SPAM calls. This feature applies to the information provided in a reason header within an egress INVITE towards the callee.

You use the **sti-reason-header-config** element to create multiple reason header configurations that define how the SBC behaves when confronted with specific verification issue scenarios.

Each instance of **sti-reason-header-config** includes a **name** along with the **sti-reason-header-entries** parameter, which provides access to detailed reason header configuration. You enter a **name** for your element to apply to a **sti-server** or, globally, to the **sti-config**. When both elements have a **sti-reason-header-config**, the **sti-server** configuration takes precedence.

An example of a configured **sti-reason-header-config** is provided below. This configuration has two reason header entries, which define reason header behavior that is in addition to your other configured reason header behavior.

```
ORACLE(sti-reason-header-config) # show
sti-reason-header-config
name reason-header-scenario-list
         sti-reason-header-entry
                  scenario              sti-server-
timeout
                  cause-code            690
                  reason-text           STI Server Timeout
         sti-reason-header-entry
                  scenario              invalid-sti-
response
                  cause-code            691
                  reason-text           Invalid STI Response
```

Under **sti-reason-header-entry**, there are three parameters:

- **scenario** (Required)—You configure the **scenario** parameter using the scenario list below. The default setting is **sti-server-timeout**. Settings include:
  - **sti-server-timeout**—The system recognizes this timeout scenario when it does not receive a reply from the STI server.
    Oracle recommends you configure this scenario as sti-server-timeout, with a cause-code (690) and reason-text STI Server Timeout.
  - **invalid-sti-response**—The system recognizes this scenario when it receives an STI-VS answer that it cannot derive. Applicable scenarios include the JSON being missing or malformed.
    Oracle recommends you configure this scenario as invalid-sti-response, with a cause-code to 691 and reason-text Invalid STI Response. If you leave the cause-code empty, the system populates its message with 691. If you leave the reason-text empty, the system populates its message with Invalid STI Response.

    Oracle considers a verification response as an invalid STI response based on the following:
    - 200 OK case
      - From an ATIS perspective, meaningless answer:
        - no JSON body
        - JSON body but no verificationResponse
        - JSON body and verificationResponse but no verstat
      - From a 3GPP perspective, meaningless answer:
        - no JSON body
        - JSON body but no verificationResponse
        - if SHAKEN verification: JSON body and verificationResponse but no verstatValue
        - if DIV verification:
        - JSON body and verificationResponse but no divResult
        - JSON body and verificationResponse and divResult but no verstatValue
    - 4xx/5xx case—Within 4xx/5xx responses from the STI-VS, there is no "invalid sti response" possible. The SBC, therefore, populates the SIP Reason Header as follows:
      - cause:
        - reasonCode if included in JSON body
        - HTTP response status-code if reasonCode is not available in JSON body
      - text:
        - reasonText if included in JSON body
        - OR error included in JSON body if reasonText is not available in JSON body
        - OR HTTP reason phrase if reasonText nor error is available in JSON body
  - **use-identity-header**—The system recognizes this scenario when it receives an INVITE with the Identity header missing or empty. You configure this scenario as use-identity-header, with a cause-code of 428 and reason-text of Use Identity Header. If you leave the cause-code empty, the system populates its messages with 428. You

can leave the reason-text empty or configured. If left empty, the system populates the field with Use Identity Header.

- **tn-missing**—The system recognizes this scenario when it receives an INVITE that does not include any telephone number in the From or the PAI.
  Oracle recommends you configure this scenario as tn-missing, with a cause-code of 692 and reason text of TN missing. If you leave the cause-code empty, the system populates the field with 692. You can also leave the reason-text empty or configured. If you leave the cause-code empty, the system populates the field with Invalid STI Response.
- **sti-constraints-exceeded**—The system recognizes this scenario when it does not send the request because traffic has exceeded a configured STI admission control parameter, including **max-burst-rate**, **max-sustain-rate**, **burst-rate-window** or **sustain-rate-window**.
  Oracle recommends you configure this scenario as **sti-constraints-exceeded**, with a cause-code of 693 and a reason text of STI Constraints Exceeded. If you leave the cause-code empty, the system populates the field with 693. You can also leave the reason-text left empty or configured. If left empty, the system populates the text with STI Constraints Exceeded.

  > **Note:**
  >
  > In case of SAG recursion, meaning you have enabled the **retry** parameter in the **sti-config** and applied on **sti-server-group**, and none of the servers are available, meaning all the servers in the group have exceeded constraints, the system populates the SIP reason header per the Oracle recommendation.

- **sti-server-unreachable**—The system recognizes this scenario when the **sti-server** is unreachable, meaning the applicable circuit-breaker is either open or half-open. In these scenarios, the SBC does not send a request.
  Oracle recommends you configure this scenario as sti-server-unreachable, with a cause-code as 694 and reason-text as STI Server Unreachable. If you leave the cause-code empty, the system populates the field with 694. You can also leave the reason-text left empty or configured. If left empty, the system populates the text with Sti Server Unreachable.
- **internal-client-error**—The system recognizes this scenario when the SBC fails to send the request to the STI server.
  Oracle recommends you configure this scenario as internal-client-error, with a cause-code of 695 and reason-text of Internal Client Error. If you leave the cause-code empty, the system populates the field with 695. You can also leave the reason-text left empty or configured. If left empty, the system populates the text with Internal Client Error.
- **sti-server-bypass**—The system recognizes this scenario when it has bypass a STIR/SHAKEN request. You typically configure the exotic code is to 696 when you configure the scenario to 'sti server bypass'. Configure scenario as “sti-server-bypass” and cause-code as 696. If you leave the cause-code parameter empty, the system populates the cause code with 696 for this scenario. You can also leave the reason-text empty or configured. If left empty, the system populates the applicable fields with “Sti Server Bypass”.

- **cause-code** (Optional)—As described in the scenarios, you can set this to any value within the range 400-699.
- **reason-text** (Optional)—The system has this reason text field empty by default for all the scenarios. The system maps actual text, however, internally mapped to respective strings

based on scenario. As described in the scenarios, you can configure preferred text to overwrite the defaults.

The **sti-reason-header-entry** element is a multi-instance element, allowing you to create an entry for every scenario needed.

Including CALEA in Authentication Requests

You can configure the SBC to include Communications Assistance for Law Enforcement Act (CALEA) information in SHAKEN and DIV PASSporT authentication requests. This feature applies to both ATIS and 3GPP operation modes. Some environments provide CALEA information within the correlation ID (corrID) string and convey it within the P-NokiaSiemens.Session-Info SIP header in an INVITE. You use this feature is to provide corrID source information to the STI-AS. When configured, the SBC extracts this string and includes it in the proprietary SipCallId header within the proprietary JSON SipCallId parameter of a REST request for use by the STI-AS authentication process.

Upon configuration, the SBC monitors INVITEs for the presence of the SIP P-NokiaSiemens.Session-Info header. An INVITE can contain only one P-NokiaSiemens.Session-Info header. Upon detection, and assuming the scenario generated an authentication request, the SBC forwards that information to the STI-AS. The SBC does not process CALEA information for any of its own signaling purposes.

If the received INVITE already has a SHAKEN or DIV PASSporT, or you have not configured the egress for STIR/SHAKEN, the SBC does not make an authentication request to the STI-AS and, therefore, does not forward CALEA information. Furthermore, the SBC does not perform this function if it finds the P-NokiaSiemens.Session-Info header does not include a corrID parameter, of if the corrID is empty.

> **Note:**
>
> The text corrID is case sensitive.

The P-NokiaSiemens.Session-Info header may include multiple parameters in addition to a corrID, separated by semi-colons. If it contains multiple corrID parameters, the SBC uses the first corrID listed in the header for this function.

The SBC only forwards a maximum of 256 characters from a corrID. If a corrID length exceed 256 characters, the SBC truncates it so that it does not exceed 256 characters. The SBC does this by deleting all parameters and their values that cause the SipCallId to exceed 256 characters. In practice, this results in the SBC normally forwarding corrIDs that are less than 256 characters.

When enabled, the SBC inserts the corrID into a proprietary JSON parameter named “SipCallId”, and includes this parameter in standard authentication requests. This parameter does not conflict with other standard authentication parameters. An example of a SipCallId follows:

```
“SipCallId”:“LU-1583529412591367-group0.example.com”
```

You enable this functionality using the **sti-as-correlation-id** in the **sti-config**:

```
ORACLE(sti-config)# sti-as-correlation-id enabled
```

Rejecting Calls During Verification

You can configure the SBC to reject calls based on the verstat and reason code from sti-vs server. To perform this function, you configure a **sti-response-treatment-entry** element and apply it to the **sti-config** or **sti-server**. The SBC uses each entry to determine whether to reject any call that matches the entry with the verstat and/or reasoncode in an STI-VS response. This function applies to both ATIS and 3GPP deployments.

When you have configured the SBC for STI treatment and it receives an INVITE that results in an STI-VS response that includes specific verstat and/or reason code information, the SBC can reject the call. The SBC compares the verstat and, depending on the message, the HTTP reason-code provided by the STI server, to establish a match with your configuration and determine whether or not to reject the call. This rejection consists of sending a SIP message to notify the caller of the reason for rejecting the call, using your mapped sip-reason-code and sip-reason-text. Having sent this message, the SBC then drops the call.

To configure this function, you create one or more **sti-response-treatment-config** elements under **session-router**. Each **sti-response-treatment-config** has a **name** and one or more **sti-response-treatment-entries** parameter(s). The **sti-response-treatment-entries** is a multi-instance sub-element that includes the details for rejecting a call.

Parameters include:

- **verstat**—This mandatory parameter specifies the actionable verstat value returned in the STI-VS response. Values include TN-Validation-Passed, TN-Validation-Failed, No-TN-Validation, or custom, which you define.
- **reason-code**—Specifies the reason code as returned in the STI-VS response. Values include 403, 428, 436, 437, 438, or a custom value you specify within the range of 1xx to 5xx.
  You leave this parameter empty if you want to reject the call based on the verstat only.
- **role**—Specifies whether this rule applies to verification (STI-VS) or authentication (STI-AS).

  > **Note:**
  >
  > The only supported value in the S-Cz9.3.0 software version is STI-VS.

- **sip-reason-code**—Specifies the SIP reason code the system uses to generate the SIP response. Values include 403, 428, 436, 437, 438, or a custom value you specify within the range of 4xx to 6xx.
  You can leave this parameter empty if you want to use the reasoncode provided by the STI-VS in the final SIP response. If the parameter is empty and the STI-VS does not return a reasoncode, the system uses the default value of 403.
- **sip-reason-text**—SIP reason text the system uses to generate the sip response. Values include Forbidden (default), or a custom string.
  You can leave this parameter empty if you want to use the reasontext provided by the STI-VS in the final SIP response. If the STI-VS does not return a reasontext, the system uses the default value of Forbidden.

To complete your configuration, you enter the **name** of a **sti-response-treatment-config** to the **sti-response-treatment-config-name** parameter within a **sti-server** or the **sti-config**. The **sti-server** configuration takes precedence over the **sti-config** configuration.

The SBC compares the input from the STI server with your **sti-response-treatment-entries** to determine a match. The system looks to match your entries with both the verstat and reason-code. But endstations are not consistent with the call information that they provide. For this reason, you may decide to create multiple entries with the same verstat, but variations in the reason-code. Furthermore, if you do not configure a reason-code to an entry, you create a single entry that matches the verstat and any reason-code.

As an operational example, consider an STI-VS response that matches to the following **sti-response-treatment-config** entry. (The received verstat is TN-Validation-Failed and the reasoncode is 437).

```
sti-response-treatment-entry
verstat             TN-Validation-Failed
reason-code         437
role                STI-VS
sip-reason-code     403
sip-reason-text     Forbidden
```

In this case, the SBC matches any STI-VS reply with both verstat and reason-code, replies to the caller with a SIP message mapping the strings “403" and "Forbidden” into the response, and drops the call.

Applicable 3GPP versus ATIS deployment behaviors include:

- For 3GPP deployments, the SBC checks all the verstat results in the response, including SHAKEN verstat and DIV verstat values to find a match and reject the call based on first best match policy.
- For 3GPP deployments, if the reply does not have a SHAKEN or DIV verstat in the response, the SBC takes the verstat as “No-TN-Validation-None" to find a match with configured entries for call rejection purpose.
- For ATIS deployments, if the reply does not have a verstat in the response, the SBC takes the verstat as “No-TN-Validation-None" to find a match with configured entries for call rejection purpose.
- If the SBC fails to send a request to an sti-vs server due to internal error, it uses the verstat “No-TN-Validation-Timeout” to find a match for call rejection. To support this, configure an entry that uses this verstat and no reason-code, sip-reason-code or sip-reason-text.
  - For ATIS deployments, the SBC populates the case SIP reason header using SIP;cause=403;text=Forbidden.
  - For 3GPP deployments, the SBC does not populate the SIP reason header.

Applicable deployment behavior that applies to both ATIS and 3GPP STIR/SHAKEN modes:

- In ATIS and 3GPP, if 4xx and 5xx responses do not have verstat values, the SBC uses the verstat value as "No-TN-Validation-None" for call rejection purpose. Configure an entry that matches “No-TN-Validation-None” if you want to reject these calls. For SIP rejection messages to these calls, the SBC adds the default verstat, "No-TN-Validation".
- When receiving 4xx or 5xx responses, because the reason-code is not in the response body, the SBC reads the reason-code from the HTTP status code in the FROM status line of the response.
- To accommodate timeout scenarios, you can create an entry wherein the verstat is "No-TN-Validation-Timeout" and the reason code is empty or zero as a match to reject those calls.

- If an STI-VS does not answer or answers without any verstat key in the JSON body, the SBC adds "No-TN-Validation" to the From and PAI header.

When rejecting a call based on this feature, the reason-header the SBC sends in the final response message back to the UAC is dependent on the STI operational mode of the SBC:

- ATIS:
  - In the case of a 200OK response:
    * If the body contains reasoncode and reasontext Reason header is filled with reasoncode + reason text
    * If reason code is not present, the system uses 403
    * If reason text is not present, the system uses forbidden.
- 3GPP—In the case of a 200 OK response, the system leaves the reason header empty.
- ATIS and 3GPP—In the case of 4xx or 5xx responses, the system reads the reason code and reason text from the status line and uses them in the reason header.

Table of Responses to Rejected Requests

The SBC follows the table below to determine what to include as reason-text when it rejects SIP requests within the context of STIR/SHAKEN authentication and verification.

sip-reason-code Configured	sip-reason-text Configured	reasoncode in STI-VS response	reasontext in STI-VS response	Final Sip Reasoncode/ Reasontext in Response
Yes	Yes	Yes/No	Yes/No	Configured sip-reason-code and / sip-reason-text
No	No	Yes	Yes	reasoncode in STI-VS response/ reasontext in STI-VS response
No	No	No	No	403 / Forbidden
Yes	No	Yes/No	Yes	sip-reason-code Configured / reasontext in STI-VS response
Yes	No	Yes/No	No	sip-reason-code Configured / Forbidden
No	Yes	No	Yes/No	403 / sip-reason-text Configured
No	Yes	Yes	Yes/No	reasoncode in STI-VS response/ Configured sip-reason-text

Reporting on Call Rejection

The SBC provides counters on rejected calls within ACLI commands, SNMP, HDR and CDR records.

- From the ACLI, you can find these statistics listed as 'STI-VS INVITEs Rejected' within the output of the server, interface, realm, agent, and system wide versions of the **show stir** command.

- From CDRs, you can find per-call indications of call rejection presented using the Stir-VS-Invite-State VSA for RADIUS and Stir-VS-Invite-State AVP for Diameter.
- From HDRs, you can find you can find these statistics listed as 'Stir-VS-Invite-State' within the output of the server, interface, realm, agent, and system wide HDR records.
- From SNMP, you can find these statistics listed as 'vsInviteRejected' within the output of the server, interface, realm, agent, and system wide tables.

Related Configuration

This rejection feature works in conjunction with the server retry feature implemented with your **max-retry-attempts** configuration.

- If **max-retry-attempts** is greater than zero, meaning recursion is disabled, the system rejects calls based on the sti-vs response from the first server that responds.
- If **max-retry-attempts** is greater than zero, meaning recursion is enabled, the system rejects calls based on the sti-vs response to the last server the system attempts to reach.

Creating Detailed Call Rejection Information

You can configure the SBC can provide more specific reasons for rejecting a call towards the caller. You configure these reason-code and reason-text entries within your **sti-response-treatment-config** configurations. This feature applies to the messages sent by the SBC to the caller after it has completed its STI verification process.

By default, the SBC sends generalized reason detail to applicable stations for rejecting a call based on an STI-VS response. These responses are normalized verstats, presenting a standardized string for both ATIS and 3GPP deployments within the applicable SIP messages. As a result, the verstat information provided by the SBC may not be as specific as desired about the reason a call failed.

For example, without this feature the SBC tags the following call failure scenarios with the same verstat, No-TN-Validation:

- STI-VS no request
- STI-VS no answer
- Answer with no verstat

The SBC, however, allows you to configure your **sti-response-treatment-config** to provide more specific reason information for specific call fail scenarios, including:

- No-TN-Validation-Timeout—STI-VS does not answer
- No-TN-Validation-StiInvalid—STI-VS answer is meaningless (JSON missing or malformed)
- No-TN-Validation-IdentityMissing—SIP Identity is missing (no request sent to STI-VS)
- No-TN-Validation-TNMissing—TN is missing (no request sent to STI-VS)
- No-TN-Validation-StiConstraints—STI constraints are exceeded (no request sent to STI-VS)
- No-TN-Validation-Unreachable—STI-VS circuit-breaker is open (no request sent to STI-VS)
- No-TN-Validation-ClientError—SBC internal failure (no request sent to STI-VS)
- No-TN-Validation-Bypass—The STIR requests that are bypassed

No-TN-Validation-Timeout—STI-VS does not answer

When you configure a **sti-response-treatment-entry** with a verstat of **No-TN-Validation-Timeout** and a **reason-code** that is “empty" or "0”, the SBC rejects applicable calls when there is a response timeout on the STI-VS server.

When these requests to STI-VS servers time out, the SBC uses the verstat key “No-TN-Validation-Timeout” to find matches for rejecting these calls.

This first configuration example demonstrates the SBC providing a less accurate response.

```
ORACLE(sti-response-treatment-entry)show
verstat No-TN-Validation-Timeout
reason-code
role STI-VS
sip-reason-code
sip-reason-text
```

In this case, the SBC rejects the call with:

- A “403 Forbidden” Response
- STI-VS-REASON "" (Field left empty)
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—Empty - No reason header included
- The verstat No-TN-Validation in the FROM header

This second configuration example demonstrates the SBC providing a more accurate response.

```
ORACLE(sti-response-treatment-entry)show
verstat No-TN-Validation-Timeout
reason-code
role STI-VS
sip-reason-code 404
sip-reason-text NoValidationTimeout
```

In this case, the SBC rejects the call with:

- “404 NoValidationTimeout” Response
- STI-VS-REASON empty
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—Empty - No reason header included
- The verstat No-TN-Validation in the FROM header

No-TN-Validation-StiInvalid/No-TN-Validation-MalformedStiResponse—STI-VS answer is meaningless (JSON missing or malformed)

When you configure a **sti-response-treatment-entry** with a verstat of **No-TN-Validation-StiInvalid** and a **reason-code** that is “empty" or "0”, the SBC rejects applicable calls when

there is a malformed response or missing JSON body from STI-VS server. The system matches the Verstat key “No-TN-Validation-StiInvalid” for this call rejection.

This first configuration example demonstrates the SBC providing a less accurate response.

```
ORACLE(sti-response-treatment-entry)show
sti-response-treatment-entry
verstat No-TN-Validation-MalformedStiResponse
reason-code
role STI-VS
sip-reason-code
sip-reason-text
```

In this case, the SBC rejects the call with:

- A “403 Forbidden” Response
- STI-VS-REASON, composed of the HTTP status code plus status text
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—SIP;cause=HTTP status code;text=HTTP status phrase
- The verstat No-TN-Validation in the FROM header

This second configuration example demonstrates the SBC providing a more accurate response.

```
(sti-response-treatment-entry)show
sti-response-treatment-entry
verstat No-TN-Validation- MalformedStiResponse
reason-code
role STI-VS
sip-reason-code 404
sip-reason-text NoValidationStiInvalid
```

In this case, the SBC rejects the call with:

- A “404 NoValidationStiInvalid” Response
- STI-VS-REASON includes http reason code + reason text
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—SIP;cause=403;text="4xx/5xx"
- The verstat No-TN-Validation in the FROM header

Oracle considers a verification response as an invalid STI response based on the following:

- 200 OK case
  - From an ATIS perspective, meaningless answer:
    - no JSON body
    - JSON body but no verificationResponse
    - JSON body and verificationResponse but no verstat
  - From a 3GPP perspective, meaningless answer:

        - no JSON body
        - JSON body but no verificationResponse
        - if SHAKEN verification: JSON body and verificationResponse but no verstatValue
        - if DIV verification:
        - JSON body and verificationResponse but no divResult
        - JSON body and verificationResponse and divResult but no verstatValue
- 4xx/5xx case—Within 4xx/5xx responses from the STI-VS, there is no "invalid sti response" possible. The SBC, therefore, populates the SIP Reason Header as follows:
  - cause:
    - reasonCode if included in JSON body
    - HTTP response status-code if reasonCode is not available in JSON body
  - text:
    - reasonText if included in JSON body
    - OR error included in JSON body if reasonText is not available in JSON body
    - OR HTTP reason phrase if reasonText nor error is available in JSON body

No-TN-Validation-IdentityMissing—SIP Identity is missing (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-IdentityMissing
sip-reason-text NoValidationIdentityMissing
```

Here, the system provides a “404 NoValidationIdentityMissing” Response, and a adds "No-TN-Validation" in the FROM.

No-TN-Validation-TNMissing—TN is missing (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-TNMissing
sip-reason-text NoValidationTNMissing
```

Here, the system provides a “404 NoValidationTNMissing” Response, and a adds "No-TN-Validation" in the FROM.

No-TN-Validation-StiConstraints—STI constraints are exceeded (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-StiConstraints
sip-reason-text NoValidationStiConstraints
```

Here, the system provides a “404 NoValidationStiConstraints” Response, and adds "No-TN-Validation" in the FROM.

No-TN-Validation-Unreachable—STI-VS circuit-breaker is open (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-Unreachable
sip-reason-text NoValidationStiUnreachable
```

Here, the system provides a “404 NoValidationStiConstraints” Response, and adds "No-TN-Validation" in the FROM.

No-TN-Validation-ClientError—SBC internal failure (no request sent to STI-VS)

Consider this scenario to be the same as the second configuration examples above, but differing in your **sip-reason-text** and **verstat** configurations, which the SBC uses to find matches for rejecting these calls.

```
verstat No-TN-Validation-ClientError
sip-reason-text NoValidationClientError
```

Here, the system provides a “404 NoValidationStiConstraints” Response, and adds "No-TN-Validation" in the FROM.

No-TN-Validation-Bypass—The STIR requests that are bypassed

If the entry has a verstat of “No-TN-Validation-Bypass”, and a reason-code that is “empty" or "0”, the system rejects the call when the STIR request is bypassed.

```
verstat No-TN-Validation-Bypass
sip-reason-text Null
```

The Verstat match key for call rejection is “No-TN-Validation-Bypass”.

In this case, the SBC rejects the call with:

- A “403 Forbidden” Response
- STI-VS-REASON empty
- Reason Headers:
  - ATIS SIP Reason header—SIP;cause=403;text=Forbidden
  - 3GPP SIP Reason header—empty
- The verstat No-TN-Validation in the FROM header

Reason-code Missing from the Response Body

For cases in which the system receives STI-VS responses in which the reason-code is not present in the verificationResponse of the JSON body, you can configure a **sti-response-treatment-entry** that causes the system to populate the reason-code from the HTTP body based on operation mode and response:

- For ATIS—When receiving 4xx or 5xx responses from the STI-VS, use the reason-code from the HTTP response code, which the system gets from the response's status line.
- For 3GPP—When receiving any response from the STI-VS, use the reason-code from the HTTP response code, which the system gets from the response's status line.

Consider the following **sti-response-treatment-entry**:

```
(sti-response-treatment-entry)show
sti-response-treatment-entry
verstat No-TN-Validation-None
reason-code 400
role STI-VS
sip-reason-code
sip-reason-text
```

This behavior would take effect when the system receives an HTTP response, as shown below, that triggers the **sti-response-treatment-entry** above.

```
HTTP/1.1 400 Bad Request
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
"requestError": {
"serviceException":
           {"messageId": "SVC4001"
              "text": "Error: Missing mandatory parameter '%1'",
           "variables": ["iat"]
}
}
}
```

In this case the SBC rejects the call with a “400 Bad Request” response. In addition, the STI-VS-REASON is "SIP ;cause=400 ;text="Bad Request".

Mapping SIP to HTTP Headers and Parameters

You can configure the SBC with static mapping of signaling information to and from SIP INVITEs and HTTP requests or responses. This mapping provides a means of conveying SIP header and parameter information within HTTP headers and vice-versa. The HTTP exchanges can be during authentication and verification procedures. This feature applies to both ATIS and 3GPP modes.

This feature performs the following functions:

- Adds headers and their new parameters to the rules' targets
- Modifies existing headers with the new parameters presented by the rule

You configure this feature using the **sti-header-mapping-ruleset** in the **session-router**. From this parameter, you create multi-instance sub-elements that contain your SIP/HTTP mapping rules. You then assign rulesets to:

- **sti-server**
- **sti-config**

The ruleset assigned to a **sti-server** takes precedence over the **sti-config** configuration.

The SBC uses these rulesets to perform actions when presented with STIR/SHAKEN flows, including:

- Selecting the source-header and/or source-parameter you specify with configuration
- Adding any new header into the target content
- Adding or modifying the selected parameter to the target content
- Adding or modifying the selected parameter to a target-header

For each SIP call that invokes STIR/SHAKEN authentication or verification, the SBC first checks the applicable **sti-server** for a configured **sti-header-mapping-ruleset-name**. If found, the SBC uses the corresponding ruleset to get the mapping details. If it is not found, the SBC checks for an applicable configuration in the **sti-config** for the same purpose.

You configure a ruleset by specifying:

- The ruleset identifier
- The headers and/or parameters the system uses as the source for the mapping
- The headers and/or parameters the system uses as the target for the mapping
- The role, STI-AS or STI-VS, within which the ruleset operates
- The direction of the mapping, based on the perspective of the SBC

For call flows that include a single HTTP request and response, the SBC behaviors are the same for both ATIS and 3GPP modes. In 3GPP mode when there are both SHAKEN and DIV signing request/responses, the SBC:

- Adds the mapped data to both the SHAKEN and DIV signing request towards the STI-AS.
- Uses the following behaviors when receiving responses for both SHAKEN and DIV signing requests:
  - If it receives multiple responses, the SBC processes and saves the first response. When it receives the second, the SBC processes this second response but does not save it. This results in any final processing using the first response.
  - If the same header for which mapping is set is present in both responses, the SBC have either value from the response that will be processed later that overwrites value from the response that was processed first, or headers that would be duplicated (if [^] or [~] is used in the mapping-rule).

Header and Parameter Mapping Statistics

The SBC provides a command to show statistics on traffic for which is has performed header modification. To see these statistics, run the **show stir header-mapping**.

Mapping SIP to HTTP Headers

You can configure the SBC with static mapping to and from SIP INVITEs and HTTP requests or responses. This mapping provides a means of conveying SIP header information within HTTP headers and vice-versa. The HTTP exchanges can be during authentication or verification procedures. This feature adds headers in the rules' targets, or modifies existing headers.

When using these rulesets, the SBC performs functions including:

- Selecting the configured **source-header**

- Adding or modifying the configured **target-header** with that value

Selecting a Header

To perform this function, the SBC selects your configured **source-header** from a SIP INVITE or from an HTTP response received from the STI server, depending upon your **direction** configuration. After getting the value of this header, the SBC inserts this value into the **target-header** of the HTTP request or the egress SIP INVITE.

- If a mapping rule has a **source-header** configured with a SIP header and the direction of the mapping set to **outbound**, the SBC retrieves the SIP header values from the ingress INVITE.
- If a mapping rule has a **source-header** configured with an HTTP header and the direction of the mapping set to **inbound**, the SBC retrieves the HTTP header value from the HTTP response sent by the STI-AS or STI-VS server, depending on your **role** configuration.

Rules for selecting a header include:

- The SBC retrieves a SIP or HTTP header value based on the digit in the subscript, if present, after the header name in your configuration. If you have not configured a subscript, the SBC uses the first header.

> **Note:**
>
> This is equivalent to writing the header name with a 0 in the subscript. For example, Attestation-Info[0].

- The SBC only selects a header if the number of the header in the ingress INVITE that matches source-header the configuration is greater than 0 and is greater than the digit present in the subscript.

Adding/Modifying a Header

This feature allows you to add or modify the target header. When performing the function when the **target-header** is a SIP header and the direction is **inbound**, the SBC:

- Adds a new header when the target header count, meaning either the HTTP header in the POST request or the SIP header in the outgoing INVITE, is the same as the digit in the subscript of the header name in your configuration.
- Modifies the header referenced by the subscript when the digit in the header name subscript is less than the number of headers in the HTTP request or SIP INVITE.
- Does not perform the mapping when the subscript is greater than the number of headers.
- When your configuration includes the ^ symbols, adds the header at the first position.
- When your configuration includes the ~ symbols, adds the header at the last position.

When performing the function when the **target-header** is an HTTP header and the direction is **outbound**, the SBC:

The SBC does not support indexing HTTP headers. Instead, the SBC appends existing HTTP headers, such as when the SBC creates the header using multiple rules, with the new header after a comma.

Configuration

You configure this feature by creating a **sti-header-mapping-ruleset** within the **session-router** element and applying your ruleset name to the **sti-header-mapping-ruleset-name** parameter within the **sti-config** or a **sti-server** element.

Parameters for **sti-header-mapping-ruleset** include:

- **name**—Unique identifier for the ruleset
- **mapping-rules**—Enters the mapping rules sub-element, from which you define the details or each rule.

Parameters for **mapping-rules** include:

- **id**—Unique identifier for the rule, allowing the execution of multiple rules from a single **mapping-rules** element.
- **source-header**—Name of the source header. This could be a SIP header or an HTTP header.
- **target-header**—Name of the target header. This could be a SIP header or an HTTP header.
- **direction**—The direction of the translation. Values include:
  - **outbound**— Map headers from the SBC to an external STI server (SIP to HTTP).
  - **inbound**—Map headers received from an external STI server to the SBC (HTTP to SIP).
- **role**—The type of operation the system is performing, including STI-AS or STI-VS

You can configure SIP **source-header** and **target-header** parameters with the header names using the same syntax you use within HMR configurations. This formatting does not apply to HTTP headers. These header names support alphanumeric characters, which specify the header name, as well as the following special characters, which can be used in the custom header name:

- . (period)
- ! (exclamation point)
- % (percent sign)
- * (star)
- _ (underscore)
- + (plus sign)
- `(closed quote)
- ' (apostrophe)
- ~ (tilde)
- - (dash)
- @ (ampersand)

Note the example configuration using formatting:

```
ORACLE(mapping-rules)#target-header P-Origination-Id[0]
```

Header names can end with or without a subscript operator, enclosed in brackets, []. Subscript operators include a digit, a caret or a tilde. These operators indicate the sequence number of the header in the incoming or outgoing SIP INVITE, or in the HTTP request or response. Operators include:

- [^] — Last occurrence
- [~] — first occurrence
- [0] — first
- [1] — second
- [2] — third

If a subscript operator is not provided, it means that the subscript is 0, referring to the first occurrence of the header.

Note the example configuration using subscripts:

```
ORACLE(mapping-rules)#target-header P-Origination-Id[0]
```

The SBC provides syntax validation after you run the **done** command. When your source or target header values violate syntax, the SBC throws the following error message.

```
Value must be alphanumeric and can contain the following special characters
@.!%*_+`'~-.
It can optionally  be followed by a digit, ~ or ^ enclosed in a subscript
operator '[]'
```

A more specific syntax error includes the following, explaining that an outbound rule is unable to apply a target header that uses subscripts.

```
% Cannot save configuration object
When the direction is set as outbound, the target header can't contain the
subscript [] operator
```

Note also that an inbound rule is unable to apply a source header that uses subscripts.

Mapping SIP to HTTP Parameters

You can configure the SBC to extend upon header mapping with parameter mapping for both authentication and verification procedures. Parameter mapping uses similar configuration and behavior as header mapping, allowing you to target specific parameters within SIP and HTTP messages for mapping or manipulation. You configure source and target parameters in your rules, either on their own or in conjunction with header parameters, to specify changes to messages you need within STIR/SHAKEN signaling.

The values you use to specify headers and parameters in the **mapping-rules** can be SIP or HTTP headers or parameters, or null. The combinations of configured and null fields result in 8 functions that either insert new or manipulate existing information. These functions include:

- SIP Header Parameter to HTTP Body (JSON Claim)
- SIP Header to HTTP Body (JSON Claim)
- SIP Header Parameter to HTTP Header Parameter
- SIP Header Parameter to HTTP Header

- HTTP Body (JSON Claim) to SIP Header
- HTTP Body (JSON Claim) to SIP Header Parameter
- HTTP Header to SIP Header Parameter
- HTTP Header Parameter to SIP Header Parameter

> **Note:**
>
> A JSON Claim is sent within an HTTP Body.

Mapping/manipulation from SIP to HTTP uses the direction Inbound, whereas HTTP to SIP uses Outbound. Both the inbound and the outbound configuration present challenges that can generate unexpected results. To achieve the desired results, consider the behaviors the SBC takes based on your configuration and the input presented by the network for each function type. Whether you populate a given configuration parameter or leave it null can significantly impact the result.

Configuration and operations guidelines for SIP to HTTP configurations include:

- SIP Header Parameter to HTTP Body (JSON Claim):
  - If you populate both the **source-header** and **target-header** parameters and leave the **source-param** and **target-param** parameters as null, the system uses the **target-header** to carry the information. (For example, the system relays SIP Header values as HTTP Headers.)
  - If you populate the **target-param** parameter and leave the **target-header** parameter as null, the system uses the **target-param** to carry the information in a JSON claim (HTTP body).
  - If there are rules configured to manipulate mandatory/default/standard JSON claims, the configured rules take priority over any mandatory claims.
- SIP Header Parameter to HTTP Header Parameter:
  - You must populate both the **source-header** and **source-param** parameters for these mapping-rules.
  - If you populate the **target-header** parameter and leave the **target-param** as null, the system populates the **target-header**.
  - If you populate both the **target-header** and **target-param** parameters, the system adds the value of the **source-header**'s **source-param** value to the **target-header**'s **target-param**.
  - The system refers to the semi-colon (;) delimiter to identify an entire SIP header parameter.
- SIP Header Parameter to HTTP Header: (The SIP Header parameter value will be populated as target-header (HTTP-header.)
- SIP Header to HTTP Body (JSON Claim): The system relays the SIP Header value to the HTTP Body - JSON Claim

Configuration and operations guidelines for HTTP to SIP configurations:

- HTTP Body (JSON Claim) to SIP Header:
  - If you populate the **source-param** parameter and leave the **source-header** parameter as null, the system uses your **source-param** value to fetch the information.

- If you populate the **target-header** parameter and leave the **target-param** as null, the system uses source values to create or populate the target SIP Header. (For example, the system relays the HTTP body, JSON Claim value as the SIP header.)
- If you populate both **target-header** and **target-param** parameters, the system uses source values to populate the existing SIP Header and parameter.
- If the target SIP header parameter is already present in the incoming INVITE, the system replaces the value of the parameter with the **source-param** value.
- If you populate the **source-header** parameter and leave the **source-param** parameter as null, the system fetches the value of the **source-header**. (For example, the system relays HTTP header values as SIP headers.)
- If there are multiple keys in the JSON body with same name, the system fetches the last occurrence of the key to specify the **source-param** value.

> **Note:**
>
> A 'Key' is the key from key-value pair in a JSON claim.

- When the system looks within the HTTP Body (JSON) to find a target, it only looks as far as the second level of objects within JSON syntax to find a **source-param**. For example,consider the following JSON body as a source for the SBC to fetch a target. In this message, the syntax defines "serviceException" as a second level object, and “messageId” is third level. The SBC can find "serviceException" as a **source-param**, but not “messageId”,

```
JSON body
{
“requestError”: {
“serviceException”: {
       “messageId”: “SVC4001”,
        “text”: “Error: Missing mandatory parameter ‘%1’”,
        “variables”: [“iat”],
        } }     }
```

- If the system is fetching is a JSON claim object or array, it copies the contents of the **source-param** parameter into the **target-header** as a string. In addition, the system removes any new lines (\n).

> **Note:**
>
> If the contents of the **source-param** parameter is a table or an array, the SBC adds the content value in the format **DQUOTE word DQUOTE**. If the content is a generic string, the SBC adds the value in the format of a **word**, where **word** is defined in RFC 3261 as follows:
>
> ```
> word =1*(alphanum / "-" / "." / "!" / "%" / "*" /
>  "_" / "+" / "`" / "'" / "~" /
> "(" / ")" / "<" / ">" /
> ":" / "\" / DQUOTE /
> "/" / "[" / "]" / "?" /
> "{" / "}" )
> ```

  - The system does not support indexing during an HTTP body to SIP header mapping procedure.

- HTTP Header Parameter to SIP Header Parameter:
  - You must configure both the **source-param** and **source-header** parameters.
  - If you populate the **target-header** parameter and leave the **target-param** parameter as null, the system uses source values to create or overwrite the SIP Header.
  - If you populate both the **target-header** and **target-param** parameters, the system uses source values to populate the parameter in the existing SIP Header.
  - The system refers to the semi-colon (;) and comma (,) delimiters to identify an entire HTTP header parameter.
- HTTP Body (JSON Claim) to SIP Header Parameter:
  - Keep the **source-header** null; configure the **source-param**
  - The system uses the **source-param** to find the JSON claim from HTTP-body and populates the SIP Header Parameter with that value.
- HTTP Header to SIP Header Parameter: Keep the **source-param** null; the **target-header** and **target-param** are mandatory.

Parameter Mapping Configuration Examples

Similar to header mapping, you configure this parameter mapping feature by creating a **sti-header-mapping-ruleset** within the **session-router** element and applying your ruleset name to the **sti-header-mapping-ruleset-name** parameter within the **sti-config** or a **sti-server** element.

Parameters within **mapping-rules** that apply to parameter mapping include:

- **source-param**—Case insensitive name of the source parameter. This could be a SIP or an HTTP header parameter based on the source header. If the source-header is null and the direction is inbound, the system identifies this value as a key from the HTTP body JSON claim.
- **target-param**—Case insensitive name of the target parameter. This could be a SIP or an HTTP header parameter based on the source header. If the source-header is null and the direction is inbound, the system identifies this value as a key from the HTTP body JSON claim.

The **id**, **source-header**, **target-header**, **direction**, and **role** have the same or similar functions within rules you configure for header mapping.

Outbound Configuration - SIP Header Parameter to HTTP Body (JSON Claim)

This configuration uses an empty **target-header**, resulting in this rule manipulating the HTTP body.

```
source-header        P-Charging-Vector
source-param         ICID-Value
target-header        ""
target-param         ICID
direction            outbound
```

Result—The system copies the contents of the ICID value from the P-Charging-Vector header into the HTTP body JSON claim.

Outbound Configuration - SIP Header Parameter to HTTP Header Parameter

This configuration sets all parameters, resulting in this rule manipulating the HTTP Header parameter, X-Charging-Vector.

```
source-header        P-Charging-Vector
source-param         ICID-Value
target-header        X-Charging-Vector
target-param         ICID
direction            outbound
```

Result—The system copies the contents of the ICID-value from the P-Charging-Vector header into the HTTP Header X-Charging-Vector as the ICID parameter.

Outbound Configuration - SIP Header Parameter to HTTP Header

This configuration uses an empty **target-param**, resulting in this rule manipulating the HTTP header.

```
source-header        P-Charging-Vector
source-param         ICID-Value
target-header        X-Charging-Vector
target-param         ""
direction            outbound
```

Result—The system copies the contents of the ICID-value from the P-Charging-Vector header into the HTTP Header X-Charging-Vector.

Outbound Configuration - SIP Header to HTTP Body (JSON Claim)

This configuration uses an empty **source-param** and **target-header**, resulting in this rule manipulating the HTTP header.

```
source-header        P-Charging-Vector
```

```
source-param          ""
target-header         ""
target-param          icid
direction             outbound
```

Result—The system copies the contents of the P-Charging-Vector header into the HTTP body JSON claim with the key: icid.

Inbound Configuration - HTTP Body (JSON Claim) to SIP Header

This configuration uses an empty **source-header** and **target-param**, resulting in this rule manipulating the SIP header from the HTTP body. If the incoming INVITE does not contain the **target-header** you configured, the system creates a SIP header using your **target-header** value as a name that includes the value of your **source-param**.

```
source-header         ""
source-param          ICID
target-header         P-Charging-Vector
target-param          ""
direction             inbound
```

Result—The system copies the contents of ICID from the **source-param** into the P-Charging-Vector SIP header with the value ICID (from the **source-param**).

Inbound Configuration - HTTP Body (JSON Claim) to SIP Header Parameter

This configuration uses an empty **source-header**, resulting in this rule manipulating the SIP header parameter from the HTTP body. If the incoming INVITE already contains the **target-param** you configured, the system overwrites that parameter with the value of your **source-param**.

```
source-header         ""
source-param          ICID
target-header         P-Charging-Vector
target-param          icid-value
direction             inbound
```

Pre-requisite—HTTP response should have the configured **source-param**.

Result—The system copies the contents of the ICID from the HTTP body into the existing P-Charging-Vector SIP header with a new parameter called icid-value (from the **source-param**).

Inbound Configuration - HTTP Header to SIP Header Parameter

This configuration uses an empty **source-param**, resulting in this rule manipulating the SIP header parameter using the entire contents of the HTTP header. If the incoming SIP INVITE does not contain your configured **target-param**, the system does not apply this rule.

```
source-header         X-Charging-Vector
source-param          ""
target-header         P-Charging-Vector
```

```
target-param          icid-value
direction             inbound
```

Pre-requisite—HTTP response should have the configured souce-header.

Result—The system copies the contents of the X-Charging-Vector into the existing P-Charging-Vector SIP header with a new parameter called icid-value (from the **target-param**).

Inbound Configuration - HTTP Header Parameter to SIP Header Parameter

This configuration sets all parameters, resulting in this rule manipulating the SIP header parameter using the entire contents of the HTTP header. If the incoming SIP INVITE already contains your configured **target-param**, the system overwrites the value of that parameter with you configured **source-param**.

```
source-header         X-Charging-Vector
source-param          ICID
target-header         P-Charging-Vector
target-param          icid-value
direction             inbound
```

Result—The system copies the contents of the ICID parameter from the X-Charging-Vector into the existing P-Charging-Vector SIP header with a new parameter called icid-value (from the **target-param**).

Bypass STI-AS Signature and STI-VS Verification Requests

You can configure the SBC to bypass STI-AS signature requests and/or STI-VS verification requests despite the presence of the applicable headers in incoming INVITEs. These bypass configurations, including the **stivs-bypass-header** and **stias-bypass-header** parameters are available within the **sti-config**. These parameters enable the following behaviors:

- **stivs-bypass-header**—When enabled, the SBC detects SIP INVITEs that are subject to STI-VS verification and bypasses the entire verification process, instead adding "verstat=No-TN-Validation" and forwarding the egress INVITE.
- **stias-bypass-header**—When enabled, the SBC detects SIP INVITEs that are subject to STI-AS authentication.

The system tracks and reports on bypassed STIR requests using server, system, STIR realm, session agent and sip interface statistics with the labels "STI-AS ByPass" and "STI-VS ByPass".

> **Note:**
>
> When you enable these flags, the bypass scenario takes precedence over "TN missing" and "Identity missing" scenarios.

Remove the verstat from Messages to the Caller

You can configure the SBC to remove the verstat parameter from SIP messages sent from the system to the calling side. You enable this behavior using the **stop-adding-verstat-towards-caller** parameter within the **sti-config**.

The default SBC behavior is to add the verstat parameter to responses sent out to both the UAC (calling side) and the UAS (called side) of applicable calls. This feature keeps the verstat out of messages sent to the calling side. When configured, the system does not add verstat to all 18x and 200 responses to the INVITE towards the calling side, and all in-dialog requests towards the calling party.

Configure STIR/SHAKEN Functions

The subsections below provide instructions on configuring the functions the STIR/SHAKEN client perform.

Configure Header and Parameter Mapping

Use the steps below to configure SIP and HTTP header and parameter mapping within your STIR/SHAKEN deployment.

You configure this feature by creating a **sti-header-mapping-ruleset** within the **session-router** element and applying your ruleset name to the **sti-header-mapping-ruleset-name** parameter within the **sti-config** or a **sti-server** element.

This is a subset of overall STIR/SHAKE configuration.

1. Navigate to the **sti-header-mapping-ruleset**.

```
ORACLE# configure terminal
ORACLE#(config)session-router
ORACLE(session-router)#sti-header-mapping-ruleset
ORACLE(sti-header-mapping-ruleset)#
target-header P-Origination-Id[0]
```

2. Name your ruleset.

```
ORACLE(sti-header-mapping-ruleset)# name MyRuleset
```

3. Navigate to the mapping rules.

```
ORACLE(sti-header-mapping-ruleset)# mapping-rules
ORACLE(mapping-rules)#
```

4. **id**—Enter a unique identifier for the rule, allowing the execution of multiple rules from a single **mapping-rules** element.

```
ORACLEORACLE(mapping-rules)# id My1stRule
```

5. **source-header**—Enter the name of the source header. This could be a SIP header or an HTTP header.
6. **source-param**—Enter the case insensitive name of the source parameter. This could be a SIP parameter or an HTTP parameter.

   If the source-header is null and the direction is inbound, the system identifies this value as a key from the key-value pair from the HTTP Body JSON claim.

7. **target-header**—Enter the name of the target header. This could be a SIP header or an HTTP header.
8. **target-param**—Enter the case insensitive name of the target parameter. This could be a SIP parameter or an HTTP parameter.

   If the target-header is null and the direction is inbound, the system uses this value as a key from the HTTP body JSON claim.
9. **direction**—Choose the direction of the translation. Values include:
   - **outbound**— Map headers from the SBC to an external STI server (SIP to HTTP).
   - **inbound**—Map headers received from an external STI server to the SBC (HTTP to SIP).
10. **role**—Enter the type of operation the system is performing, including STI-AS or STI-VS
11. Execute the Done command, then save and activate your configuration.

Configure the sti-reason-header-config

To configure global STIR/SHAKEN configuration parameters on the SBC, use the **sti-config** object under **session-router**.

1. Access the **sti-config** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sti-config
ACMEPACKET(sti-config)#
```

2. Type **sti-reason-header-config** and press Enter.

```
ACMEPACKET(sti-config)# sti-reason-header-config
ACMEPACKET(sti-reason-header-config)#
```

3. Type **name** followed by the alphanumeric string you want to use to name this **sti-reason-header-config** and press Enter.

```
ACMEPACKET(sti-reason-header-config)# name MySTIReasonHeaderConfig
```

4. Type **sti-reason-header-entry** to access the entry configuration sub-element and press Enter.

```
ACMEPACKET(sti-config)# sti-reason-header-entry
ACMEPACKET(sti-reason-header-entry)#
```

5. Type **scenario** followed by the enumerated value that this header-entry must match and press Enter. Values include:
   - sti-server-timeout (Default)
   - invalid-sti-response
   - use-identity-header
   - tn-missing
   - sti-constraints-exceeded

- sti-server-unreachable
- internal-client-error
- sti-server-bypass

```
ACMEPACKET(sti-reason-header-entry)# scenario invalid-sti-response
```

6. Type **cause-code** followed by your selection for this cause code and press Enter. You can use any value within the range 400-699.

```
ACMEPACKET(sti-reason-header-entry)# cause-code 500
```

7. Type **reason-text** followed by the alphanumeric string you want to use for this code and press Enter. The system sets this parameter to an empty string by default for all scenarios. However, it is internally mapped to applicable strings based on scenario. Your setting overwrites the default to your value for each scenario.

```
ACMEPACKET(sti-reason-header-entry)# reason-text MyReasonText
```

8. Type **done** to complete this **sti-reason-header-entry**. Create as many entries as needed to support your deployment.
9. Type **done** to complete this **sti-reason-header-config**.
10. You must apply this **sti-reason-header-config** to either a **sti-server** or globally to the **sti-config**. The **sti-server** setting takes precedence. If applying to the **sti-config**, type **sti-reason-header-config-name** followed by the name you configured in Step 6 above.

```
ACMEPACKET(sti-config)# sti-reason-header-config-name
MySTIReasonHeaderConfig
```

11. Type **done** to complete this **sti-reason-header-config** configuration.
12. Save and activate your configuration.

Configure the sti-response-treatment-config

To configure global STIR/SHAKEN configuration parameters on the SBC, use the **sti-config** object under **session-router**.

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)#
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)#
```

3. Type **sti-config** and press Enter.

```
ACMEPACKET(session-router)# sti-config
ACMEPACKET(sti-config)#
```

4. Type **sti-response-treatment-config** and press Enter.

```
ACMEPACKET(sti-config)# sti-response-treatment-config
ACMEPACKET(sti-response-treatment-config)#
```

5. Type **name** followed by the alphanumeric string you want to use to name this **sti-response-treatment-config** and press Enter.

```
ACMEPACKET(sti-response-treatment-config)# name
MySTIResponseTreatmentConfig
```

6. Type **sti-response-treatment-entry** to access the entry configuration sub-element and press Enter.

```
ACMEPACKET(sti-config)# sti-response-treatment-entry
ACMEPACKET(sti-response-treatment-entry)#
```

7. Type **verstat** followed by the enumerated value that this verstat must match and press Enter. Values include:
   - No-TN-Validation
   - TN-Validation-Passed
   - TN-Validation-Failed
   - Any custom string

```
ACMEPACKET(sti-response-treatment-entry)# verstat No-TN-Validation
```

8. Type **reason-code** followed by your selection for this cause code and press Enter. You can use any value within the range 0 or 100 to 599.
   - Default: 0
   - 403
   - 428
   - 436
   - 437
   - 438
   - custom-code

```
ACMEPACKET(sti-response-treatment-entry)# cause-code 500
```

9. **role**—
   - STI-VS

```
ACMEPACKET(sti-response-treatment-entry)# role SI-VS
```

10. Type **sip-reason-code** followed by the alphanumeric string you want to use for this code and press Enter. The system sets this parameter to an empty string by default for all

scenarios. However, it is internally mapped to applicable strings based on scenario. Your setting overwrites the default to your value for each scenario.

```
ACMEPACKET(sti-response-treatment-entry)# reason-code 404
```

11. Type **sip-reason-text** followed by the alphanumeric string you want to use for this code and press Enter. The system sets this parameter to an empty string by default for all scenarios. However, it is internally mapped to applicable strings based on scenario. Your setting overwrites the default to your value for each scenario.

    ```
    ACMEPACKET(sti-response-treatment-entry)# reason-text MyReasonText
    ```

12. Type **done** to complete this **sti-response-treatment-entry**. Create as many entries as needed to support your deployment.
13. Type **done** to complete this **sti-response-treatment-config**.
14. You must apply this **sti-response-treatment-config** to either a **sti-server** or globally to the **sti-config**. The **sti-server** setting takes precedence. If applying to the **sti-config**, type **sti-response-treatment-config-name** followed by the name you configured in Step 6 above.

    ```
    ACMEPACKET(sti-config)# sti-response-treatment-config-name
    MySTIResponseTreatmentConfig
    ```

15. Type **done** to complete this **sti-response-treatment-config** configuration.
16. Save and activate your configuration.

Additional Administrative Features

This section groups assorted behaviors that the SBC supports within its STIR/SHAKEN operations.

The SBC supports the following additional features, some of which include further configuration:

- Accepting Non-Standard verstat
- Bypass STI-AS Signature and STI-VS Verification Requests
- Media Policy within the **http-server** element
- TLS Session Key Logging
- Remove the verstat from Incoming Requests
- Extract verifyResults Content from div Passports for 3GPP
- Managing Curl within HTTP Server Interactions

Accepting Non-Standard verstat

By default, the SBC treats some non-standard verstat responses as failed. This is true even if verification is ultimately successful. This happens because the SBC cannot parse these verstat

values. The SBC also increments the “STI-VS Failed Verification” statistics when it receives these non-standard verstat responses. An example non-standard verstat is shown below.

```
verstat=TN-Validation-Passed;attest=A;origid=1cab14fd-7c29-4c95-9177-
a8317bb0bbbb@172.22.247.57
```

When you configure the **verstat-delimiter** option under **sti-server**, the SBC can parse customized verstat value on the basis of configured delimiter and fetch the valid verstat value. An example for configuring **verstat-delimiter** is shown below.

```
ORACLE(sti-server)# options + verstat-delimiter=;
```

Here, the delimiter is a semi-colon (;).

Oracle recommends that you consider configuring the **verstat-delimiter** option with the applicable character, such as a semi-colon, as a tool for parsing the entire verstat.

Media Policy for the HTTP Client Element

You can configure the SBC to perform DSCP tagging for both media and the wancom0 interface on STIR/SHAKEN HTTP traffic. To support this, enabling the MAN compliance option exposes a **media-policy** reference parameter within the **http-client** element. You configure a **media-policy** for this traffic and apply it to the **http-client**.

The applicable parameter within this **media-policy** is the **tos-value**. When configured, the **tos-value** sets egress packets towards the targeted HTTP servers with this TOS value to inform the network about the traffic and handle that traffic with the appropriate priority to the extent that it can.

If there is also a **media-policy** applied to the applicable egress realm, that **media-policy** takes precedence. This means that, if the SBC finds a **media-policy** applied to the realm, it applies the realm's **media-policy** to the HTTP traffic and uses that policy's configured **tos-value**, ignoring the **media-policy** applied to the **http-client**.

TLS Session Key Logging

You can configure the SBC to save TLS session keys used for STIR/SHAKEN TLS traffic between the SBC and the STIR server to a file named curltlskey.log.

To enable this feature you enable the **log-curl-tls-key** parameter in the **system-config**. For security reasons, the SBC also deletes this log hourly.

Managing Curl within HTTP Server Interactions

By default, the SBC cannot configure some Libcurl options. In conjunction with this compliance configuration, you can configure these Libcurl options using the **http-profile** element within the **session-router**. You can create multiple profiles and apply them to individual **sti-server** elements.

Parameters within the **http-profile** include:

- **name**—Specifies the name you use to apply your profile to a **sti-server**. You configure the **http-profile** parameter to a **sti-server** using this name.

- **tcp-keepalive-idle-timer**—Configures the CURLOPT_TCP_KEEPIDLE Libcurl options, which sets the time the SBC waits while a connection is idle before sending keepalive probes.
- **tcp-keepalive-conntimeout-timer**—Configures the CURLOPT_CONNECTTIMEOUT Libcurl option, which sets the maximum time in seconds that the SBC waits for the connection phase to the server to complete.
- **tcp-conn-max-life-time**—Configures the CURLOPT_MAXLIFETIME_CONN Libcurl option, which sets the maximum time in seconds after the creation of a connection that the SBC waits for the connection to become available for reuse for this request.
- **tcp-conn-terminate-method**—Configures the SO_LINGER Libcurl option, which forces the TCP/IP stack to send the connection reset state to the server immediately.
- CURLOPT_TCP_KEEPINTVL—Sets the wait time the SBC uses between sending keepalive probes.
- CURLOPT_MAXAGE_CONN—Sets the maximum time in seconds the SBC waits for an existing connection to have been idle before it reuses the connection for a subsequent request.

> **Note:**
>
> Regarding the handling with **http-clearDead-conn-timer**
> As per the current libcurl code, TCP idle/max life time connection termination is dependent on a new request or existing config **http-clearDead-conn-timer**. The **http-clearDead-conn-timer** value specifies the regular time interval, the SBC uses to monitor/clear the TCP connections.
>
> After current (man-compliance) implementation, even though we configure some value for CURLOPT_MAXAGE_CONN to disconnect the idle connection, connection will not be disconnected after expiration of the max age timer. This is because, by default, libcurl will not disconnect the idle connections as soon as CURLOPT_MAXAGE_CONN expires. Instead it will wait for the new request or dummy request, initiated by the SBC after the expiration of the **http-clearDead-conn-timer**, to execute the CURLOPT_MAXAGE_CONN expiration logic.
>
> This same functionality also applies to the CURLOPT_MAXLIFETIME_CONN option.

Logging Compliance for Number Authentication Mechanism

This section explains how the SBC manages STIR/SHAKEN operations logging.

The SBC creates a separate log file named "stirmsg.log" to record all STIR/SHAKEN transactions. This effectively separates STIR/SHAKEN logs from SIPD (Session Initiation Protocol Daemon) logs and facilitates the collection of STIR/SHAKEN details without adversely impacting the performance of the SBC.

You use this file for STIR/SHAKEN troubleshooting purposes. It can be especially useful when troubleshooting encrypted transactions using TLS 1.2 later TLS 1.3. You can display the contents of this log from the ACLI using the following command.

```
ORACLE#show logfile /opt/logs/stirmsg.log
```

You can configure the verbosity level of these STIR/SHAKEN logs:

- Set the log level for SIPD to debug and logType to STIR to get STIR logs in log.sipd.

```
ORACLE#log-level sipd debug STIR
```

- Reset the log level for SIPD to no-debug for logtype STIR.

```
ORACLE#log-level sipd notice STIR
```

Commands for Testing STIR Deployments

You can configure the SBC to generate and execute test scenarios that you use with a specific **sti-server** on demand. These tests verify STIR/SHAKEN functionality between the SBC and that server. These scenarios are broadly flexible, allowing you to specify the SIP message, the message direction, as well as the HTTP and JSON content before executing the test.

This feature utilizes the **test-stir** branch from the **session-router**. This branch includes all the parameters needed to configure the test. This branch also includes the command to execute the test, and parameters to specify conditions surrounding the test, including whether to execute the test in debug mode. The **test-stir** branch is a single-instance element.

The SBC does not have prepacked tests. You must create them from scratch, limited to the parameters available in the **test-stir** branch. To configure and run a test, you:

1. Set the **sti-server** with which the SBC interacts;
2. Specify the direction of the STIR message as either to-AS, to-VS, from-AS or from-VS;
3. Create the test content as text using the message syntax of the applicable content, including SIP, HTTP and JSON;
4. Apply any pre-configured objects as test context; the **http-client** is required;
5. Configure any test conditions, such as **display-message**;
6. Execute your test;
7. Evaluate your results.

Important considerations when creating and using **test-stir** include:

- When you set the **load-json-content**, **load-http-content** and **load-sip-message** parameters, you typically copy text and paste it after the parameter name. When done pasting, press the Ctrl-D sequence to inform the system you are done.
- The SBC does not validate or verify strings that you paste into the HTTP content or JSON body parameters.
- When you set the **http-mapping-rule**, **tls-profile** and **http-client** parameters, you use the value of the **name** parameter of those pre-configured objects. These settings take precedence over any global or server-specific settings.
- The SBC does not include statistics generated by **test-stir** is its statistics counters.
- The SBC does not include content generated by **test-stir** in CDRs.
- Requests generated by **test-stir** do not trigger call rejection functionality.
- When you enable the **display-message** parameter, the SBC displays the loaded message, HTTP, and JSON content.
- When you enable the **debugging-enabled** parameter, the system sets logging verbosity to debug during the test, and resets the logging verbosity to its original setting after the test is complete.

- When you enable the **debugging-enabled** parameter, the system captures curl and sipd events, such as http-mapping-rule events, in the corresponding log files, which include log.sipd and log.tCurld.
- When you enable the **debugging-enabled** parameter, **http-mode**. which is the HTTP version used, is visible.

Test Direction

When you set the **direction** parameter, you establish whether your test sends information to an AS, to a VS or to the SBC itself. This setting defines the architecture of the test, which parameters are required, and the type of content for the test results:

- When directing test content towards the **sti-server**, the SBC,
- When directing test content towards the SBC, the system does not contact the **sti-server** at all. These tests display the resultant INVITE from the SBC, based on an AS or VS input.

When you set direction to **to-AS**:

- Based on SIP Message from **load-sip-message**
- Request generated to the configured **sti-server**
- Result: The SBC makes an HTTP post request to the loaded **sti-server**. The SBC displays the resultant SIP Message with Identity header, or with Reason header based on the test configuration.

When you set direction to **to-VS**:

- Based on SIP Message from **load-sip-message**
- Request generated to the configured **sti-server**
- Result: SIP Message post VS (with verstat and Reason header if needed)
- Result: The SBC makes an HTTP post request to the loaded **sti-server**. The SBC displays the resultant SIP Message with Identity header, or with Reason header based on the test configuration.

When you set direction to **from-AS**:

- Based on http (load-http-content) or json body (load-json-content) only
- There is no CURL activity performed for these tests. The system only applies the test logic to generate output
- Result: The SBC does not make an HTTP post request to the configured **sti-server**. The SBC displays the resultant SIP Message with Identity header, or with Reason header based on the loaded configuration to test.

When you set direction to **from-VS**:

- Based on http (load-http-content) or json body (load-json-content) only
- There is no CURL activity performed for these tests. The system only applies the test logic to generate output
- Result: SIP Message post VS (with verstat and Reason header if needed)
- Result: The SBC does not make an HTTP post request to the configured **sti-server**. The SBC displays the resultant SIP Message with Identity header, or with Reason header based on the loaded configuration to test.

Test Results Detail

After configuration, you enter the **execute** command, which executes your test. On successful execution of the **test-stir** command, the SBC displays:

1. The Request Sent
2. The <Post Request>
3. The <Headers>
4. The <JSON Body>

Consider the following example output, which the SBC has generated for a test.

```
POST /StirShakenWeb/resources/v1/verification HTTP/1.1
Host: 10.196.176.132
X-InstanceID: 00000000-0000-0000-0000-000000000f00
Content-Type: application/json
X-RequestID: a6bfa948-d529-42e1-7d68-3513da6f90ab
Accept: application/json
Content-Length: 695

{"verificationRequest":

{"identity":"eyJhbGciOiJFUzI1NiIsInBwdCI6InNoYWtlbiIsInR5cCI6InBhc3Nwb3J0Iiwie
DV1IjoiaHR0cDovL2xvY2
FsaG9zdDo5MDAxL1N0aXJTaGFrZW5fU1RJX0NSV2ViL3Jlc291cmNlcy92MS9jZXJ0cy9jZXJ0XzIw
MjAwNDA2MTQzNzA1X0VTM
jU2LnBlbSJ9.eyJhdHRlc3QiOiJDIiwiZGVzdCI6eyJ0biI6IjE4MDAzOTIyOTk5In0sImlhdCI6MT
U5NTYxMDY4Niwib3JpZy
I6eyJ0biI6IjE2MTc4Njc1MzA5In0sIm9yaWdpZCI6IjAwMDAwMDAwLTAwMDAtMDAwMC0wMDAwLTAw
MDAwMDAwMGYwMCJ9.
7u2KnSXyyycXlol3MCVcRGiIgVHZjUX1b8dVTdPOuNZpOrL-7xDy2vs82T4tkafQNpXLEKk7MSrwGC
SQ0Ywnug;
info=<<http:\/\/localhost:9001\/StirShaken_STI_CRWeb\/resources\/v1\/certs\/
cert_20200406143705_ES256.pem>;
ppt=shaken>","time":1695999486,"to":{"tn":["19000100201"]},"from":
{"tn":"1234567891"}}}
```

The response received (if any) includes:

1. <HTTP 200/OK (or any other status line>
2. <Headers>
3. <JSON Body>

For example:

```
HTTP/1.1 200 OK
Server: Oracle Stir Server/0.2 Python/2.7
content-type: application/json
content-length: 70

{"verificationResponse": {"verstatValue": "TN-Validation-Passed"}}
```

The following displays test results that generated an INVITE with verstat or Identity headers.

```
INVITE sip:1900010020@192.168.84.188:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.84.151:5060;branch=z9hG4bK-23178-7153-0
From: sipp
<sip:1234567891@192.168.84.151:5060>;tag=23178SIPpTag007153;verstat=TN-
Validation-Passed
To: 12345 <sip:19000100201@192.168.84.188:5060>
Call-ID: 7153-23178@192.168.84.151
CSeq: 1 INVITE
Contact: sip:sipp@192.168.84.151:5060
Identity:
eyJhbGciOiJFUzI1NiIsInBwdCI6InNoYWtlbiIsInR5cCI6InBhc3Nwb3J0IiwieDV1IjoiaHR0cD
ovL2xvY2FsaG9zdDo5
MDAxL1N0aXJTaGFrZW5fU1RJX0NSV2ViL3Jlc291cmNlcy92MS9jZXJ0cy9jZXJ0XzIwMjAwNDA2MT
QzNzA1X0VTMjU2LnBlbSJ9.eyJhd
HRlc3QiOiJDIiwiZGVzdCI6eyJ0biI6IjE4MDAzOTIyOTk5In0sImlhdCI6MTU5NTYxMDY4Niwib3J
pZyI6eyJ0biI6IjE2MTc4Njc1
MzA5In0sIm9yaWdpZCI6IjAwMDAwMDAwLTAwMDAtMDAwMC0wMDAwLTAwMDAwMDAwMGYwMCJ9.7u2Kn
SXyyycXlol3MCVcRGiIgVHZj
UX1b8dVTdPOuNZpOrL-7xDy2vs82T4tkafQNpXLEKk7MSrwGCSQ0Ywnug;info=<<http://
localhost:9001/StirShaken_STI_CR
Web/resources/v1/certs/cert_20200406143705_ES256.pem>;ppt=shaken>
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 139
v=0
o=user1 53655765 2353687637 IN IP4 192.168.84.151
s=-
c=IN IP4 192.168.84.151
t=0 0
m=audio 6000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Configure test-stir

To configure global STIR/SHAKEN configuration parameters on the SBC, use the **test-stir** object under **session-router**.

1. Access the test-stir branch from Superuser mode.

   ```
   ACMEPACKET# test-stir
   ACMEPACKET(test-stir)#
   ```

2. Type **sti-server** followed by the configured name of the **sti-server** with which you are targeting this test.
3. Type **direction** followed by the applicable value and press Enter. Available values include:
   - to-AS
   - to-VS
   - from-AS
   - from-VS

4. Type **load-json-content**, then paste the JSON body as the value to use as for this test. Press Ctrl + D to end the message, and press Enter. This parameter is required when using the directions from-AS and from-VS.
5. Type **load-http-content**, then paste the HTTP content (Headers and Body) as the value to use as for this test. Press Ctrl + D to end the message, and press Enter. This parameter is required when using the directions from-AS and from-VS.
6. Type **load-sip-message**, the paste the SIP INVITE content as the value to use for this test. Press Ctrl + D to end the message, and press Enter. This parameter is required when using the directions to-AS and to-VS.
7. Type **http-mapping-rule** followed by the name of the configured **http-mapping-rule** you want to use for this test, and press Enter.

   This content can be used as a source for VS or AS requests only, and therefore only applies to tests that use the **to-AS** or **to-AS** directions.
8. Type **tls-profile** followed by the name of the configured **tls-profile** you want to use for this test, and press Enter.
9. Type **http-client** followed by the name of the configured **http-client** you want to use for this test, and press Enter.
10. Type **execute** to run this test, and press Enter.
11. Type **display-sip-message** , followed by your value to enable or disable the display of the trigger message for this test, and press Enter. Values include:
    - disabled (default)
    - enabled
12. Type **debugging**, followed by your value to run this test while the system is operating in debug mode, and press Enter. Values include:
    - disabled (default)
    - enabled
13. Type **done**.
14. Save and activate your configuration.

<Enter Concept Title Here>

(Required) <Enter a short description here.>

(Required) <Enter introductory text here, including the definition and purpose of the concept.>

STIR/SHAKEN Client Statistics

The SBC provides standard tools to evaluate, track and troubleshoot client operations. You obtain applicable statistics from the ACLI, with information separated between AS and VS server interactions. The output includes period and lifetime monitoring spans. You can also configure the SBC to provide these statistics using SNMP, HDR and REST. In addition to the above, the SBC enhances applicable CDRs to include calling party authentication information.

You can retrieve STIR/SHAKEN statistics from the SBC using ACLI, SNMP, HDR and REST. For this reporting, the term “system wide” is the sum of all requests. For all statistics outputs, the SBC reports values on **session-agent**, **sip-interface**, **realm**, STI Server and system wide bases:

- Successful requests and responses—Based on receipt of a 200 OK.
- Unsuccessful requests and responses
- Successful and unsuccessful verifications
- Policy exceptions
- Service exceptions
- The presence and absence of a SIP Identity header in INVITEs
- The number of egress signing requests initiated
- The number of signing requests failed due to policy exceptions
- The number of signing requests failed due to service exceptions.
- The number of verification requests failed due to policy exceptions
- The number of verification requests failed due to service exceptions
- The number of successful/unsuccessful signing responses (with status/error codes) received
- The number of verification requests initiated per ingress session agent, sip-interface, realm and system wide
- The number of successful/unsuccessful HTTP and verification responses. A successful response is defined as a 200OK containing the verstat parameter. Response categories include:
  - TN-Validation-Passed
  - TN-Validation-Failed
  - No-TN-Validation – For example, the system detected a syntax error in a verification request.
  - 403 – Stale Data
  - 436 – Bad_Identity_Info
  - Policy exceptions
  - Service exceptions

Detail on how the SBC increments statistics include:

- The value for **STI-VS Success Responses** includes all 200 OK responses from the STI-VS service that have a valid payload (JSON).
- The value for **STI-VS Unsuccessful Responses** include those in which the STI-VS server is unable to verify a call, including:
  - Service exceptions
  - Policy exceptions
  - There is no JSON in the response
  - Timeouts and other signaling issues that cause the response to fail
- Successful responses include the STI-VS service verifying the call or authoritatively providing a rejection cause, all provided within a 200 OK STI-VS response.
- Validated calls are calculated by subtracting the value of **STI-VS Failed Verification** from **STI-VS Success Responses**.
- Successful verifications include each call the STI-VS service can verify.

STIR/SHAKEN Statistics in the ACLI

You display basic STIR traffic information at the ACLI using the **show stir** command, appending it with its arguments to further specify the desired information. When you issue the **show stir** command without any parameters, the SBC displays the available arguments.

Valid arguments, along with any subsequent arguments, include:

- show stir agents < agent_id | all >—STIR/SHAKEN Session Agents Statistics
- show stir interface < interface_id | all >—STIR/SHAKEN Sip Interface Statistics
- show stir realm < realm_id | all >—STIR/SHAKEN Realm Statistics
- show stir stats <sti-server | all > —STIR/SHAKEN Server Statistics
- show stir < all >—STIR/SHAKEN System Wide Statistics
- show stir header-mapping < sti-server | all > —Configured STIR/SHAKEN header-mapping statistics

The output below shows all STIR/SHAKEN statistics corresponding to the system's active interfaces referencing the realm name of each interface, including "core" and "access". ACLI statistics correspond to those available using SNMP and HDR. When appending the command with arguments, the system produces these same statistics filtered to your arguments' target.

```
ORACLEsbc# show stir interface all
12:00:30-100
Interface: core
STIR/SHAKEN Statistics                              ---- Lifetime ----
                                                 Recent    Total    PerMax
STI-AS Queries                                     0        4        1
STI-AS Success Responses                           0        4        1
STI-AS Unsuccessful Responses                      0        0        0
STI-AS Service Exception                           0        0        0
STI-AS Policy Exception                            0        0        0

STI-AS sent INVITEs with added shaken PASSporT attest A
                                                   0        0        0
STI-AS sent INVITEs with added shaken PASSporT attest B
                                                   0        0        0
STI-AS sent INVITEs with added shaken PASSporT attest C
                                                   0        0        0
STI-AS sent INVITEs with added div PASSporT
                                                   0        0        0
STI-VS Queries                                     0        0        0
STI-VS Success Responses                           0        0        0
STI-VS Unsuccessful Responses                      0        0        0
STI-VS Success Verification                        0        0        0
STI-VS Failed Verification                         0        0        0
STI-VS Service Exception                           0        0        0
STI-VS Policy Exception                            0        0        0

STI-VS received INVITEs with no PASSporT(s)
                                                   0        0        0
STI-VS received INVITEs with shaken PASSporT
                                                   0        0        0
STI-VS received INVITEs with div PASSporT(s)
```

```
                                          0        0        0
STI-VS sent INVITEs with verstat set to TN-Validation-Passed
                                          0        0        0
STI-VS sent INVITEs with verstat set to TN-Validation-Failed
                                          0        0        0
STI-VS sent INVITEs with verstat set to No-TN-Validation
                                          0        0        0
STI-VS INVITEs Rejected                   0        0        0
STI Server Unreachable                    0        0        0
STI AS Service Unreachable                0        0        0
STI VS Service Unreachable                0        0        0

Realm: access
STIR/SHAKEN Statistics                              ---- Lifetime ----
                                       Recent     Total     PerMax
STI-AS Queries                            0        4        1
STI-AS Success Responses                  0        4        1
STI-AS Unsuccessful Responses             0        0        0
STI-AS Service Exception                  0        0        0
STI-AS Policy Exception                   0        0        0

STI-AS sent INVITEs with added shaken PASSporT attest A
                                          0        0        0
STI-AS sent INVITEs with added shaken PASSporT attest B
                                          0        0        0
STI-AS sent INVITEs with added shaken PASSporT attest C
                                          0        0        0
STI-AS sent INVITEs with added div PASSporT
                                          0        0        0
STI-VS Queries                            0        0        0
STI-VS Success Responses                  0        0        0
STI-VS Unsuccessful Responses             0        0        0
STI-VS Success Verification               0        0        0
STI-VS Failed Verification                0        0        0
STI-VS Service Exception                  0        0        0
STI-VS Policy Exception                   0        0        0

STI-VS received INVITEs with no PASSporT(s)
                                          0        0        0
STI-VS received INVITEs with shaken PASSporT
          0          0          0
STI-VS received INVITEs with div PASSporT(s)
                                          0        0        0
STI-VS sent INVITEs with verstat set to TN-Validation-Passed
                                          0        0        0
STI-VS sent INVITEs with verstat set to TN-Validation-Failed
                                          0        0        0
STI-VS sent INVITEs with verstat set to No-TN-Validation
                                          0        0        0
STI-VS INVITEs Rejected                   0        0        0
STI Server Unreachable                    0        0        0
STI AS Service Unreachable                0        0        0
STI VS Service Unreachable                0        0        0
```

Header Mapping Statistics

You display basic STIR header-mapping information at the ACLI using the **show stir header-mapping** command, appending it with its arguments to further specify the desired information. When you issue the **show stir header-mapping** command without any parameters, the SBC displays the available arguments.

Valid arguments, along with any subsequent arguments, include:

- show stir header-mapping
- show stir header-mapping all
- show stir header-mapping -all
- show stir header-mapping <sti-server-name>

The output below shows statistics corresponding to your configured header-mappings.

```
ORACLEsbc# show stir header-mapping
12:00:30-100
Server: STIR-signing-1
STIR/SHAKEN Header Mapping Statistics        ---- Lifetime ----
                                          Recent    Total    PerMax
STI-AS Outbound Mapping                     0        4        1
STI-AS Inbound Mapping                      0        4        1
STI-VS Outbound Mapping                     0        0        0
STI-VS Inbound Mapping                      0        0        0


Server: STIR-signing-2
STIR/SHAKEN Header Mapping Statistics        ---- Lifetime ----
                                          Recent    Total    PerMax
STI-AS Outbound Mapping                     0        24       11
STI-AS Inbound Mapping                      0        24       11
STI-VS Outbound Mapping                     0        0        0
STI-VS Inbound Mapping                      0        0        0
```

All commands provide reset capability using the same syntax with the **reset stir** command that you use with the **show stir** command.

For example, **reset stir header-mapping -all**

STIR/SHAKEN Statistics in SNMP

You can retrieve statistics on STIR/SHAKEN status using SNMP. The SBC provides statistics on system-wide, server, **sip-interface**, **realm** and **session-agent** bases. The SBC presents STIR statistics using its standard recent, total, and permax statistics.

STIR Stats are supported in SNMP beginning with the apAppsStirMIBObjectsTable. The system organizes all STIR OIDs under this table. To perform a complete SNMP-walk, for example, you invoke this object in your command line. For example:

```
snmpwalk -v 2c -c public -r 0 10.10.10.9 APAPPS-MIB::apAppsStirMIBObjects
```

Subsequent high-level tables provide the basis for each type of configuration source and use entry rows of the configuration objects' names and objects IDs. The agent table, for example,

uses the apAppsStirAgentStatsEntry entry, which consists of apStirAgentName, which is configured in ACLI, and apStirStatsAgentIndex, which is the object ID of the ACLI config element. These tables include:

- apAppsStirServerTable
- apAppsStirAgentTable
- apAppsStirSipInterfaceTable
- apAppsStirRealmTable
- apAppsStirSystemStatsTable—This table's ApAppsStirSystemStatsEntry uses only the apCounterStatsType and apStirStatsType.

The SBC nests a series of index and identification OIDs to specify each target object together with index and counter OIDs to fully specify the complete output. For example, Server-oriented indexes and data category OIDs include:

- 1.3.6.1.4.1.9148.3.16.1.2.4.1 — apAppsStirServerTable Parent Stir Server table
- 1.3.6.1.4.1.9148.3.16.1.2.4.1.1 — apAppsStirServerEntry Server Entry Object
- 1.3.6.1.4.1.9148.3.16.1.2.4.1.1.1 — apStirServerIndex Server Index Object (Index)
- 1.3.6.1.4.1.9148.3.16.1.2.4.1.1.2 — apStirServerName Server Name Object

Data-oriented indexes and data category OIDs used to index and provide counter objects include:

- 1.3.6.1.4.1.9148.3.16.1.2.4.2 — apAppsStirStatsTable Stats Table
- 1.3.6.1.4.1.9148.3.16.1.2.4.2.1— apAppsStirStatsEntry
- 1.3.6.1.4.1.9148.3.16.1.2.4.2.1.2 — apCounterStatsType Stats counter type Object (Index)
- 1.3.6.1.4.1.9148.3.16.1.2.4.2.1.3 — apStirStatsType Stats type Object (Index)

Objects that capture and specify per-server data include:

- 1.3.6.1.4.1.9148.3.16.1.2.4.2.1.1 — apStirStatsServerIndex stats server index Object (Index)
- 1.3.6.1.4.1.9148.3.16.1.2.4.2.1.4 — apStirServerStats Stats Object

The SBC uses this same model to assemble objects and specify data to produce statistics specific to **sip-interface**, **realm** and **session-agent** elements. The SBC uses a similar, albeit simpler model, to capture system-wide data using 5 OID objects.

Deciphering STIR Data in SNMP

The STI server labels are contained in OID APAPPS-MIB::apStirServerName. The OID of each **sti-server** is the same ID as the config object instance. This means that the SNMP table does not start from 0, instead starting from the config-object ID. This is also true for the object IDs of your **sip-interface**, **realm** and **session-agent** elements. Each SBC assigns its own reference numbers sequentially from 0. These numbers serve only as labels. The entries below present the beginning of a walk of two servers, named stirServer1 and stirServer2, using the reference numbers 80 and 99.

- APAPPS-MIB::apStirServerName.80 = STRING: stirServer1
- APAPPS-MIB::apStirServerName.99 = STRING: stirServer2

The format for STIR OID strings is:

- Root label—For example, APAPPS-MIB::apStirServerStats

- The Instance OID
- The type of statistics, including recent, total and permax
- The type of statistics, such as asQueries and vsQueries

For example, the output for an agent query is composed of:

- apStirAgentName which is configured in ACLI
- apStirStatsAgentIndex which is the object ID of the ACLI config element
- The table is indexed by apStirStatsAgentIndex.
- The apCounterStatsType has any of the value in set {recent(1), total(2), or permax(3)}.
- The apStirStatsType is one of the following:
  - asQueries(1)
  - asSuccessResponses(2)
  - asFailResponses(3)
  - asFailServiceException(4)
  - asFailPolicyException(5)
  - vsQueries(6)
  - vsSuccessResponses(7)
  - vsFailResponses(8)
  - vsSuccessVerification(9)
  - vsFailVerification(10)
  - vsFailServiceException(11)
  - vsFailPolicyException(12)
  - serverUnreachable(13)
  - asSentInviteswithShakenPASSportA(14)
  - asSentInviteswithShakenPASSportB(15)
  - asSentInviteswithShakenPASSportC(16)
  - asSentInviteswithdivPASSport(17)
  - vsReceivedInviteswithNoPASSport(18)
  - vsReceivedInviteswithShakenPASSport(19)
  - vsReceivedInviteswithDivPASSport(20)
  - vsSentInviteswithTNValidationPassed(21)
  - sSentInviteswithTNValidationFailed(22)
  - vsSentInviteswithNoTNValidation(23)
  - asServiceUnreachable(24)
  - vsServiceUnreachable(25)
  - vsInviteRejected(26)
  - apStirStatsTypeMax(27)

Example responses to two STIR agent statistics queries, with one value set to 5 and the other to 50, are:

- APAPPS-MIB::apStirAgentStats.80.recent.asQueries = Gauge32: 5
- APAPPS-MIB::apStirAgentStats.80.recent.asSuccessResponses = Gauge32: 50

See the SBC *MIB Guide* for further detail about SNMP function, configuration, and STIR OID detail.

STIR/SHAKEN Statistics in HDR

HDR based STI server statistics are available on the following bases when you enable collection on the HDR groups specified:

- System-wide—stir-stats-system
- Per server—stir-server-stats
- Per **sip-interface**—stir-stats-sip-interface
- Per **realm**—stir-stats-realm
- Per **session-agent**—stir-stats-session-agent

You can use the **stir-stats** HDR group to collect STIR stats. As with all HDR groups, you can configure dynamic collection or collect manually using the **request collection start** command. See the SBC *HDR Resource Guide* for further detail about HDR function, configuration, and STIR object detail.

Once the collector is running, the SBC generates an output file for the statistics category you configure for collection. HDR files that apply to STIR/SHAKEN operations, assuming default file location, include:

- /opt/collect/stir-stats—This group collects per-STI-server statistics.
- /opt/collect/stir-stats-session-agent
- /opt/collect/stir-stats-sip-interface
- /opt/collect/stir-stats-realm
- /opt/collect/stir-stats-system

Each HDR collection includes:

- TimeStamp
- Group Dependent Variable, including:
  - STI-Server
  - Session-Agent
  - Sip-Interface
  - Realm

> **Note:**
>
> The System group has no variable equivalent to this name field.

- AS Queries
- AS Success Responses
- AS Fail Responses

- AS Fail Service Exception
- AS Fail Policy Exception
- AS Shaken Passport A
- AS Shaken Passport B
- AS Shaken Passport C
- AS Div Passport
- VS Queries
- VS Success Responses
- VS Fail Responses
- VS Success Verification
- VS Fail Verification
- VS Fail Service Exception
- VS Fail Policy Exception
- VS No Passport
- VS Shaken Passport
- VS Div Passport
- VS TN-Validation-Passed
- VS TN-Validation-Failed
- VS No-TN-Validation
- VS Invite Rejected

The per-STI-server statistics file (stir-stats) also includes:

- STI Server Unreachable, which tracks the number of time a STI Server was unreachable.
- STI AS Service Unreachable, which tracks the number of time an AS Server was unreachable.
- STI VS Service Unreachable, which tracks the number of time a VS Server was unreachable.

See the SBC *HDR Guide* for further detail about HDR function, configuration, and output detail.

STIR/SHAKEN Information in CDRs

The SBC adds CDR information to the appropriate RADIUS, Diameter and CSV call detail records documenting STIR/SHAKEN activity on the system.

The attribute value pairs (AVP) the system uses to populate RADIUS and/or DIAMETER CDRs are similar in nature and data. The system translates applicable STIR/SHAKEN data for use within these AVPs. You configure local CSV CDRs to collect and retain call data, including STIR/SHAKEN information, by enabling and configuring the system's **account-config**. When configured, the system includes applicable STIR/SHAKEN detail within CDRs that you can send using RADIUS or DIAMETER, as well as local CDRs generated as CSVs for storage within the system or to a configured **push-receiver**.

For RADIUS, the system populates AVPs within extensions carried by VSA 249, which is a custom VSA used to carry multiple extensions. For DIAMETER, the system populates these

AVPs within ACME-specific DIAMETER attributes for distribution by DIAMETER ACRs, according to the DIAMETER protocol.

> **Note:**
>
> When determining how you want to configure CDR collection, bear in mind how XSR output changes when the **cdr-output-inclusive** parameter is enabled or disabled.

See the SBC *Accounting Guide* for detail about CDR generation and output detail.

> **Note:**
>
> You can refine the verstat included in CDRs during verstat retrieval process using the **verstat-delimiter** option in a **sti-server**. Specifying this delimeter can ensure you include the verstat only, as opposed to the entire verstat response.

Populating CDRs with Rejections based on Response Treatments

You configure the **enhanced-cdr** option within the **sip-config** to cause the SBC to populate the following CDR fields with the applicable information:

- Stir-VS-Invite-State:
  When collecting traffic for a server configured as an STI-VS, the SBC populates this field with “Terminated” for rejecting the INVITE due to call rejection, and with “Continued” when the SBC does not reject the INVITE. When collecting for an STI-AS, the SBC leaves this field empty.

  The SBC also writes the value of the Stir-VS-Invite-State into the local CDR files if the value is "Terminated". This function also requires that you enable the **cdr-output-inclusive** parameter in the **account-config** element.
- Stir-VS-Reason:
  The examples below provide additional detail on how the SBC populates this CDR attribute. When you enable the Call Rejection feature and the SBC rejects a call due to this feature, the SBC populates “Stir-Vs-Reason”:
  - For ATIS success responses (HTTP 200 OK):
    - The Sti-Vs-Reason is reasoncode and the reason text from the JSON body
    - If there is no reasoncode and reason text returned in the JSON body, the Sti-VS-Reason is empty
  - For ATIS 4xx/5xx responses, the Sti-VS-Reason is empty
  - For 3GPP 4xx/5xx responses, the Sti-Vs-Reason is the HTTP status code and the response phrase
  - For both ATIS and 3GPP responses:
    - If there is no answer from STI-VS, the Sti-Vs-Reason is empty
    - If there is no request is send to STI-VS, the Sti-Vs-Reason is empty
- Stir-VS-Verstat
  Specific Stir-VS-Verstat population is dependent on STIR/SHAKEN operation mode:
  - ATIS—The value of the matched verstat when the SBC is rejecting the call.

- 3GPP—The final verstat value, which it derives from the SHAKEN and DIV verstat values.

> **Note:**
>
> When a call is rejected based on custom verstat match, the system populates the Stir-VS-Verstat with "No-TN-Validation.

You can further define the contents of the CDR by configuring the **verstat-delimiter** option in each **sti-server**. You can also use this option to ensure accurate matches during call rejection during verification procedures.

> **Note:**
>
> When conveyed over RADIUS, the SBC limits the data for custom or generic verstat values it inserts into the “Sti-Vs-Verstat” attribute to the first 30 bytes, truncating the remaining part.

You configure this option by specifying your desired delimiter, such as a semi-colon.

```
ORACLE(sti-server)+options verstat-delimiter=;
```

This configuration example specifies that the CDR value includes verstat characters up to the first semi-colon. The examples below present population behavior by the SBC:

- Behaviors when you set the **verstat-delimiter** option:
  - If the sti-vs response has the verstat=TN-Validation-Passed
    The Sti-Vs-Verstat in the CDR would be “TN-Validation-Passed”
  - If the sti-vs response has the verstat=TN-Validation-Passed;attest=A;origid=1cab14fd-7c29-4c95-9177-a8317bb0bbbb
    The Sti-Vs-Verstat in the CDR would be “TN-Validation-Passed”
  - If the sti-vs response has the verstat= No-TN-Validation-Custom1234567891011
    The Sti-Vs-Verstat in CDR would be “No-TN-Validation-Custom1234567“
  - If the sti-vs response has the verstat=Custom-verstat;attest=A;origid=1cab14fd-7c29-4c95-9177-a8317bb0bbbb
    The Sti-Vs-Verstat in CDR would be "Custom-verstat”
- Behaviors when you do not set the **verstat-delimiter** option:
  - If the sti-vs response has the verstat=TN-Validation-Passed
    The Sti-Vs-Verstat in the CDR would be "TN-Validation-Passed"
  - If the sti-vs response has the verstat=TN-Validation-Passed;attest=A;origid=1cab14fd-7c29-4c95-9177-a8317bb0bbbb
    The Sti-Vs-Verstat in the CDR would be "TN-Validation-Passed;attest=A;"
  - If the sti-vs response has the verstat= No-TN-Validation-Custom1234567891011
    The Sti-Vs-Verstat in the CDR would be "No-TN-Validation-Custom1234567"

The SBC handles rejected call CDR data differently based on protocol. For Diameter, the SBC generates an EVENT record for rejected calls. This is because the SBC only generates a STOP record for a call if it generated a START record. For Radius, however, the SBC generates a START record even if it does not generate a STOP.

STIR/SHAKEN Client Alarms, Traps and Logs

The SBC provides for standard tools to evaluate, track and troubleshoot client operations.

Alarms

The SBC generates alarms for STI server connection failure and failed REST responses. The SBC raises the trap when the circuit-breaker trips and clears it when the circuit-breaker closes again. Examples of events that would trigger the alarm include:

- Invalid credentials with STI-AS or STI-VS
- Cannot resolve host
- REST API response time out
- Internal REST API query time-out

This alarm name is STIR SERVER UNREACHABLE, generated by the system using a dynamic alarm ID. An example of this alarm is shown below.

```
ORACLE# display-alarms
1 alarms to show
ID      Task    Severity        First Occurred          Last Occurred
4114305 2759    1           2020-10-01 01:02:03     2020-10-01 01:02:03
Count   Description
1    STIR Server 'stirTest1' connection timeout
```

> **Note:**
>
> The SBC sends a corresponding SNMP trap in addition to the alarm when the circuit breaker trips, which happens whenever the server does not reply, and for whatever reason.

You configure the **sti-config** with the circuit-breaker settings from which the SBC determines the status of servers, including when it is unavailable. When unavailable, the SBC raises this alarm and issues the trap. When raised, the alarm contains the IP address or FQDN of the server within a message indicating the issue with the STI and a count number for each time the server has experienced that issue, such as a connection timeout.

When a server goes out of service, the SBC sends the retry requests to verify whether or not the server is back in service. When the server responds, the SBC clears the alarm and issues the clear trap, indicating the condition cleared. You can also clear the alarm manually with the **clear-alarm** command.

The system alarms and SNMP traps are tied together and the SBC raises them at the same time. Whenever the SBC processes a new STIR-enabled call, if the resulting state of the circuit breaker changes, it triggers a new alarm or clears it. This is not executed for every call, but for every circuit breaker status change. So, if an alarm is cleared manually via ACLI or GUI, but the circuit breaker is still in ‘open’ state, the SBC does not raise any new alarms for this **sti-server**. The SBC can only raise a new alarm for an **sti-server** when the circuit breaker closes.

Major and Minor Alarms

When configured to a single IP address, the system raises a major alarm any time it finds that address unavailable.

If you have configured a server with an FQDN, resulting in multiple server resolutions, the system creates a circuit breaker for each server. When there are multiple server resolutions, the SBC implements a major/minor alarm distinction based on the availability of backup servers.

The system generates a major alarm:

- When there is only one address associates with the server, due to a single FQDN resolution, and that address has its circuit-breaker go into the Open state
- When the last available address out of multiple FQDN resolutions has its circuit breaker go into the Open state

The system generates a minor alarm:

- When one of the IP addresses out of multiple FQDN resolutions has its circuit breaker go into the Open state, but there is at least one IP address in the resolution list that still has a circuit breaker in the Close state

Both the alarm and SNMP trap follow these rules.

SNMP Traps

In the event of repeated sequential REST STI server response timeouts the SBC generates an SNMP trap containing the same information as the associated alarm. These objects include:

- apStirServerUnreachableTrap (1.3.6.1.4.1.9148.3.16.2.2.5.0.1)
- apStirServerUnreachableClearTrap (1.3.6.1.4.1.9148.3.16.2.2.5.0.2)

SNMP traps and alarms are generated by the same system processes. You can disrupt this synchronization if you make manual changes to these objects during operation.

> **Note:**
>
> The SBC does not send an SNMP clear if you close the alarm manually. In this state, the SBC cannot raise a STIR server trap again, despite new alarms, without you rebooting the system. Exercise great care when clearing alarms manually.

Applicable Logs

The SBC logs operational functions performed for the STIR/SHAKEN client to the sipd log. You can find these log entries using the following tips:

- Set the sipd log level to debug, and search for lines containing 'STIR', or 'STIR:' for SPL Logic.
- Logs starting with "[SPL]" originate from SPL scripts,
- Logs from STIR scripts contain the prefix "STIR:".

A

RTC Support

This appendix summarizes real-time configuration (RTC) support status for the Oracle Communications Session Border Controller . The table below lists which configuration elements are supported by RTC and which are not.

ACLI Configuration Elements	Parameter Details
Access Control	The access-control configuration object is partially supported by RTC. Specifically, you can add and delete static ACLs without rebooting. Modifying existing static ACLs, reaching ACL capacity, or fixing configuration typos, however, require a reboot.
Accounting Config	N/A
Authentication	N/A
Certificate Record	N/A
Class Profile	N/A
Codec Policy	N/A
DNS ALG Service	N/A
DNS Config	N/A
Enum	N/A
External Policy Server	N/A
External Policy Server	The following H.323 stack subelement parameters are not RTC supported in that, if you save and activate a configuration, calls already in progress are dropped and a reboot is required: • state • isgateway • realm-id • assoc-stack • options • proxy-mode • local-ip • max-calls • max-channels • registration-ttl • terminal-alias • prefixes • ras-port • q931-port • auto-gk-discovery • multicast • gatekeeper • h245-tunneling • gk-identifier • alternate-transport • q931-max-calls • filename

ACLI Configuration Elements	Parameter Details
H.323	The following H.323 stack subelement parameters are not RTC supported in that, if you save and activate a configuration, calls already in progress are dropped and a reboot is required: • q931-start-port • q931-number-ports • dynamic-start-port • dynamic-number-ports
IPSEC	N/A
IWF	N/A
Licensing	N/A
Local Policy	N/A
Local Response Map	N/A
Local Routing Config	N/A
Media Manager	The Media Manager element is supported with the exception of the following parameters: • red-flow-port • red-max-trans • red-sync-start-time • red-sync-comp-time
Media Policy	N/A
Media Profile	N/A
Network Interface	N/A
Net Management Control	N/A
Network Parameters	The Network Parameters element is supported with the exception of the following parameters: • SCTP parameters
NTP Sync	N/A
Packet Trace Config	N/A
Q850 SIP Map	N/A
Realm Config	N/A
Redundancy Config	The Redundancy Config element is supported with the exception of the following parameters: • state • port • cfg-port • cfg-max-trans • cfg-sync-start-time • cfg-sync-comp-time
Session Agent	N/A
Session Group	N/A
Session Router	N/A
Session Constraints	N/A
Session Translation	N/A
SIP Config	The SIP Config element is supported with the exception of the following parameters: • red-sip-port • red-max-trans • red-sync-start-time • red-sync-stop-time

ACLI Configuration Elements	Parameter Details
SIP Feature	N/A
SIP Interface	The SIP Interface element is supported with the exception of the following parameters: • collect, boot-state
SIP Manipulation	N/A
SIP NAT	The SIP NAT element is supported with the exception of the following parameters: • ext-address
SIP Response Map	N/A
SNMP	N/A
Static Flow	N/A
Steering Pool	N/A
Surrogate Agent	N/A
System	The System Config element is supported with the exception of the following parameters: • options
Test Pattern Rule	N/A
Test Policy	N/A
Test Translation	N/A
TLS Global	When configured to support MSRP over TLS, this configuration element is not RTC enabled. If a single TLS profile is used by both SIP and MSRP, RTC changes are applied for the SIP TLS configuration only.
TLS Profile	N/A
Translation Rules	N/A
Trap Receiver	N/A

B

SIP Compatibility Options

The Oracle Communications Session Border Controller (SBC) addresses a wide array of functions that address the need to adapt SIP signaling disparities between UACs, UASs, proxies, softswitches and standards. Many of these functions are primary features of the SBC. But many support limited or customer-specific use cases. For the latter, Oracle often develops element options. These options are not visible in the ACLI help or documented in the *ACLI Reference Guide*, due to their limited scope.

This appendix documents many of those options, limiting the presentation to the users with deployments that need them. Be sure you fully understand the ramifications of these configurations as they are intended to target a specific segment of SBC deployments and may not be appropriate for your environment.

LMSD SIP Call Progress Tone Interworking

The Oracle Communications Session Border Controller supports Legacy Mobile Station Domain interworking tht allows SIP interworking with User Agents that support the 3GPP2 LMSD. The LMSD provides support for existing Mobile Stations in a network that supports IP bearers.

LMSD uses Alert-Info headers in a 180 Ringing response to an INVITE to indicate to a User Agent specific tones to generate locally by the UAC. Most User Agents that do not support LMSD will ignore the SDP in the Alert-Info and will not play ringback locally. The lmsd-interworking option allows for the system to suppress SDP in 180 Ringing, 486 Busy Here, and 503 Service Unavailable responses, so that the UAC plays local ringback.

LMSD Interworking Configuration

You can apply **lmsd-interworking** to the sip-interface facing the LMSD User Agents. For Example, if the endpoints uspporting LMSD are located in the core realm, then the **lmsd-interworking** option would be added to the core **sip-interface**.

To enable the LMSD Interworking option:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing configuration, you will need to select and edit it.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **options**—Set the options parameter by typing options, a Space, the option name **lmsd-interworking** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-interface)# options +lmsd-interworking
```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save your work.

Ensuring Telephone Event Negotiation within Delayed Media and Conflicting Configurations

In some call flows, the Oracle Communications Session Border Controller (SBC) may stop including telephone events (101) in SIP responses sent to the calling UAC, including 180 and 200 OK messages. These cases typically include early or late media and additional configuration on the interface. You can configure the **sip-interface** with the **add-sdp-baseBehavior** option to ensure telephone event support for these cases without having to change other configuration.

A common, applicable call flow involves a delayed-offer INVITE, where the SDP is not present in the initial INVITE request from the calling UAC. In these cases, the configuration of the **sip-interface** facing the caller typically includes:

- **add-sdp-invite** within a related profile,
- **rfc2833-mode** set to **transparent**, and possibly
- transcoding configuration.

The SBC adds SDP based on the **add-sdp-invite** configuration. It begins call setup and subsequently receives 180 and 200 OK messages from the UAS that include a media offering, such as 0, 101. The conflicting configuration causes the SBC to not forward the 101 to the caller.

To mitigate against this behavior, configure the caller's **sip-interface** with the **add-sdp-baseBehavior** option.

```
ORACLE(sip-config)# options + add-sdp-baseBehavior
```

Accommodating m=line Omissions During Call Setup

The Oracle Communications Session Border Controller (SBC) can accommodate for signaling from non-compliant endpoints that omit mlines during SDP negotiation. You can configure the **fix-missing-mlines** option in **media-manager** to make the SBC allow the call to proceed despite the missing lines.

In SDP, an M line specifies the media endpoints are negotiating for a call. To complete the negotiation, the answer should contain exactly the same number of m= lines as the offer, as specified by RFC 3264. This allows for streams to be matched up based on their order. For example, if an SDP offer contains two m= lines, the answer should also contain two m= lines.

When processing an answer the SBC normally checks the offer/answer pair to ensure the same number of media descriptors are present in the offer and answer SDPs. To work around

the non-compliant endpoint, you can enable the **fix-missing-mlines** option under the **media-manager** configuration. The **fix-missing-mlines** option removes this check.

The syntax for enabling the **fix-missing-mlines** option in **media-manager** follows.

```
ORACLE(media-manager)# options + fix-missing-mlines=yes
```

SIP re-INVITE Suppression

Some of the Interactive Voice Response (IVR) systems that support SIP frequently change the media transport address (IP address and/or port number) when switching between voice menus and/or prompts by sending a re-INVITE.

Often, no other parameters or properties of the session are changed in these re-INVITEs. The frequent re-INVITEs can create performance and capacity problems in other systems along the path of the IVR system and the caller's User Agent.

You can configure the **suppress-reinvite** option on your Oracle Communications Session Border Controller, allowing it to store the previous INVITE and its 200-OK response. Having this information allows the system to reply locally when a re-INVITE that changes only the media transport addresses is recieved.

SIP re-INVITE Suppression Configuration

To enable SIP re-INVITE Suppression:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **suppress-reinvite** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +suppress-reinvite
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Ensuring Compliant SDP Management for P-Early Media Call Flows

In some call flows, the Oracle Communications Session Border Controller (SBC) erroneously inserts SDP into messaging that was already set up for P-Early Media (PEM), causing unexpected media behavior. You can configure the **sip-config** with the **strip-restored-sdp** option to prevent this insertion under certain conditions and avoid subsequent signaling conflicts.

The SBC complies with RFC 5009 to support PEM management of media for all modes and directions. The SBC also includes a function wherein it stores SDP from early dialog messages so it can insert that SDP into subsequent signaling. It does this to maintain continuity for those sessions' media. Within some PEM sequences, however, the SBC updates the media flow with directions from the p-early-media header even without SDP in the message.

Problem behaviors avoided by configuring the **sip-config** with the **strip-restored-sdp** option include:

- If an incoming response includes a 18x and has a PEM header with the valid attribute **P-Early-Media:inactive**, even if there is no SDP body, the SBC default behavior restores the previously received SDP and updates the media flow based on the value of the restored PEM header causing conflicts with the new PEM header.
- If a final response to an INVITE arrives without SDP, and if the state of early media is not **sendrecv** due to a of previously received PEM header, the SBC default behavior creates a 1-way MBCD flow by restoring the previously received SDP, resulting in 1-way audio.

For these examples, the SBC needs to remove the restored SDP before it forwards the message. To remove this SDP in the cases described above, configure the **sip-config** with the **strip-restored-sdp** option using the following syntax.

```
ORACLE(sip-config)# options +strip-restored-sdp
```

The **strip-restored-sdp** option is enabled by default.

To disable this option:

```
ORACLE(sip-config)# options +strip-restored-sdp=no
```

Be careful to consider all the consequences of this configuration prior to deployment as it generates a global change.