THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOSS PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2019 Cisco Systems, Inc. All rights reserved.
CHAPTER 9

Call Admission Control 63

Configuring CAC Based on Total Calls, CPU or Memory 63
Configuring CAC Based on Call Spike Detection 64
Configuring CAC Based on Maximum Calls per Destination 66
Bandwidth-Based Call Admission Control 67

Finding Feature Information 67
Restrictions for Bandwidth-Based Call Admission Control 67
Information About Bandwidth-Based Call Admission Control 68

Maximum Bandwidth Calculation 68
Bandwidth Tables 68

How to Configure Bandwidth-Based Call Admission Control 70

Configuring Bandwidth-Based Call Admission Control at the Interface Level 70
Configuring Bandwidth-Based Call Admission Control at the Dial Peer Level 71
Configuring the Bandwidth-Based Call Admission Control SIP Error Response Code Mapping 72

Verifying Bandwidth-Based Call Admission Control 74
Troubleshooting Tips 76

Configuration Examples for Bandwidth-Based Call Admission Control 76

Example: Configuring Bandwidth-Based Call Admission Control at the Interface Level 76
Example: Configuring Bandwidth-Based Call Admission Control at the Dial Peer Level 77
Example: Configuring the Bandwidth-Based Call Admission Control SIP Error Response Code Mapping at the Global Level 77
Example: Configuring the Bandwidth-Based Call Admission Control SIP Error Response Code Mapping at the Dial Peer Level 77

Feature Information for Bandwidth-Based Call Admission Control 78

CHAPTER 10

Basic SIP Configuration 79

Prerequisites for Basic SIP Configuration 79
Restrictions for Basic SIP Configuration 79
Information About Basic SIP Configuration 80
Contents

SIP Register Support 80
SIP Redirect Processing Enhancement 80
Sending SIP 300 Multiple Choice Messages 81
How to Perform Basic SIP Configuration 81
  Configuring SIP VoIP Services on a Cisco Gateway 82
    Shut Down or Enable VoIP Service on Cisco Gateways 82
    Shut Down or Enable VoIP Submodes on Cisco Gateways 82
  Configuring SIP Register Support 83
  Configuring SIP Redirect Processing Enhancement 85
    Configure Call-Redirect Processing Enhancement 85
  Configuring SIP 300 Multiple Choice Messages 88
    Configuring Sending of SIP 300 Multiple Choice Messages 88
  Configuring SIP Implementation Enhancements 89
    Interaction with Forking Proxies 89
    SIP Intra-Gateway Hairpinning 90
  Verifying SIP Gateway Status 91
  General Troubleshooting Tips 95
Configuration Examples for Basic SIP Configuration 97
  SIP Register Support Example 97
  SIP Redirect Processing Enhancement Examples 99
  SIP 300 Multiple Choice Messages Example 103
Toll Fraud Prevention 104

CHAPTER 11  SIP Binding 107
  Feature Information for SIP Binding 107
  Information About SIP Binding 108
    Benefits of SIP Binding 108
    Source Address 109
    Voice Media Stream Processing 112
  Configuring SIP Binding 114
  Verifying SIP Binding 116

CHAPTER 12  Media Path 123
  Feature Information for Media Path 123
CHAPTER 13

SIP Profiles 131

Feature Information for SIP Profiles 131

Information About SIP Profiles 132

Important Characteristics of SIP Profiles 133

Restrictions for SIP Profiles 134

How to Configure SIP Profiles 135

Configuring a SIP Profile to Manipulate SIP Request or Response Headers 135

Configuring SIP Profiles for Copying Unsupported SDP Headers 137

Example: Configuring SIP Profile Rules (Attribute Passing) 138

Example: Configuring SIP Profile Rules (Parameter Passing) 139

Example: Configuration to Remove an Attribute 139

Configuring SIP Profile Using Rule Tag 139

Configuring a SIP Profile for Non-standard SIP Header 141

Upgrading or Downgrading SIP Profile Configurations 142

Configuring a SIP Profile as an Outbound Profile 143

Configuring a SIP Profile as an Inbound Profile 144

Verifying SIP Profiles 146

Troubleshooting SIP Profiles 146

Examples: Adding, Modifying, Removing SIP Profiles 147

Example: Adding a SIP, SDP, or Peer Header 147

Example: Modifying a SIP, SDP, or Peer Header 149

Example: Remove a SIP, SDP, or Peer Header 151

Example: Inserting SIP Profile Rules 152

Example: Upgrading and Downgrading SIP Profiles automatically 153
Finding Feature Information  191
Prerequisites for VoIP for IPv6  191
Restrictions for Implementing VoIP for IPv6  192
Information About VoIP for IPv6  193
  SIP Features Supported on IPv6  193
  SIP Voice Gateways in VoIPv6  194
  VoIPv6 Support on Cisco UBE  195
How to Configure VoIP for IPv6  199
  Configuring VoIP for IPv6  199
    Shutting Down or Enabling VoIPv6 Service on Cisco Gateways  200
    Shutting Down or Enabling VoIPv6 Submodes on Cisco Gateways  201
    Configuring the Protocol Mode of the SIP Stack  201
    Verifying SIP Gateway Status  203
RTCP Pass-Through  205
  Configuring IPv6 Support for Cisco UBE  205
  Verifying RTP Pass-Through  206
Configuring the Source IPv6 Address of Signaling and Media Packets  207
Configuring the SIP Server  208
Configuring the Session Target  209
Configuring SIP Register Support  210
Configuring Outbound Proxy Server Globally on a SIP Gateway  212
Configuring UDP Checksum  213
Configuring IP Toll Fraud  214
Configuring the RTP Port Range for an Interface  215
Configuring Message Waiting Indicator Server Address  216
Configuring Voice Ports  217
Configuring Cisco UBE Mid-call Re-INVITE Consumption  218
  Configuring Passthrough of Mid-call Signalling  218
  Configuring Passthrough SIP Messages at Dial Peer Level  219
Configuring H.323 IPv4-to-SIPv6 Connections in a Cisco UBE  220
Configuration Examples for VoIP over IPv6  222
  Example: Configuring the SIP Trunk  222
Troubleshooting Tips for VoIP for IPv6  223
Verifying and Troubleshooting Tips  223
CHAPTER 17

Configurable SIP Parameters via DHCP 247
Finding Feature Information 247
Prerequisites for Configurable SIP Parameters via DHCP 247
Restrictions for Configurable SIP Parameters via DHCP 248
Information About Configurable SIP Parameters via DHCP 248
How to Configure SIP Parameters via DHCP 252
  Configuring the DHCP Client 252
  Configuring the DHCP Client Example 253
  Enabling the SIP Configuration 254
  Enabling the SIP Configuration Example 255
  Troubleshooting Tips 255
  Configuring a SIP Outbound Proxy Server 256
  Configuring a SIP Outbound Proxy Server in Voice Service VoIP Configuration Mode 256
  Configuring a SIP Outbound Proxy Server in Voice Service VoIP Configuration Mode Example 257
  Configuring a SIP Outbound Proxy Server and Session Target in Dial Peer Configuration Mode 257
  Configuring a SIP Outbound Proxy Server in Dial Peer Configuration Mode Example 258
Feature Information for Configurable SIP Parameters via DHCP 259

PART II

Dial Peer Enhancements 261

CHAPTER 18

Matching Inbound Dial Peers by URI 263
  Feature Information for Matching Inbound Dial Peers by URI 263
  Configuring an Inbound Dial Peer to Match on URI 264
  Examples for Configuring an Inbound Dial Peer to Match on a URI 266
CHAPTER 19  URI-Based Dialing Enhancements  269
Finding Feature Information  269
Feature Information for URI-Based Dialing Enhancements  269
Information About URI-Based Dialing Enhancements  270
Call Flows for URI-Based Dialing Enhancements  271
How to Configure URI-Based Dialing Enhancements  273
Configuring Pass Through of SIP URI Headers  273
Configuring Pass Though of Request URI and To Header URI (Global Level)  273
Configuring Pass Though of Request URI and To Header URI (Dial Peer Level)  274
Configuring Pass Through of 302 Contact Header  276
Configuring Pass Through of 302 Contact Header (Global Level)  276
Configuring Pass Through of 302 Contact Header (Dial Peer Level)  277
Deriving of Session Target from URI  278
Configuration Examples for URI-Based Dialing Enhancements  280
Example: Configuring Pass Though of Request URI and To Header URI  280
Example: Configuring Pass Though of Request URI and To Header URI (Global Level)  280
Example: Configuring Pass Though of Request URI and To Header URI (Dial Peer Level)  280
Example: Configuring Pass Through of 302 Contact Header  281
Example: Configuring Pass Through of 302 Contact Header (Global Level)  281
Example: Configuring Pass Through of 302 Contact Header (Dial Peer Level)  281
Example: Deriving Session Target from URI  281
Additional References for URI-Based Dialing Enhancements  282

CHAPTER 20  Multiple Pattern Support on a Voice Dial Peer  283
Feature Information for Multiple Pattern Support on a Voice Dial Peer  283
Restrictions for Multiple Pattern Support on a Voice Dial Peer  284
Information About Multiple Pattern Support on a Voice Dial Peer  284
Configuring Multiple Pattern Support on a Voice Dial Peer  284
Verifying Multiple Pattern Support on a Voice Dial Peer  286
Configuration Examples for Multiple Pattern Support on a Voice Dial Peer  288

CHAPTER 21  Outbound Dial-Peer Group as an Inbound Dial-Peer Destination  289
Feature Information for Outbound Dial-Peer Group as an Inbound Dial-Peer Destination  289
Restrictions 290

Information About Outbound Dial-Peer Group as an Inbound Dial-Peer Destination 290

Configuring Outbound Dial-Peer Group as an Inbound Dial-Peer Destination 291

Verifying Outbound Dial-Peer Groups as an Inbound Dial-Peer Destination 293

Troubleshooting Tips 294

Configuration Examples for Outbound Dial Peer Group as an Inbound Dial-Peer Destination 295

CHAPTER 22

Inbound Leg Headers for Outbound Dial-Peer Matching 299

Feature Information for Inbound Leg Headers for Outbound Dial-Peer Matching 299

Prerequisites for Inbound Leg Headers for Outbound Dial-Peer Matching 300

Restrictions for Inbound Leg Headers for Outbound Dial-Peer Matching 300

Information About Inbound Leg Headers for Outbound Dial-Peer Matching 300

Configuring Inbound Leg Headers for Outbound Dial-Peer Matching 301

Verifying Inbound Leg Headers for Outbound Dial-Peer Matching 304

Configuration Example: Inbound Leg Headers for Outbound Dial-Peer Matching 306

CHAPTER 23

Server Groups in Outbound Dial Peers 309

Feature Information for Configuring Server Groups in Outbound Dial Peers 309

Information About Server Groups in Outbound Dial Peers 310

How to Configure Server Groups in Outbound Dial Peers 310

Configuring Server Groups in Outbound Dial Peers 310

Verifying Server Groups in Outbound Dial Peers 312

Configuration Examples for Server Groups in Outbound Dial Peers 313

CHAPTER 24

Domain-Based Routing Support on the Cisco UBE 317

Feature Information for Domain-Based Routing Support on the Cisco UBE 317

Restrictions for Domain-Based Routing Support on the Cisco UBE 318

Information About Domain-Based Routing Support on the Cisco UBE 318

How to Configure Domain-Based Routing Support on the Cisco UBE 319

Configuring Domain-Based Routing at Global Level 319

Configuring Domain-Based Routing at Dial Peer Level 320

Verifying and Troubleshooting Domain-Based Routing Support on the Cisco UBE 321

Configuration Examples for Domain-Based Routing Support on the Cisco UBE 324

Example Configuring Domain-Based Routing Support on the Cisco UBE 324
Feature Information for Video Recording - Additional Configurations 493
Information About Additional Configurations for Video Recording 494
  Full Intra-Frame Request 494
How to Configure Additional Configurations for Video Recording 494
  Enabling FIR for Video Calls (Using RTCP of SIP INFO) 494
  Configuring H.264 Packetization Mode 495
  Monitoring Reference files or Intra Frames 496
Verifying Additional Configurations for Video Recording 497

CHAPTER 38
Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording 499
  Feature Information for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording 499
  Restrictions for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording 500
  Information About Third-Party GUID Capture for Correlation Between Calls and SIP-based recording 500
  How to Capture Third-Party GUID for Correlation Between Calls and SIP-based Recording 500
  Verifying Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording 503
  Configuration Examples for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording 504

CHAPTER 39
Cisco Unified Communications Gateway Services--Extended Media Forking 507
  Feature Information for Cisco Unified Communications Gateway Services—Extended Media Forking 507
  Restrictions for Unified Communications Gateway Services—Extended Media Forking 508
  Information About Cisco Unified Communications Gateway Services 508
    Extended Media Forking (XMF) Provider and XMF Connection 508
    XMF Call-Based Media Forking 509
    XMF Connection-Based Media Forking 509
    Cisco UC Gateway Services Media Forking API with Survivability TCL 510
    Media Forking for SRTP Calls 511
      Crypto Tag 511
      Example of SDP Data sent in an SRTP Call 512
      Multiple XMF Applications and Recording Tone 512
      Forking Preservation 514
How to Configure UC Gateway Services 515
  Configuring Cisco Unified Communication IOS Services on the Device 515
  Configuring the XMF Provider 518
  Verifying the UC Gateway Services 519
  Troubleshooting Tips 521
Configuration Examples for UC Gateway Services 522
  Example: Configuring Cisco Unified Communication IOS Services 522
  Example: Configuring the XMF Provider 522
  Example: Configuring UC Gateway Services 522

CUBEB Media Proxy 523

PART IX

CHAPTER 40

CUBEB Media Proxy 525
  Feature Information for CUBEB Media Proxy 525
  Supported Platforms 526
  Restrictions for CUBEB Media Proxy 526
  Information About Multiple Media Forking Using CUBEB Media Proxy 526
    Deployment Scenarios for CUBEB Media Proxy Based Media Forking 526
    Media Forking Topologies for CUBEB Media Proxy 528
      Gateway Forking to CUBEB Media Proxy 528
      Phone BiB Forking to CUBEB Media Proxy 528
  Recording Metadata 529
    Session Identifier 529
      Configuring Support for Session Identifier 530
      Session-ID Handling 530
      Troubleshooting Tips 531
  How to Configure CUBEB Media Proxy for Media Forking 532
    Configuring Outbound Dial-Peers to the Recorders 532
    Configuring CUBEB Media Proxy 533
    Configuring Inbound Dial-Peer from CUCM 535
    Verifying Media Forking Using CUBEB Media Proxy Configuration 536
  Recording State Notification 545
    SIP Info Messages from CUBEB Media Proxy to CUCM 545
    SIP Info Message Sent During the Initial Call 546
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Info Message Sent During the Initial Call (All the Recorders as Optional)</td>
<td>546</td>
</tr>
<tr>
<td>SIP Info Message Sent During the Initial Call (One Recorder as Mandatory and Remaining as Optional)</td>
<td>547</td>
</tr>
<tr>
<td>Support for Mid-Call Features</td>
<td>548</td>
</tr>
<tr>
<td>Secure Recording of Secure Calls</td>
<td>551</td>
</tr>
<tr>
<td>Support for High Availability (HA)</td>
<td>552</td>
</tr>
<tr>
<td>Media Latch</td>
<td>553</td>
</tr>
<tr>
<td><strong>PART X</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SIP Header Manipulation</strong></td>
<td>555</td>
</tr>
<tr>
<td><strong>CHAPTER 41</strong></td>
<td></td>
</tr>
<tr>
<td>Passing Headers Unsupported by CUBE</td>
<td>557</td>
</tr>
<tr>
<td>Feature Information for Copying with SIP Profiles</td>
<td>557</td>
</tr>
<tr>
<td>Example: Passing a Header Not Supported by CUBE</td>
<td>557</td>
</tr>
<tr>
<td><strong>CHAPTER 42</strong></td>
<td></td>
</tr>
<tr>
<td>Copying SIP Headers</td>
<td>559</td>
</tr>
<tr>
<td>Feature Information for Copying with SIP Profiles</td>
<td>559</td>
</tr>
<tr>
<td>How to Copy SIP Header Fields to Another</td>
<td>560</td>
</tr>
<tr>
<td>Copying From an Incoming Header and Modifying an Outgoing Header</td>
<td>560</td>
</tr>
<tr>
<td>Copying From One Outgoing Header to Another</td>
<td>562</td>
</tr>
<tr>
<td>Example: Copying the To Header into the SIP-Req-URI</td>
<td>563</td>
</tr>
<tr>
<td><strong>CHAPTER 43</strong></td>
<td></td>
</tr>
<tr>
<td>Manipulating SIP Status-Line Header of SIP Responses</td>
<td>567</td>
</tr>
<tr>
<td>Feature Information for Manipulating SIP Responses</td>
<td>567</td>
</tr>
<tr>
<td>Copying Incoming SIP Response Status Line to Outgoing SIP Response</td>
<td>568</td>
</tr>
<tr>
<td>Modifying Status-Line Header of Outgoing SIP Response with User Defined Values</td>
<td>571</td>
</tr>
<tr>
<td><strong>PART XI</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Payload Type Interoperability</strong></td>
<td>573</td>
</tr>
<tr>
<td><strong>CHAPTER 44</strong></td>
<td></td>
</tr>
<tr>
<td>Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls</td>
<td>575</td>
</tr>
<tr>
<td>Feature Information for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls</td>
<td>575</td>
</tr>
<tr>
<td>Restrictions for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls</td>
<td>576</td>
</tr>
<tr>
<td>Symmetric and Asymmetric Calls</td>
<td>576</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>High Availability Checkpointing Support for Asymmetric Payload</td>
<td>577</td>
</tr>
<tr>
<td>How to Configure Dynamic Payload Type Passthrough for DTMF and Codec</td>
<td>578</td>
</tr>
<tr>
<td>Packets for SIP-to-SIP Calls</td>
<td></td>
</tr>
<tr>
<td>Configuring Dynamic Payload Type Passthrough at the Global Level</td>
<td>578</td>
</tr>
<tr>
<td>Configuring Dynamic Payload Type Passthrough for a Dial Peer</td>
<td>579</td>
</tr>
<tr>
<td>Verifying Dynamic Payload Interworking for DTMF and Codec Packets</td>
<td>580</td>
</tr>
<tr>
<td>Support</td>
<td></td>
</tr>
<tr>
<td>Troubleshooting Tips</td>
<td>580</td>
</tr>
<tr>
<td>Configuration Examples for Asymmetric Payload Interworking</td>
<td>581</td>
</tr>
<tr>
<td>Example: Asymmetric Payload Interworking—Passthrough Configuration</td>
<td>581</td>
</tr>
<tr>
<td>Example: Asymmetric Payload Interworking—Interworking Configuration</td>
<td>582</td>
</tr>
</tbody>
</table>

**PART XII**

**Protocol Interworking** 583

**CHAPTER 45**

**Delayed-Offer to Early-Offer** 585

- Feature Information for Delayed-Offer to Early-Offer 585
- Prerequisites for Delayed-Offer to Early-Offer 586
- Restrictions for Delayed-Offer to Early-Offer Media Flow-Around 586
- Delayed-Offer to Early-Offer in Media Flow-Around Calls 586
  - Configuring Delayed Offer to Early Offer 587
  - Configuring Delayed Offer to Early Offer for Video Calls 588
  - Configuring Delayed Offer to Early Offer Medial Flow-Around 589
- MidCall Renegotiation Support for Delayed-Offer to Early-Offer Calls 590
  - Restrictions for MidCall Renegotiation Support for DO-EO Calls 591
  - Configuring Mid Call Renegotiation Support for Delayed-Offer to Early-Offer Calls 591
- High-Density Transcoding Calls in Delayed-Offer to Early-Offer 592
  - Restrictions for High-Density Transcoding DO-EO Calls 593
  - Configuring High-Density Transcoding 593

**CHAPTER 46**

**H323-to-SIP Interworking on CUBE** 595

- Prerequisites 595
- Restrictions 595
- H323-to-SIP Basic Call Interworking 596
- H323-to-SIP Supplementary Features Interworking 598
- H.323-to-SIP Codec Progress Indicator Interworking for Media Cut-Through 599
Information About SRTP-SRTP Interworking  624
  Supplementary Services Support  625
How to Configure SRTP-SRTP Interworking  626
  Configuring SRTP  626
    Configuring Cipher Suite Preference (optional)  628
    Applying Crypto Suite Selection Preference (optional)  629
    Enabling SRTP Fallback  631
Configuration Examples  634
  Example: Configuring SRTP-SRTP Interworking  634
  Example: Changing the Cipher-Suite Preference  636

CHAPTER 50  SRTP-RTP Interworking  639
  Feature Information for SRTP-RTP Interworking  639
  Prerequisites for SRTP-RTP Interworking  640
  Restrictions for SRTP-RTP Interworking  640
  Information About SRTP-RTP Interworking  640
    Support for SRTP-RTP Interworking  640
      Using SRTP-RTP Chain for Interworking Between AES_CM_128_HMAC_SHA1_32 and
      AES_CM_128_HMAC_SHA1_80 Crypto Suites  642
    Supplementary Services Support  643
  How to Configure Support for SRTP-RTP Interworking  644
    Configuring SRTP-RTP Interworking Support  644
    Configuring Crypto Authentication  646
    Enabling SRTP Fallback  648
      Troubleshooting Tips  650
    Verifying SRTP-RTP Supplementary Services Support  650
  Configuration Examples for SRTP-RTP Interworking  651
    Example: SRTP-RTP Interworking  651
    Example: Configuring Crypto Authentication  652
      Example: Configuring Crypto Authentication (Dial Peer Level)  652
      Example: Configuring Crypto Authentication (Global Level)  652

CHAPTER 51  SRTP-SRTP Pass-Through  653
  Feature Information for Support of SRTP-SRTP Pass-Through Calls  653
Information About SRTP-SRTP Pass-Through 654
Pass-Through of Unsupported Crypto Suites 654
Configure Pass-Through of Unsupported Crypto Suites for a Specific Dial Peer 655
Configure Pass-Through of Unsupported Crypto Suites Globally 657
Configuration Examples for SRTP-SRTP Pass-Through 658

PART XIV

High Availability 661

CHAPTER 52
High Availability Overview 663
Information About High Availability 663
Inbox versus Box-to-Box Redundancy 663
Route Processor Redundancy 663
Stateful Switchover 664
Nonstop Forwarding 664
HA Checkpointing 664
CUBE High Availability Options 664
Hot Standby Routing Protocol 664
Redundancy Group Infrastructure 665
Considerations for Choosing an HA Configuration 665
Restrictions for CUBE High Availability 665

CHAPTER 53
DSP High Availability Support 667
Feature Information for DSP High Availability Support on CUBE 667
Prerequisites for DSP High Availability 668
Features Supported with DSP High Availability 668
Restrictions for DSP High Availability 668
Troubleshooting DSP HA Support on CUBE 669
How to Configure High Availability 669
Configuration Examples for DSP HA 669

CHAPTER 54
Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices 671
Feature Information for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices 671
Prerequisites for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices 672
Restrictions for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices 673
Information About Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices 673
Call Escalation with Stateful Switchover 674
Call De-escalation with Stateful Switchover 674
Media Forking with High Availability 675
High Availability Protected Mode and Box-to-Box Redundancy for ASR 675
Support for Box-to-Box High Availability with Virtual IP Addresses 676
Monitoring Call Escalation and De-escalation with Stateful Switchover 676
Monitoring Media Forking with High Availability 678
Verifying the High Availability Protected Mode 680
Support for REFER and BYE/Also after Stateful Switch-Over 681
Troubleshooting Tips 681
Example: Configuring the Interfaces for ISR-G2 Devices 683
Example: Configuring the Interfaces for ASR Devices 683
Example: Configuring SIP Binding 683

CHAPTER 55

CVP Survivability TCL support with High Availability 685
Feature Information for CVP Survivability TCL support with High Availability 685
Prerequisites 686
Restrictions 686
Recommendations 686
CVP Survivability TCL support with High Availability 686
Configuring CVP Survivability TCL support with High Availability 686

PART XV

ICE-Lite Support on CUBE 687

CHAPTER 56

ICE-Lite Support on CUBE 689
Feature Information for ICE-Lite Support on CUBE 689
Restrictions for ICE-lite Support on CUBE 690
Information About ICE-Lite Support on CUBE 690
Characteristics 690
ICE Candidate 691
ICE Lite 691
High Availability Support with ICE 691
How to Configure ICE-Lite Support on CUBE  692
  Configuring ICE on the CUBE  692
  Verifying ICE-Lite on the CUBE (Success Flow Calls)  693
  ICE-Lite on CUBE (Error Flow Calls)  696
  Troubleshooting ICE-Lite Support on CUBE  701

Additional References  701

PART XVI

SIP Protocol Handling  703

CHAPTER 57

Mid-call Signaling Consumption  705
  Feature Information for Mid-call Signaling  705
  Prerequisites  706
  Mid-call Signaling Passthrough - Media Change  707
    Restrictions for Mid-Call Signaling Passthrough - Media Change  707
  Behavior of Mid-call Re-INVITE Consumption  707
  Configuring Passthrough of Mid-call Signalling  708
  Example Configuring Passthrough SIP Messages at Dial Peer Level  709
  Example Configuring Passthrough SIP Messages at the Global Level  710
  Mid-call Signaling Block  710
    Restrictions for Mid-Call Signaling Block  710
    Blocking Mid-Call Signaling  711
  Example Blocking SIP Messages at Dial Peer Level  712
  Example: Blocking SIP Messages at the Global Level  712
  Mid Call Codec Preservation  712
    Configuring Mid Call Codec Preservation  712
    Example: Configuring Mid Call Codec Preservation at the Dial Peer Level  713
    Example: Configuring Mid Call Codec Preservation at the Global Level  714

CHAPTER 58

Early Dialog UPDATE Block  715
  Feature Information for Early Dialog UPDATE Block  715
  Prerequisites  716
  Restrictions  716
  Information about Early Dialog UPDATE Block  716
    Important Characteristics of Early Dialog UPDATE Block  716
Restrictions 777
Configuring Cisco UBE SIP Registration Proxy 777
   Enabling Local SIP Registrar 777
   Configuring SIP Registration Proxy at the Global Level 778
   Configuring SIP Registration Proxy at the Tenant Level 780
   Configuring SIP Registration Proxy at the Dial Peer Level 781
   Configuring Registration Overload Protection Functionality 783
   Configuring Cisco UBE to Route a Call to the Registrar Endpoint 784
   Verifying the SIP Registration on Cisco UBE 785
Configuration Example—Cisco UBE SIP Registration Proxy 786
Feature Information for Cisco UBE SIP Registration Proxy 786

CHAPTER 66 787
Survivability for Hosted and Cloud Services
   Information About Survivability for Hosted and Cloud Services 787
   Advantages of Using Cisco UBE Survivability Feature 787
      Local Fallback 787
      Registration Synchronization 788
   Registration Through Alias Mapping 788
      Cisco UBE when WAN is UP 789
      Cisco UBE Survivability When WAN Is Down 790
   How to Configure Survivability for Hosted and Cloud Services 792
      Configuring Local Fallback or Registration Synchronization Globally 792
      Configuring Local Fallback or Registration Synchronization at the Tenant Level 793
      Configuring Local Fallback or Registration Synchronization on a Dial Peer 794
      Configuring Survivability for Phones Sending Single Register Request 795
      Configuring OPTIONS Ping 796
      Configuring Registration Timer 797
      Configuring the REGISTER Message Throttling in Cisco UBE 798
      Configuring the Class of Restrictions (COR) List 799
      Verifying Survivability for Hosted and Cloud Services 801
   Configuration Examples—Survivability for Hosted and Cloud Services 803
      Example: Configuring Local Fallback Globally 803
      Example: Configuring Local Fallback at the Tenant Level 804
      Example: Configuring Local Fallback on a Dial Peer 804
# Voice Quality in CUBE

## CHAPTER 69

**CUBE Call Quality Statistics Enhancement**
- Feature Information for Call Quality Statistics Enhancement: 851
- Restrictions for Call Quality Statistics Enhancement: 852
- Information About Call Quality Statistics Enhancement: 852
- How to Configure Call Quality Parameters: 853
  - Configuring Call Quality Criteria Parameters: 853
  - Troubleshooting Call Quality Statistics: 854
- Configuration Example for Call Quality Statistics: 855

## CHAPTER 70

**Voice Quality Monitoring**
- Feature Information for Voice Quality Monitoring: 857
- Prerequisites for Voice Quality Monitoring: 858
- Restrictions for Voice Quality Monitoring and Voice Quality Statistics: 858
- Information About Voice Quality Monitoring: 859
  - VQM Metrics: 859
- How to Configure Voice Quality Monitoring: 860
  - Enabling Media Statistics Globally: 860
  - Verifying Voice Quality Monitoring: 861
  - Troubleshooting Tips: 863
- Configuration Examples for Voice Quality Monitoring: 863
  - Example: Configuring Media Statistics Globally: 863
  - Example: CDR Enabled MOS Output: 863
CHAPTER 1

Read Me First

Important Information about Cisco IOS XE 16
Effective Cisco IOS XE Release 3.7.0E for Catalyst Switching and Cisco IOS XE Release 3.17S (for Access and Edge Routing) the two releases evolve (merge) into a single version of converged release—the Cisco IOS XE 16—providing one release covering the extensive range of access and edge products in the Switching and Routing portfolio.

Feature Information
Use Cisco Feature Navigator to find information about feature support, platform support, and Cisco software image support. An account on Cisco.com is not required.

Related References
• Cisco IOS Command References, All Releases

Obtaining Documentation and Submitting a Service Request
• To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
• To get the business impact you’re looking for with the technologies that matter, visit Cisco Services.
• To submit a service request, visit Cisco Support.
• To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
• To obtain general networking, training, and certification titles, visit Cisco Press.
• To find warranty information for a specific product or product family, access Cisco Warranty Finder.
New and Changed Information

For detailed information on CUBE features supported on Cisco IOS Releases, Cisco IOS XE 3S Releases, and Cisco IOS XE Denali 16 Releases, refer to CUBE Cisco IOS Feature Roadmap, CUBE Cisco IOS-XE Feature Roadmap, and CUBE Cisco IOS XE 16 Feature Roadmap respectively.

<table>
<thead>
<tr>
<th>Description</th>
<th>Documented at</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Cisco 1000 Series Integrated Services Routers (ISR)</td>
<td>Supported Platforms, on page 5</td>
</tr>
<tr>
<td>The output of the following commands are enhanced to display SRTP ROC information:</td>
<td>Cisco IOS Voice Command Reference - S commands</td>
</tr>
<tr>
<td>• show voip fpi calls</td>
<td></td>
</tr>
<tr>
<td>• show voip fpi stats</td>
<td></td>
</tr>
<tr>
<td>• show voip rtp connections</td>
<td></td>
</tr>
<tr>
<td>Hosted and Cloud Services Support</td>
<td>Hosted and Cloud Services Delivery with Cisco UBE, on page 767</td>
</tr>
<tr>
<td>Support for Common Criteria and FIPS Compliance</td>
<td>Common Criteria (CC) and The Federal Information Processing Standards (FIPS) Compliance, on page 895</td>
</tr>
</tbody>
</table>
CHAPTER 3

Supported Platforms

Cisco Unified Border Element is supported on various platforms running on Cisco IOS Software Releases and Cisco IOS XE Software Releases.

For information on migrating from existing Cisco IOS XE 3S releases to the Cisco IOS XE Denali 16.3 release, see Cisco IOS XE Denali 16.3 Migration Guide for Access and Edge Routers

The following table provides information on Cisco router platform support for Cisco Unified Border Element:

<table>
<thead>
<tr>
<th>Cisco Router Platforms</th>
<th>Cisco Router Models</th>
<th>Cisco IOS Software Releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco Integrated</td>
<td>Cisco 2900 Series Integrated Services Routers</td>
<td>Cisco IOS 12 M and T Cisco IOS 15 M and T</td>
</tr>
<tr>
<td>Services Generation</td>
<td>Cisco 3900 Series Integrated Services Routers</td>
<td></td>
</tr>
<tr>
<td>2 Routers (ISR G2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cisco 4000 Series</td>
<td>Cisco 4321 Integrated Services Routers</td>
<td>Cisco IOS XE 3S</td>
</tr>
<tr>
<td>Integrated Services</td>
<td>Cisco 4331 Integrated Services Routers</td>
<td></td>
</tr>
<tr>
<td>Routers (ISR G3)</td>
<td>Cisco 4351 Integrated Services Routers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cisco 4431 Integrated Services Routers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cisco 4451 Integrated Services Routers</td>
<td></td>
</tr>
</tbody>
</table>

1 Denali 16.3 onwards
<table>
<thead>
<tr>
<th>Cisco Router Platforms</th>
<th>Cisco Router Models</th>
<th>Cisco IOS Software Releases</th>
</tr>
</thead>
</table>
| Cisco 1000 Series Integrated Services Routers (ISR) (cont.) | Cisco 1111-8PLTE Series Integrated Services Routers  
Cisco 1111-4PLTE Series Integrated Services Routers  
Cisco 1111-8PW Series Integrated Services Routers  
Cisco 1112-8P Integrated Services Routers  
Cisco 1112-8PWE Integrated Services Routers  
Cisco 1112-8PLTE Series Integrated Services Routers  
Cisco 1113-8P Integrated Services Routers  
Cisco 1113-8PM Integrated Services Routers  
Cisco 1113-8PW Series Integrated Services Routers  
Cisco 1113-8PLTE Series Integrated Services Routers  
Cisco 1113-8PMLTEEA Integrated Services Routers | Cisco IOS XE Gibraltar Release 16.12.1a onwards |
<table>
<thead>
<tr>
<th>Cisco Router Platforms</th>
<th>Cisco Router Models</th>
<th>Cisco IOS Software Releases</th>
</tr>
</thead>
</table>
| Cisco 1000 Series Integrated Services Routers (ISR) (cont.) | Cisco 1113-8PMWE Integrated Services Routers  
Cisco 1116-4P Integrated Services Routers  
Cisco 1116-4PWE Integrated Services Routers  
Cisco 1116-4PLTE Series Integrated Services Routers  
Cisco 1117-4P Integrated Services Routers  
Cisco 1117-4PM Integrated Services Routers  
Cisco 1117-4PW Series Integrated Services Routers  
Cisco 1117-4PMWE Integrated Services Routers  
Cisco 1117-4PLTE Series Integrated Services Routers  
Cisco 1117-4PMLTE Series Integrated Services Routers  
Cisco 1118-8P Integrated Services Routers | Cisco IOS XE Gibraltar Release 16.12.1a onwards |
| Cisco Aggregated Services Routers (ASR) | Cisco 1001-X Aggregated Services Routers  
Cisco 1002-X Aggregated Services Routers  
Cisco 1004 Aggregated Services Routers with RP2  
Cisco 1006 Aggregated Services Routers with RP2 | Cisco IOS XE 3S  
Cisco IOS XE Denali 16.3.1 onwards ² |
| Cisco Cloud Services Routers (CSR) | Cisco Cloud Services Router 1000V series | Cisco IOS XE 3.15 onwards  
Cisco IOS XE Denali 16.3.1 onwards |

1 All CUBE features from release 11.5.0 (Cisco IOS XE Release 3.17) and features introduced in CUBE 11.5.1 on Cisco Integrated Services Generation 2 Routers (ISR G2) are included in CUBE release 11.5.2 for the Cisco IOS XE based platforms from Cisco IOS XE Denali 16.3.1 onwards.

2 Cisco IOS XE 16 requires a minimum of ASR1001-X, 1002-X, 1004/1006 RP2, ESP20 (Embedded Service Processor), and SIP40 (SPA Interface processor).

- Feature Comparison on Supported Platforms, on page 8
Feature Comparison on Supported Platforms

The following table provides high level details of CUBE features supported on different platforms.

For collaboration feature support on Cisco ISR 4000 Series Routers, Cisco IOS XE Release 3.13.1S or later is recommended. Cisco Cloud Services Routers 1000V Series support is available from Cisco IOS XE Release 3.15S onwards.

<table>
<thead>
<tr>
<th>Features</th>
<th>Cisco ASR 1000 Series Routers</th>
<th>Cisco ISR G2 Series Routers</th>
<th>Cisco ISR 4000 Series Routers</th>
<th>Cisco ISR 1000 Series Routers</th>
<th>Cisco CSR 1000V Series Routers</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Availability Implementation</td>
<td>Redundancy Group Infrastructure</td>
<td>Hot Standby Router Protocol (HSRP) Based</td>
<td>Redundancy Group Infrastructure</td>
<td>No</td>
<td>Redundancy Group Infrastructure</td>
</tr>
<tr>
<td>Media Forking</td>
<td>Yes (Cisco IOS XE Release 3.8S onwards)</td>
<td>Yes (Cisco IOS Release 15.2(1)T onwards)</td>
<td>Yes (Cisco IOS XE Release 3.10S onwards)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DSP Card Type</td>
<td>SPA-DSP</td>
<td>PVDM2/PVDM3</td>
<td>PVDM4</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Transcoder registered to CUCM</td>
<td>No</td>
<td>Yes (Exists via SCCP)</td>
<td>Yes (Exists via SCCP - Cisco IOS XE Release 3.11S onwards)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Transcoder—LTI</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Cisco UC Gateway Services API</td>
<td>Yes (Cisco IOS XE Release 3.8S onwards)</td>
<td>Yes (Cisco IOS Release 15.2(2)T onwards)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Noise Reduction &amp; ASP</td>
<td>Yes (Cisco IOS Release 15.2(3)T onwards)</td>
<td>Yes (Cisco IOS Release 15.2(3)T onwards)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Feature Comparison on Supported Platforms

<table>
<thead>
<tr>
<th>Features</th>
<th>Cisco ASR 1000 Series Routers</th>
<th>Cisco ISR G2 Series Routers</th>
<th>Cisco ISR 4000 Series Routers</th>
<th>Cisco IS 1000 Series Routers</th>
<th>Cisco CSR 1000V Series Routers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Progress Analysis</td>
<td>Yes (Cisco IOS XE Release 3.9S onwards; Recommended - Cisco IOS XE Release 3.15S)</td>
<td>Yes (Cisco IOS Release 15.3(2)T onwards; Recommended -Cisco IOS Release 15.5(2)T onwards)</td>
<td>Yes Recommended - Cisco IOS XE Release 3.15S</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRTP-RTP Interworking</td>
<td>Yes - No DSP resources required (Cisco IOS XE Release 3.7S onwards)</td>
<td>Yes - DSP resources required (Cisco IOS Release 12.4(22)YB onwards)</td>
<td>Yes - No DSP resources required Cisco IOS XE Release 3.12S onwards</td>
<td>Yes - No DSP resources required</td>
<td>Yes - No DSP resources required (Cisco IOS XE Release 3.15S onwards)</td>
</tr>
<tr>
<td>CUBE for SP Managed and Hosted Services</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Unified SRST colocation with CUBE</td>
<td>Not supported</td>
<td>SCCP SRST is supported</td>
<td>SIP SRST is not supported</td>
<td>No</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**For more information on Unified SRST and Unified Border Element Co-location, see [Unified SRST and Unified Border Element Co-location](#).**

Co-location of Cisco Unified Border Element - High Availability (HA) with Unified SRST is not supported.
PART I

CUBE Fundamentals and Basic Setup

- Overview of Cisco Unified Border Element, on page 13
- Virtual CUBE, on page 23
- Dial-Peer Matching, on page 31
- DTMF Relay, on page 37
- Introduction to Codecs, on page 51
- Call Admission Control, on page 63
- Basic SIP Configuration, on page 79
- SIP Binding, on page 107
- Media Path, on page 123
- SIP Profiles, on page 131
- SIP Out-of-Dialog OPTIONS Ping Group, on page 157
- Configure TCL IVR Applications, on page 165
- VoIP for IPv6, on page 191
- Configurable SIP Parameters via DHCP, on page 247
Overview of Cisco Unified Border Element

Cisco Unified Border Element (CUBE) bridges voice and video connectivity between two separate VoIP networks. It is similar to a traditional voice gateway, except for the replacement of physical voice trunks with an IP connection. Traditional gateways connect VoIP networks to telephone companies using a circuit-switched connection, such as PRI. The CUBE connects VoIP networks to other VoIP networks and is often used to connect enterprise networks to Internet telephony service providers (ITSPs).

- Information About Cisco Unified Border Element, on page 13
- How to Configure Basic CUBE Features, on page 18

Information About Cisco Unified Border Element

Cisco Unified Border Element (CUBE) can terminate and originate signaling (H.323 and Session Initiation Protocol [SIP]) and media streams (Real-Time Transport Protocol [RTP] and RTP Control Protocol [RTCP]).

CUBE extends the functionality provided by conventional session border controllers (SBCs) in terms of protocol interworking, especially on the enterprise side. As shown in the chart below, the CUBE provides the following additional features:
The CUBE provides a network-to-network interface point for:

- Signaling interworking—H.323 and SIP.
- Media interworking—dual-tone multifrequency (DTMF), fax, modem, and codec transcoding.
- Address and port translations—privacy and topology hiding.
- Billing and call detail record (CDR) normalization.
- Quality-of-service (QoS) and bandwidth management—QoS marking using differentiated services code point (DSCP) or type of service (ToS), bandwidth enforcement using Resource Reservation Protocol (RSVP), and codec filtering.

CUBE functionality is implemented on devices using a special IOS feature set, which allows CUBE to route a call from one VoIP dial peer to another.

Protocol interworking is possible for the following combinations:

- H.323-to-SIP interworking
- H.323-to-H.323 interworking
- SIP-to-SIP interworking
The CUBE provides a network-to-network demarcation interface for signaling interworking, media interworking, address and port translations, billing, security, quality of service, call admission control, and bandwidth management.

The CUBE is used by enterprise and small and medium-sized organizations to interconnect SIP PSTN access with SIP and H.323 enterprise unified communications networks.

A CUBE interoperates with several different network elements including voice gateways, IP phones, and call-control servers in many different application environments, from advanced enterprise voice and/or video services with Cisco Unified Communications Manager or Cisco Unified Communications Manager Express, as well as simpler toll bypass and voice over IP (VoIP) transport applications. The CUBE provides organizations with all the border controller functions integrated into the network layer to interconnect unified communications voice and video enterprise-to-service-provider architectures.

Figure 2: Why Does an Enterprise Need the CUBE?

If an enterprise subscribes to VoIP services offered by an ITSP, connecting the enterprise CUCM through a CUBE provides network demarcation capabilities, such as security, topology hiding, transcoding, call admission control, protocol normalization and SIP registration, none of which is possible if CUCM connects directly to the ITSP. Another use case involves mergers or acquisitions in an enterprise and the need to integrate voice equipment, such as CUCMs, IP PBXs, VM servers, and so on. If the networks in the two organizations have overlapping IP addresses, CUBE can be used to connect the two distinct networks until the acquired organization can be migrated into the enterprise addressing plan.

SIP/H.323 Trunking

The Session Initiation Protocol (SIP) is a signaling communications protocol, widely used for controlling multimedia communication sessions such as voice and video calls over Internet Protocol (IP) networks. SIP (or H.323) trunking is the use of VoIP to facilitate the connection of a private branch exchange (PBX) to other VoIP endpoints across the Internet. To use SIP trunking, an enterprise must have a PBX (internal VoIP system) that connects to all internal end users, an Internet telephony service provider (ITSP) and a gateway that serves as the interface between the PBX and the ITSP. One of the most significant advantages of SIP and H.323...
trunking is the ability to combine data, voice, and video in a single line, eliminating the need for separate physical media for each mode.

**Figure 3: SIP/H.323 Trunking**

SIP trunking overcomes TDM barriers, in that it:
- Improves efficiency of interconnection between networks
- Simplifies PSTN interconnection with IP end-to-end
- Enables rich media services to employees, customers, and partners
- Carries converged voice, video, and data traffic

**Figure 4: SIP Trunking Overcomes TDM Barriers**
For Cisco IOS XE Gibraltar 16.11.1a and later releases, the SIP processes are initiated only when either of the following CLIs are configured on CUBE:

- **voice dialpeer** with session protocol as SIP.
- **voice register global**
- **sip-ua**

In the releases before Cisco IOS XE Gibraltar 16.11.1a, the following commands that are configured on CUBE initiated the SIP processes:

- **dial-peer voice (any)**
- **ephone-dn**
- **max-dn under call-manager-fallback**
- **ds0-group 0 timeslots 1 type e&m-wink-start**

**Typical Deployment Scenarios for CUBE**

CUBE in an enterprise environment serves two main purposes:

- **External Connections**—CUBE is the demarcation point within a unified communications network and provides interconnectivity with external networks. This includes H.323 and SIP voice and video connections.
- **Internal Connections**—When used within a VoIP network, CUBE increases flexibility and interoperability between devices.
Consider a scenario where XYZ corporation uses a VoIP network to provide phone services and uses a PRI connection for telecommunications services, and the PRI trunk is controlled by MGCP. Migration from MGCP PRI to SIP trunk is provided by ITSP telecommunications. CUCM sends the telephone number, as 10 digits, to CUBE. CUCM may send only the extension (4 digits) to the CUBE. When the call is diverted (using call-forward), the requirement of the ITSP is that they need the full 10-digit number in the SIP Diversion field.
The following sections describe the basic setup of CUBE through the steps involved in migrating the XYZ corporation to CUBE using a SIP trunk.

**Enabling the CUBE Application on a Device**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. mode border-element license capacity sessions
5. allow-connections from-type to to-type
6. end
### Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: <code>enable</code></td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td>Step 2: <code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Step 3: <code>voice service voip</code></td>
<td>Enters global VoIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# voice service voip</code></td>
<td></td>
</tr>
<tr>
<td>Step 4: <code>mode border-element license capacity sessions</code></td>
<td>Enables the set of commands used in the CUBE.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(conf-voi-serv)# mode border-element license capacity 200</code></td>
<td></td>
</tr>
<tr>
<td>Step 5: <code>allow-connections from-type to to-type</code></td>
<td>Allows connections between specific types of endpoints in a VoIP network.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(conf-voi-serv)# allow-connections sip to sip</code></td>
<td></td>
</tr>
<tr>
<td>Step 6: <code>end</code></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(conf-voi-serv)# end</code></td>
<td></td>
</tr>
</tbody>
</table>

### Verifying the CUBE Application on the Device

### Summary Steps

1. `enable`
2. `show cube status`

### Detailed Steps

**Step 1**  
`enable`  
Enables privileged EXEC mode.  
**Example:**
Device> enable

Step 2  show cube status
Displays the CUBE status, the software version, the license capacity, the image version, and the platform name of the device. The CUBE status display is enabled only if the mode border-element command is configured with call license capacity.

Example:
Device# show cube status
CUBE-Version : 10.0.1
SW-Version : 15.4.2.T, Platform 3845
HA-Type : none
Licensed-Capacity : 200

Configuring a Trusted IP Address List for Toll-Fraud Prevention

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. ip address trusted list
5. ipv4 ipv4-address [network-mask]
6. ipv6 ipv6-address
7. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters global VoIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ip address trusted list</td>
<td>Enters IP address trusted list mode and enables the addition of valid IP addresses.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# ip address trusted list</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring a Trusted IP Address List for Toll-Fraud Prevention

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 5**
- ipv4 ipv4-address [network-mask]  
  **Example:**  
  Device(cfg-iptrust-list)# ipv4 192.0.2.1 255.255.255.0 | Allows you to add up to 100 IPv4 addresses in the IP address trusted list. Duplicate IP addresses are not allowed.  
  - The *network-mask* argument allows you to define a subnet IP address. |
| **Step 6**
- ipv6 ipv6-address  
  **Example:**  
  Device(cfg-iptrust-list)# ipv6 2001:DB8:0:ABCD::1/48 | Allows you to add IPv6 addresses to the trusted IP address list. |
| **Step 7**
- end  
  **Example:**  
  Device(cfg-iptrust-list)# end | Returns to privileged EXEC mode. |
CHAPTER 5

Virtual CUBE

Cisco Unified Border Element (CUBE) has been traditionally supported as an installation on physical routers such as Cisco Aggregation Services Router Series (ASR) and Cisco Integration Services Router Series (ISR). From Cisco IOS XE 3.15S release onwards, the CUBE feature set is supported on the Cisco CSR 1000V Series Cloud Services Routers (Cisco CSR 1000V) in a virtualized form factor. Virtual CUBE (vCUBE) enables the traditional CUBE feature set to be deployed in cloud-based data center deployments.

- Feature Information for Virtual CUBE, on page 23
- Prerequisites for Virtual CUBE, on page 24
- Features Supported with Virtual CUBE, on page 24
- Restrictions, on page 25
- Information about Virtual CUBE Support on Cisco CSR 1000V Series Routers, on page 25
- Installation, on page 28
- How to Enable Virtual CUBE on Cisco CSR 1000V Series Router, on page 29
- Troubleshooting Virtual CUBE Support, on page 29

Feature Information for Virtual CUBE

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

**Table 1: Feature Information for Virtual CUBE Support**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual CUBE Feature Support</td>
<td>Cisco IOS XE Release 3.15S</td>
<td>Provides support for virtual CUBE on Cisco Cloud Services Router, Cisco CSR 1000V Series.</td>
</tr>
</tbody>
</table>
**Prerequisites for Virtual CUBE**

**Hardware**

- The Virtual CUBE (vCUBE) feature set is bundled as part of the Cisco CSR 1000V Series and is supported on VMware's ESXi hypervisor and Cisco UCS servers (UCS-C and UCS-E series). For more information on the Cisco CSR1000V series and their supported ESXi versions, see Installing the Cisco CSR 1000v in VMware ESXi Environments.
  
  For Cisco IOS XE Gibraltar 16.10 and later releases, VMware ESXi 6.5 update 1 is the recommended VMware version. For more information on supported hypervisors, see Product Specifications.

- We recommend that you disable hyper threading while configuring Virtual CUBE on Cisco UCS. For more information on the best practices for setting BIOS parameters for performance, see BIOS Settings.

- A minimum of two network interfaces are required for configuring Virtual CUBE.

**Software**

- Obtain the relevant license for the Cisco CSR 1000V router. For details on the licensing package support, see Licensing Package Support, on page 28.

- Install the appropriate Cisco IOS image on the Cisco CSR 1000V router and configure a working VoIP network. For details on installation, see Installation, on page 28.

  For details on the ESXi hypervisor support, see the section on Hardware, on page 24.

**Features Supported with Virtual CUBE**

Virtual CUBE supports most of the features available in CUBE. However, any feature that manages the media plane is not expected to work in the Cisco CSR 1000V router. The following features are not supported in virtual CUBE:

- All DSP based features
  - Codec Transcoding, Transrating
  - DTMF interworking
  - Call Progress Analysis (CPA)
  - Noise Reduction (NR), Acoustic Shock Protection (ASP), and Audio Gain

- Limited Voice Class Codec (VCC) support
  - Codec supported on peer leg will be included in offer. Other codecs will be filtered out.

- IOS Gatekeeper

- H.323 Interworking
• IOS based Hardware MTP

Restrictions

• Software MTP is not supported on Virtual CUBE.

Note

All caveats, restrictions, and limitations of Cisco ASR IOS-XE 3.15 and later releases are applicable to Virtual CUBE.

Information about Virtual CUBE Support on Cisco CSR 1000V Series Routers

High Availability

Virtual CUBE uses Redundancy Group infrastructure for High Availability (HA). For High Availability, the two virtual CUBE CSR instances must be running on the same host or across different hosts that are connected through a switch. Geographic stateful switchover is not supported.

*Figure 7: Virtual CUBE High Availability*

On the Cisco CSR 1000V, the box-to-box redundancy option uses the Redundancy Group (RG) Infrastructure to form an Active and Standby pair of routers. The Active and Standby pair share the same virtual IP address (VIP) and continually exchange status messages. Virtual CUBE session information is check-pointed across the Active and Standby pair of routers enabling the Standby router to take over all Virtual CUBE call processing responsibilities if the Active router should go out of service. The Standby router sends and receives all...
redundancy-related traffic (protocol packets, configuration data, keepalives, and peer status). Calls in the transient state at the time of failover are disconnected.

**Introduction of CSR 1000V Network Interfaces for HA Interaction**

Cisco CSR 1000V can run on the VMware ESXi hypervisor. Install the Cisco CSR 1000V .iso file on your host and manually create the virtual machine (VM) using your hypervisor software. VMware ESXi runs on servers with x86-based CPUs. You can use the same hypervisor to run several VMs. The ESXi host is connected to a port on the physical switch. In this environment, VMware ESXi provides the virtual switch (vSwitch) functionality that routes traffic internally between virtual machines and links to external networks.

*Figure 8: vNICs Mapped to Cisco CSR 1000V Router Interfaces*

**Stateful Switchover for Virtual CUBE**

Stateful switchover (SSO) occurs when the Cisco CSR 1000V Active router crashes and reload happens. A switchover from the active to the standby router occurs when the active Route Processor fails, or when the keepalive messages between the Active and Standby router is lost. In a usual scenario, the Standby router sends RG Infra keepalive message to the Active router and expects an acknowledgment within 100 msec as per the default timer. If there is no acknowledgment of keepalive within 100 msec, the Standby router immediately sends a message to the Active router to check the HA status. If there is no keepalive message
response, the Standby router declares the Active router (it goes into reload state) is dead and assumes the role of the Active router.

**Key Considerations for Deploying HA on Virtual CUBE**

The following are some of the key points to consider while deploying a Virtual CUBE instance in HA mode on the Cisco CSR 1000V router:

- **Virtual CUBE stateful switchover occurs only due to software failures**—The Active router reload when the Cisco CSR 1000V crashes due to software faults. When a software fault causes the active router to go to the reload state, the Standby router assumes the role of the new Active router.

- **Virtual CUBE tracks only the next vSwitch interface**—Virtual CUBE HA supports tracking of only the interface link status. In the case of virtual CUBE, the interface is always connected to the vSwitch of the hypervisor. The interface link status of the Virtual CUBE shows that it is up even when the physical link of the server is down as the vSwitch link is still active. It may be possible to configure the hypervisor vSwitch to propagate the physical link status to the Virtual CUBE interface. Here, the interface tracking accurately reflects the status of the physical interface. Refer to the software documentation of the hypervisor you are using.

- **High Availability Connectivity using vSwitch Redundancy**—We recommend that you use vSwitch redundancy as suggested by the Cisco CSR 1000V reference documents. See [Cisco CSR 1000V Series Cloud Services Router Software Configuration Guide](#).

  In a scenario where the physical switch is down, both the Active and Standby routers become the Active router. Here, the Active router is still up and running whereas the Standby router assumes that the Active router has gone down as it does not receive any keepalive messages or peer information from the Active router. Hence, the Standby Router also turns to the Active router. Though Virtual CUBE continues to maintain existing calls and also receive new calls while both routers are active, HA checkpointing fails.

- **Virtual CUBE does not track uplink failures**—You must not use switches that are used to connect non-networking end devices or must not track LAN for determining uplink failures. Basically, switches used in the aggregation layers are not tracked.

Implementation of High Availability is same for the physical CUBE and Virtual CUBE on IOS XE platforms. For more information on High Availability, see [High Availability Overview](#), on page 663.

**Configuration Example for High Availability on Virtual CUBE**

Configuration on both the Virtual CUBEs must have identical physical configuration and must be running on the same type of platform and the IOS XE version. Anytime a platform is reloaded in a Virtual CUBE high availability implementation, it always boots up as the standby router.

```
Router> enable
Router# configure terminal
redundancy
    mode none
    application redundancy
        group 1
            name voice-b2bha
            priority 100 failover threshold 75
            control GigabitEthernet 0/0/2 protocol 1
            data GigabitEthernet 0/0/2
            timers delay 30 reload 60
voice service voip
    mode border-element
    allow-connections sip to sip
redundancy-group 1
```
Media

Virtual CUBE media performance depends on the underlying VM platform consistently providing packet switching latency of less than 5 ms. Latency and jitter values observed on a virtual CUBE are same as the values obtained on CUBE running on a hardware platform with recommended hardware configuration and identical software configuration, under the same network conditions.

For more information on how to calculate the performance requirements, see Voice Quality Monitoring, on page 857.

Licensing Package Support

Virtual CUBE is enabled with the APPX and AX license packages. The AX license package provides access to all features supported in virtual CUBE. When the license is installed, the virtual CUBE related CLI commands such as voice and dial-peer configurations are visible. Also, relevant CUBE processes are instantiated.

The following table details the license package support for a virtual CUBE.

<table>
<thead>
<tr>
<th>Virtual CUBE Session License</th>
<th>CSR Package</th>
<th>Features</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same CUBE Licensing SKUs as Cisco ASR 1000 Series</td>
<td>APPX</td>
<td>No TLS / SRTP support</td>
<td>session count * (signaling + media bandwidth)</td>
</tr>
<tr>
<td></td>
<td>AX</td>
<td>All vCUBE features</td>
<td></td>
</tr>
</tbody>
</table>

For detailed information on licensing, see Cisco CSR 1000V Series Cloud Services Router Configuration Guide.

Installation

You can install Virtual CUBE in two ways:

- Install using an OVA file
- Install using an ISO image

Install Virtual CUBE Using an OVA File

You can use the OVA file that is included in the Cisco CSR 1000V router software image package to install virtual CUBE. The file is used for deploying the OVA template on the VM (in TAR format).
Explicit subscription of CPUs and memory is required while deploying OVA provided by Cisco CSR 1000V.

For details on how to perform the deployment, see Cisco CSR 1000V Series Cloud Services Router Software Configuration Guide.

### Install Virtual CUBE Using an ISO Image

You can use the .iso file to install virtual CUBE. This file is used for installing the software image on the VM (requires manually creating the VM).

For details on how to perform the installation, see Cisco CSR 1000V Series Cloud Services Router Software Configuration Guide.

### How to Enable Virtual CUBE on Cisco CSR 1000V Series Router

For details on the steps to enable virtual CUBE on a CSR 1000V router, see Enabling the CUBE Application on a Device.

### Troubleshooting Virtual CUBE Support


To troubleshoot Virtual Machine (VM) issues, see Cisco CSR 1000V Series Cloud Services Router Software Configuration Guide.
Dial-Peer Matching

CUBE allows VoIP-to-VoIP connection by routing calls from one VoIP dial peer to another. As VoIP dial peers can be handled by either SIP or H.323, CUBE can be used to interconnect VoIP networks of different signaling protocols. VoIP interworking is achieved by connecting an inbound dial peer with an outbound dial peer.

All CUBE Enterprise deployments must have signaling and media bind statements specified at the dial-peer or voice class tenant level. For voice call tenants, you must apply tenants to dial-peers used for CUBE call flows if these dial-peers do not have bind statements specified.

- Dial Peers in CUBE, on page 31
- Configuring Inbound and Outbound Dial-Peer Matching for CUBE, on page 33
- Preference for Dial-Peer Matching, on page 34

Dial Peers in CUBE

A dial peer is a static routing table, mapping phone numbers to interfaces or IP addresses.

A call leg is a logical connection between two routers or between a router and a VoIP endpoint. A dial peer is associated or matched to each call leg according to attributes that define a packet-switched network, such as the destination address.

Voice-network dial peers are matched to call legs based on configured parameters, after which an outbound dial peer is provisioned to an external component using the component's IP address. For more information, refer to the Dial Peer Configuration Guide.

Dial-peer matching can also be done based on the VRF ID associated with a particular interface. For more information, see Inbound Dial-Peer Matching Based on Multi-VRF, on page 352.

In CUBE, dial peers can also be classified as LAN dial peers and WAN dial peers based on the connecting entity from which CUBE sends or receives calls.
A LAN dial peer is used to send or receive calls between CUBE and the Private Branch Exchange (PBX)—a system of telephone extensions within an enterprise. Given below are examples of inbound and outbound LAN dial peers.

Inbound Dial-Peer for calls from CUCM to CUBE

dial-peer voice 100 voip
description *** Inbound LAN side dial-peer ***
incoming called-number 9T
session protocol sipv2
codec g711ulaw
dtmf-relay rtp-nte

A WAN dial peer is used to send or receive calls between CUBE and the SIP trunk provider. Given below are examples of inbound and outbound WAN dial peers.

Outbound Dial-Peer for calls from CUBE to CUCM

dial-peer voice 200 voip
description *** Outbound LAN side dial-peer ***
destination-pattern [2-9]........
session protocol sipv2
session target ipv4:<CUCM_Address>
codec g711ulaw
dtmf-relay rtp-nte

CUCM sending 9 + All digits dialed (Outgoing calls)
Incoming call number used to match the inbound LAN dial peer

SP will be sending 10 digits inbound (Incoming Calls)
Destination pattern used to match the outbound LAN dial peer
## Configuring Inbound and Outbound Dial-Peer Matching for CUBE

The following commands can be used for inbound and outbound dial peer matching in the CUBE:

### Table 2: Incoming Dial-Peer Matching

<table>
<thead>
<tr>
<th>Command in Dial-Peer Configuration</th>
<th>Description</th>
<th>Call Setup Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>incoming called-number DNIS-string</td>
<td>This command uses the destination number that was called to match the incoming call leg to an inbound dial peer. This number is called the dialed number identification service (DNIS) number.</td>
<td>DNIS number</td>
</tr>
<tr>
<td>answer-address ANI-string</td>
<td>This command uses the calling number to match the incoming call leg to an inbound dial peer. This number is called the originating calling number or automatic number identification (ANI) string.</td>
<td>ANI string</td>
</tr>
<tr>
<td>destination-pattern ANI-string</td>
<td>This command uses the inbound call leg to the inbound dial peer.</td>
<td>ANI string for inbound</td>
</tr>
</tbody>
</table>
### Call Setup Element

<table>
<thead>
<tr>
<th>Command in Dial-Peer Configuration</th>
<th>Description</th>
<th>Call Setup Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{incoming called</td>
<td>incoming calling} e164-pattern-map pattern-map-group-id`</td>
<td>This command uses a group of incoming called (DNIS) or incoming calling (ANI) number patterns to match the inbound call leg to an inbound dial peer. The command calls a globally defined voice class identifier where the E.164 pattern groups are configured.</td>
</tr>
<tr>
<td><code>voice class uri</code> `URI-class-identifier with incoming uri {from</td>
<td>request</td>
<td>to</td>
</tr>
<tr>
<td>`incoming uri {called</td>
<td>calling} URI-class-identifier`</td>
<td>This command uses the directory URI (Uniform Resource Identifier) number to match the outgoing H.323 call leg to an outgoing dial peer. The command calls a globally defined voice class identifier where the directory URI is configured.</td>
</tr>
</tbody>
</table>

### Table 3: Outgoing Dial-Peer Matching

<table>
<thead>
<tr>
<th>Dial-Peer Command</th>
<th>Description</th>
<th>Call Setup Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>destination-pattern DNIS-string</code></td>
<td>This command uses DNIS string to match the outbound call leg to the outbound dial peer.</td>
<td>DNIS string for outbound ANI string for inbound</td>
</tr>
<tr>
<td><code>destination URI-class-identifier</code></td>
<td>This command uses the directory URI (Uniform Resource Identifier) number to match the outgoing call leg to an outgoing dial peer. This directory URI is part of the SIP address of a device. The command actually refers to a globally defined voice class identifier where the directory URI is configured.</td>
<td>Directory URI</td>
</tr>
<tr>
<td><code>destination e164-pattern-map pattern-map-group-id</code></td>
<td>This command uses a group of destination number patterns to match the outbound call leg to an outbound dial peer. The command calls a globally defined voice class identifier where the E.164 pattern groups are configured.</td>
<td>E.164 patterns</td>
</tr>
</tbody>
</table>

### Preference for Dial-Peer Matching

The following is the order in which inbound dial-peer is matched for SIP call-legs:
• voice class uri \textit{URI-class-identifier} with \textit{incoming uri} \{\textit{via}\} \textit{URI-class-identifier}
• voice class uri \textit{URI-class-identifier} with \textit{incoming uri} \{\textit{request}\} \textit{URI-class-identifier}
• voice class uri \textit{URI-class-identifier} with \textit{incoming uri} \{\textit{to}\} \textit{URI-class-identifier}
• voice class uri \textit{URI-class-identifier} with \textit{incoming uri} \{\textit{from}\} \textit{URI-class-identifier}
• \textit{incoming called-number} \textit{DNIS-string}
• \textit{answer-address} \textit{ANI-string}

The following is the order in which inbound dial-peer is matched for H.323 call-legs:
• \textit{incoming uri} \{\textit{called}\} \textit{URI-class-identifier}
• \textit{incoming uri} \{\textit{callling}\} \textit{URI-class-identifier}
• \textit{incoming called-number} \textit{DNIS-string}
• \textit{answer-address} \textit{ANI-string}

The following is the order in which outbound dial-peer is matched for SIP call-legs:
• \textit{destination route-string}
• \textit{destination} \textit{URI-class-identifier} with \textit{target carrier-id} \textit{string}
• \textit{destination-pattern} with \textit{target carrier-id} \textit{string}
• \textit{destination} \textit{URI-class-identifier}
• \textit{destination-pattern}
• \textit{target carrier-id} \textit{string}

\begin{quote}
\textbf{Note} \\
If CUBE with Cisco Unified Communications Manager Express (CUCME) is configured with the same DNs, then the ANI is given the preference. The system dial-peer for the DN is selected over the other dial-peers created.
\end{quote}
Preference for Dial-Peer Matching
CHAPTER 7

DTMF Relay

The DTMF Relay feature allows CUBE to send dual-tone multi-frequency (DTMF) digits over IP. This chapter talks about DTMF tones, DTMF relay mechanisms, how to configure DTMF relays, and interoperability and priority with multiple relay methods.

- Feature Information for DTMF Relay, on page 37
- Information About DTMF Relay, on page 38
- Verifying DTMF Relay, on page 46

Feature Information for DTMF Relay

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 4: Feature Information for DTMF Relay

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTMF Relay</td>
<td>Cisco IOS 12.1(2)T</td>
<td>The DTMF Relay feature allows CUBE to send DTMF digits over IP. The <code>dtmf-relay</code> command was added.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 2.1</td>
<td></td>
</tr>
<tr>
<td>Support for <code>sip-info</code> to <code>rtp-nte</code></td>
<td>Cisco IOS XE Everest 16.6.1</td>
<td>This feature adds support for <code>sip-info</code> to <code>rtp-nte</code> DTMF relay mechanism for SIP-SIP calls.</td>
</tr>
</tbody>
</table>
Information About DTMF Relay

DTMF Tones

DTMF tones are used during a call to signal to a far-end device; these signals may be for navigating a menu system, entering data, or for other types of manipulation. They are processed differently from the DTMF tones that are sent during the call setup as part of the call control. TDM interfaces on Cisco devices support DTMF by default. Cisco VoIP dial-peers do not support DTMF relay by default and require DTMF relay capabilities to be enabled.

Note

DTMF tones sent by phones do not traverse the CUBE.

DTMF Relay

Dual-tone multi-frequency (DTMF) relay is the mechanism for sending DTMF digits over IP. The VoIP dial peer can pass the DTMF digits either in band or out of band.

In-band DTMF-Relay passes the DTMF digits using the RTP media stream and uses a special payload type identifier in the RTP header to distinguish DTMF digits from actual voice communication. This method is more likely to work on lossless codecs, such as G.711.

Note

The main advantage of DTMF relay is that low bandwidth codecs like G.729 and G.723 is sent with greater fidelity when sent using in-band DTMF relay. Without the use of DTMF relay, calls established with low-bandwidth codecs may have trouble accessing automated DTMF-based systems, such as voice mail, menu-based Automatic Call Distributor (ACD) systems, and automated banking systems.

Out-of-band DTMF-Relay passes DTMF digits using a signaling protocol (SIP or H.323) instead of using the RTP media stream.

DTMF relay prevents loss of integrity of DTMF digits caused by VoIP compressed codecs. The relayed DTMF is then regenerated transparently on the peer side.

Figure 12: DTMF Relay Mechanism
DTMF relay mechanisms supported on VoIP dial-peers are listed below based on the keywords used to configure them. The DTMF relay mechanism can be either out-of-band (H.323 or SIP) or inband (RTP).

- **h245-alphanumeric and h245-signal**—These two methods are available only on H.323 dial peers. This is an out-of-band DTMF relay mechanism that transports the DTMF signals using H.245, which is the media control protocol of the H.323 protocol suite.

  The H245-signal method carries more information about the DTMF event (such as its actual duration) than the H245-Alphanumeric method. This addresses a potential problem with the alphanumeric method when interworking with other vendors’ systems.

- **sip-notify**—This method is available on SIP dial peers only. This is a Cisco proprietary out-of-band DTMF relay mechanism that transports DTMF signals using SIP-Notify message. The SIP Call-Info header is used to indicate the use of the SIP-Notify DTMF relay mechanism. The message is acknowledged with a 180 or 200 response message containing a similar SIP Call-Info header.

  The Call-Info header for NOTIFY-based out-of-band relay is as follows:

  ```
  Call-Info: <sip: address>; method="NOTIFY;Event=telephone-event;Duration=msec"
  ```

  DTMF relay digits are sent as 4 bytes in a binary encoded format.

  This mechanism is useful for communicating with SCCP IP phones that do not support inband DTMF digits and analog phones attached to analog voice ports (FXS) on the router.

  If multiple DTMF relay mechanisms are enabled on a SIP dial peer and are negotiated successfully, NOTIFY-based out-of-band DTMF relay takes precedence.

- **sip-kpml**—This method is available only on SIP dial peers. This is an out-of-band DTMF relay mechanism defined by RFC 4730 that registers the DTMF signals using SIP-Subscribe messages and transports the DTMF signals using SIP-Notify messages containing an XML-encoded body. This method is also known as Key Press Markup Language.

  If you configure KPML on the dial peer, the gateway sends INVITE messages with kpml in the Allow-Events header.

  This method is mostly used for SIP endpoints registered to CUCM or CME. This method is useful for non-conferencing calls and for interoperability between SIP products and SIP phones.

  If you configure rtp-nte, sip-notify, and sip-kpml, the outgoing INVITE contains an SDP with rtp-nte payload, a SIP Call-Info header, and an Allow-Events header with KPML.

  The following SIP-Notify message is a sample taken after the subscription has taken place. The endpoints transmit digits using SIP-Notify messages with KPML events through XML. In the following example, the digit “1” is being transmitted:

  ```
  NOTIFY sip:192.168.105.25:5060 SIP/2.0
  Event: kpml
  <?xml version="1.0" encoding="UTF-8"?>
  <kpml-response version="1.0" code="200" text="OK" digits="1" tag="dtmf"/>
  ```

- **sip-info**—The sip-info method is available only on SIP dial peers. This is an out-of-band DTMF relay mechanism that registers the DTMF signals using SIP-Info messages. The body of the SIP message consists of signaling information and uses the content-type application/dtmf-relay.

  The method is always enabled for SIP dial peers, and is invoked when a SIP INFO message is received with DTMF relay content.

  This following sample message shows a SIP INFO message received by the gateway with specifics about the DTMF tone to be generated. The combination of the From, To, and Call-ID headers identifies the
call leg. The signal and duration headers specify the digit, in this case 1, and duration, 160 milliseconds in the example, for DTMF tone play.

```
INFO sip:2143302100@172.17.2.33 SIP/2.0
Via: SIP/2.0/UDP 172.80.2.100:5060
From: <sip:9724401003@172.80.2.100>;tag=43
To: <sip:2143302100@172.17.2.33>;tag=9753.0207
Call-ID: 984072_15401962@172.80.2.100
CSeq: 25634 INFO
Supported: 100rel
Supported: timer
Content-Length: 26
Content-Type: application/dtmf-relay
Signal= 1
Duration= 160
```

- **rtp-n-te**—Real-Time Transport Protocol (RTP) Named Telephone Events (NTE). This is an in-band DTMF relay mechanism defined by RFC2833. RFC2833 defines formats of NTE-RTP packets used to transport DTMF digits, hook flash, and other telephony events between two peer endpoints. DTMF tones are sent as packet data after call media has been established using the RTP stream and are distinguished from the audio by the RTP payload type field, preventing compression of DTMF-based RTP packets. For example, the audio of a call can be sent on a session with an RTP payload type that identifies it as G.711 data, and the DTMF packets are sent with an RTP payload type that identifies them as NTEs. The consumer of the stream utilizes the G.711 packets and the NTE packets separately.

The SIP NTE DTMF relay feature provides reliable digit relay between Cisco VoIP gateways when a low-bandwidth codec is used.

**Note**

Payload types 96 and 97 are used for fax by default in Cisco devices. A third party device may use payload type 96 and 97 for dtmf. In such scenarios, we recommend you to perform one of the following:

- Change the payload type for fax in both incoming and outgoing dial-peers using `rtp payload-type` command
- Use `assymmetric payload dtmf` command

For more information on configuring `rtp payload-type` and `assymmetric payload dtmf`, see Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls.

Payload types and attributes of this method are negotiated between the two ends at call setup using the Session Description Protocol (SDP) within the body section of the SIP message.

**Note**

This method should not be confused with the “Voice in-band audio/G711” transport because the latter is just the audible tones being passed as normal audio without any relay signaling method being “aware” or involved in the process. This is just plain audio passing through end-to-end using the G711Ulaw/Alaw codec.

- **cisco-rtp**—This is an in-band DTMF relay mechanism that is Cisco proprietary, where the DTMF digits are encoded differently from the audio and are identified as payload type 121. The DTMF digits are part
of the RTP data stream and distinguished from the audio by the RTP payload type field. This method is not supported by CUCM and its use has been discontinued.

Note: The `cisco-rtp` operates only between two Cisco 2600 series or Cisco 3600 series devices. Otherwise, the DTMF relay feature does not function, and the gateway sends DTMF tones in-band.

- **G711 audio**—This is an in-band DTMF relay mechanism that is enabled by default and requires no configuration. Digits are transmitted within the audio of the phone conversation, that is, it is audible to the conversation partners; therefore, only uncompressed codecs like g711 alaw or ulaw can carry inband DTMF reliably. Female voices are known to, sometimes, trigger the recognition of a DTMF tone.

  Digits are passed along just like the rest of your voice as normal audio tones with no special coding or markers using the same codec as your voice does and are generated by your phone.

### Configuring DTMF Relays

You can configure DTMF relay using the `dtmf-relay method1 `[...`method6]` command in the VoIP dial peer.

DTMF negotiation is performed based on the matching inbound dial-peer configuration.

The `method` variable used here can be any of the following:

- `h245-alphanumeric`
- `h245-signal`
- `sip-notify`
- `sip-kpml`
- `sip-info`
- `rtp-nte [digit-drop]`
- `cisco-rtp`

Multiple DTMF methods may be configured on CUBE simultaneously in order to minimize MTP requirements. If you configure more than one out-of-band DTMF method, preference goes from highest to lowest in the order of configuration. If an endpoint does not support any of the DTMF relay mechanism configured on CUBE, an MTP or transcoder is required.

The following table lists the DTMF relay types supported on a SIP and H.322 gateway.

<table>
<thead>
<tr>
<th></th>
<th>H.323 Gateway</th>
<th>SIP Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-band</td>
<td><code>cisco-rtp, rtp-nte</code></td>
<td><code>rtp-nte</code></td>
</tr>
<tr>
<td>Out-of-band</td>
<td><code>h245-alphanumeric, h245-signal</code></td>
<td><code>sip-notify, sip-kpml, sip-info</code></td>
</tr>
</tbody>
</table>

### Interoperability and Priority with Multiple DTMF Relay Methods

- CUBE negotiates both `rtp-nte` and `sip-kpml` if both are supported and advertised in the incoming INVITE. However, CUBE relies on the `rtp-nte` DTMF method to receive digits and a SUBSCRIBE if `sip-kpml`
is not initiated. CUBE still accepts SUBSCRIBEs for KPML. This prevents double-digit reporting problems at CUBE.

• CUBE negotiates to one of the following:
  • cisco-rtp
  • rtp-nte
  • rtp-nte and kpml
  • kpml
  • sip-notify

• If you configure rtp-nte, sip-notify, and sip-kpml, the outgoing INVITE contains a SIP Call-Info header, an Allow-Events header with KPML, and an sdpl with rtp-nte payload.

• If you configure more than one out-of-band DTMF method, preference goes from highest to lowest in the order of configuration.

• CUBE selects DTMF relay mechanisms using the following priority:
  • sip-notify or sip-kpml (highest priority)
  • rtp-nte
  • None—Send DTMF in-band

H.323 gateways select DTMF relay mechanisms using the following priority:

• cisco-rtp
• h245-signal
• h245-alphanumeric
• rtp-nte
• None—Send DTMF in-band

### DTMF Interoperability Table

This table provides the DTMF interoperability information between various DTMF relay types in different call flow scenarios. For instance, if you need to configure sip-kpml on an inbound dial peer and h245-signaling on an outbound dial peer in an RTP-RTP Flow through configuration, refer table 1 to see that the combination is supported (as image information is present) and the required image is IOS 12.4(15)T or IOS XE Supported or above. The call scenarios provided are as follows:

• RTP-RTP Flow-Through
• RTP-RTP with transcoder Flow-Through
• RTP-RTP Flow Around
• RTP-RTP with high-density transcoder Flow Through
• SRTP-RTP Flow Through
### Table 6: RTP-RTP Flow-Through

<table>
<thead>
<tr>
<th>Relay Type</th>
<th>H.323</th>
<th>SIP</th>
<th>Inband</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.323 h245-alpha numeric</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>h245-signal</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>rtp-n-te</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-inf</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-kpml</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-notify</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-info</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

### Table 7: RTP-RTP with DSP involved Flow-Through Calls

<table>
<thead>
<tr>
<th>Relay Type</th>
<th>H.323</th>
<th>SIP</th>
<th>Inband</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.323 h245-alpha numeric</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>h245-signal</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>rtp-n-te</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-inf</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-kpml</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-notify</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>sip-info</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

---

3 Supported from Cisco IOS XE Everest 16.6.1 onwards for calls that do not involve DSP resources.

* media resource is required (Transcoder) for IOS versions
### DTMF Interoperability Table

<table>
<thead>
<tr>
<th>dial-peer protocol</th>
<th>H.323</th>
<th>SIP</th>
<th>Inband</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inband</td>
<td>Voice Inband (G.711)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 8: RTP-RTP Flow Around**

<table>
<thead>
<tr>
<th>dial-peer protocol</th>
<th>H.323</th>
<th>SIP</th>
<th>Inband</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.323</td>
<td>h245-alpha numeric</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>h245-signal</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rtp-nte</td>
<td>Supported</td>
<td>Supported*</td>
</tr>
<tr>
<td>SIP</td>
<td>rtp-nte</td>
<td>Supported</td>
<td>Supported*</td>
</tr>
<tr>
<td></td>
<td>sip-kpml</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-notify</td>
<td></td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-info</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inband</td>
<td>Voice Inband (G.711)</td>
<td>Supported*</td>
<td>Supported</td>
</tr>
</tbody>
</table>

* media resource is required (Transcoder) for IOS versions. CUBE falls back to flow-through mode if media resource is unavailable.
Table 9: RTP-RTP with high-density transcoder Flow Through

<table>
<thead>
<tr>
<th>dial-peer protocol</th>
<th>DTMF Relay Type</th>
<th>H.323 protocol</th>
<th>SIP protocol</th>
<th>Inband</th>
</tr>
</thead>
<tbody>
<tr>
<td>dial-peer protocol</td>
<td>DTMF Relay Type</td>
<td>h245-alpha numeric</td>
<td>h245-signal</td>
<td>rtp-nte</td>
</tr>
<tr>
<td>H.323</td>
<td>H.323</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h245-alpha numeric</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h245-signal</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rtp-nte</td>
<td>Supported</td>
<td></td>
<td>Supported</td>
</tr>
<tr>
<td>SIP</td>
<td>rtp-nte</td>
<td>Supported</td>
<td></td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-kpml</td>
<td>Supported</td>
<td></td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-notify</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-info</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inband</td>
<td>Voice Inband (G.711)</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10: SRTP-RTP Flow Through

<table>
<thead>
<tr>
<th>dial-peer protocol</th>
<th>DTMF Relay Type</th>
<th>H.323 protocol</th>
<th>SIP protocol</th>
<th>Inband</th>
</tr>
</thead>
<tbody>
<tr>
<td>dial-peer protocol</td>
<td>DTMF Relay Type</td>
<td>h245-alpha numeric</td>
<td>h245-signal</td>
<td>rtp-nte</td>
</tr>
<tr>
<td>H.323</td>
<td>H.323</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h245-alpha numeric</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h245-signal</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rtp-nte</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIP</td>
<td>rtp-nte</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-kpml</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-notify</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-info</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inband</td>
<td>Voice Inband (G.711)</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For calls sent from an in-band (RTP-NTE) to an out-of-band method, configure the `dtmf-relay rtp-nate digit-drop` command on the inbound dial-peer and the desired out-of-band method on the outgoing dial-peer. Otherwise, same digit is sent in OOB as well as in-band, and gets interpreted as duplicate digits by the receiving end. When the digit-drop option is configured on the inbound leg, CUBE suppresses NTE packets and only relay digits using the OOB method configured on the outbound leg.

### Verifying DTMF Relay

**SUMMARY STEPS**

1. `show sip-ua calls`
2. `show sip-ua calls dtmf-relay sip-info`
3. `show sip-ua history dtmf-relay kpml`
4. `show sip-ua history dtmf-relay sip-notify`

**DETAILED STEPS**

**Step 1**

**show sip-ua calls**

The following sample output shows that the DTMF method is SIP-KPML.

**Example:**

```
Device# show sip-ua calls

SIP UAC CALL INFO
Call 1
SIP Call ID : 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 
Called Number : 8888
Bit Flags : 0xD44018 0x100 0x0
CC Call ID : 6
Source IP Address (Sig) : 192.0.2.1
Destn SIP Req Addr:Port : 192.0.2.2:5060
Destn SIP Resp Addr:Port : 192.0.2.3:5060
Destination Name : 192.0.2.4.250
Number of Media Streams : 1
Number of Active Streams: 1
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 6
Stream Type : voice-only (0)
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : sip-kpml
Dtmf-relay Payload Type : 0
Media Source IP Addr:Port : 192.0.2.5:17576
Media Dest IP Addr:Port : 192.0.2.6:17468
```
Orig Media Dest IP Addr:Port : 0.0.0.0:0
Number of SIP User Agent Client(UAC) calls: 1
SIP UAS CALL INFO
Number of SIP User Agent Server(UAS) calls: 0

Step 2  show sip-uacalls dtmf-relay sip-info

The following sample output displays active SIP calls with INFO DTMF Relay mode.

Example:

Device# show sip-uacalls dtmf-relay sip-info

Total SIP call legs:2, User Agent Client:1, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID : 9598A547-5C1311E2-8008F709-2470C996@172.27.161.122
   State of the call : STATE_ACTIVE (?)
   Calling Number : sipp
   Called Number : 3269011111
   CC Call ID : 2
   No. Timestamp Digit Duration
   ---------------------------------------------------------------
   0 01/12/2013 17:23:25.615 2 250
   1 01/12/2013 17:23:25.967 5 300
   2 01/12/2013 17:23:26.367 6 300

Call 2
SIP Call ID : 1-29452@172.25.208.177
   State of the call : STATE_ACTIVE (?)
   Calling Number : sipp
   Called Number : 3269011111
   CC Call ID : 1
   No. Timestamp Digit Duration
   ---------------------------------------------------------------
   0 01/12/2013 17:23:25.615 2 250
   1 01/12/2013 17:23:25.967 5 300
   2 01/12/2013 17:23:26.367 6 300

Number of SIP User Agent Client(UAC) calls: 2

SIP UAS CALL INFO
Call 1
SIP Call ID : 1-29452@172.25.208.177
   State of the call : STATE_ACTIVE (?)
   Calling Number : sipp
   Called Number : 3269011111
   CC Call ID : 1
   No. Timestamp Digit Duration
   ---------------------------------------------------------------
   0 01/12/2013 17:23:25.615 2 250
   1 01/12/2013 17:23:25.967 5 300
   2 01/12/2013 17:23:26.367 6 300

Call 2
SIP Call ID : 9598A547-5C1311E2-8008F709-2470C996@172.27.161.122
   State of the call : STATE_ACTIVE (?)
   Calling Number : sipp
   Called Number : 3269011111
   CC Call ID : 2
   No. Timestamp Digit Duration
   ---------------------------------------------------------------
   0 01/12/2013 17:23:25.615 2 250
   1 01/12/2013 17:23:25.967 5 300

Cisco Unified Border Element Configuration Guide
Step 3 show sip-ua history dtmf-relay kpml

The following sample output displays SIP call history with KMPL DTMF Relay mode.

Example:

Device# show sip-ua history dtmf-relay kpml

Total SIP call legs:2, User Agent Client:1, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID : D0498774-F01311E3-82A0DE9F-78C438FF010.86.176.119
  State of the call : STATE_ACTIVE (?)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 257
No.  Timestamp  Digit  Duration
------------------------------------------------------------------------
Call 2
SIP Call ID : 22BC36A5-F01411E3-81808A6A-5FE95113010.86.176.142
  State of the call : STATE_ACTIVE (?)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 256
No.  Timestamp  Digit  Duration
------------------------------------------------------------------------
Number of SIP User Agent Client(UAC) calls: 2
SIP UAS CALL INFO
Call 1
SIP Call ID : 22BC36A5-F01411E3-81808A6A-5FE95113010.86.176.142
  State of the call : STATE_ACTIVE (?)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 256
No.  Timestamp  Digit  Duration
------------------------------------------------------------------------
Call 2
SIP Call ID : D0498774-F01311E3-82A0DE9F-78C438FF010.86.176.119
  State of the call : STATE_ACTIVE (?)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 257
No.  Timestamp  Digit  Duration
------------------------------------------------------------------------
Number of SIP User Agent Server(UAS) calls: 2

Step 4 show sip-ua history dtmf-relay sip-notify

The following sample output displays SIP call history with SIP Notify DTMF Relay mode.

Example:

Device# show sip-ua history dtmf-relay sip-notify
Total SIP call legs: 2, User Agent Client: 1, User Agent Server: 1
SIP UAC CALL INFO
Call 1
SIP Call ID : 29BB98C-F01311E3-8297DE9F-78C438FF@10.86.176.119
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 252
  No.  Timestamp  Digit Duration
Call 2
SIP Call ID : 550E973B-F01311E3-817A8A6A-5FE95113@10.86.176.142
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 251
  No.  Timestamp  Digit Duration
Number of SIP User Agent Client(UAC) calls: 2
SIP UAS CALL INFO
Call 1
SIP Call ID : 550E973B-F01311E3-817A8A6A-5FE95113@10.86.176.142
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 251
  No.  Timestamp  Digit Duration
Call 2
SIP Call ID : 29BB98C-F01311E3-8297DE9F-78C438FF@10.86.176.119
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 252
  No.  Timestamp  Digit Duration
Number of SIP User Agent Server(UAS) calls: 2
A codec is a device or software capable of encoding or decoding a digital data stream or signal. Audio codecs can code or decode a digital data stream of audio. Video codecs enable compression or decompression of digital video.

CUBE uses codecs to compress digital voice samples to reduce bandwidth usage per call. This chapter describes the basics of encoding digital voice samples using codecs and how to configure them.

Why CUBE Needs Codecs

CUBE uses codecs to compress digital voice samples to reduce bandwidth usage per call. Refer to Table 12: Codec and Bandwidth Information, on page 54 to see the relationship between codec and bandwidth utilization.

Configuring codecs on a device (configured as CUBE) allows the device to act as a demarcation point on a VoIP network and allows a dial peer to be established only if the desired codec criteria are satisfied. Additionally, preferences can be used to determine which codecs are selected over others.

If codec filtering is not required, CUBE also supports transparent codec negotiations. This enables negotiations between endpoints with CUBE leaving the codec information untouched.

The illustrations below show how codec negotiation is performed on CUBE. Two VoIP clouds need to be interconnected. In this scenario, both VoIP 1 and VoIP 2 networks have G.711 a-law configured as the preferred codec.
In the first example, the CUBE router is configured to use the G.729a codec. This can be done by using the appropriate codec command on both VoIP dial peers. When a call is set up, CUBE will accept only G.729a calls, thus influencing the codec negotiation.

In the second example, the CUBE dial peers are configured with a transparent codec and this leaves the codec information contained within the call signaling untouched. Because both VoIP 1 and VoIP 2 have G.711 a-law as their first choice, the resulting call will be a G.711 a-law call.

### Voice Media Transmission

When a VoIP call is established, using the signaling protocols, the digitized voice samples need to be transmitted. These voice samples are often called the voice media. Voice media protocols found in a VoIP environment are the following:

- **Real-Time Transport Protocol (RTP)**—RTP is a Layer 4 protocol that is encapsulated inside UDP segments. RTP carries the actual digitized voice samples in a call.
- **Real-Time Control Protocol (RTCP)**—RTCP is a companion protocol to RTP. Both RTP and RTCP operate at Layer 4 and are encapsulated in UDP. RTP and RTCP typically use UDP ports 16384 to 32767, though these ranges may vary according to hardware platform. However, RTP uses the even port numbers in that range, whereas RTCP uses the odd port numbers. While RTP is responsible for carrying the voice stream, RTCP carries information about the RTP stream such as latency, jitter, packets, and octets sent and received.
- **Compressed RTP (cRTP)**—One of the challenges with RTP is its overhead. Specifically, the combined IP, UDP, and RTP headers are approximately 40 bytes in size, whereas a common voice payload size on a VoIP network is only 20 bytes, which includes 20 ms of voice by default. In that case, the header is twice the size of the payload. cRTP is used for RTP header compression and can reduce the 40-byte header to 2 or 4 bytes in size (depending on whether UDP checksums are in use), as shown in the figure below.
• Secure RTP (sRTP)—To help prevent an attacker from intercepting and decoding or possibly manipulating voice packets, sRTP supports encryption of RTP packets. In addition, sRTP provides message authentication, integrity checking, and protection against replay attacks.

VPN technology like IP Security (IPSec) may be used to protect traffic between sites. Encrypting sRTP traffic at the source of transmission results in encrypting already encrypted traffic, adding significant overhead and bandwidth needs. So it is recommended that sRTP is used for voice traffic, and that this traffic is excluded from IPSec encapsulation. sRTP uses lesser bandwidth, has the same level of security, and can be used by devices at any location because the payload is originated and terminated at the voice endpoint. Because endpoints can be mobile, the security follows the phone.

**Voice Activity Detection**

Voice Activity Detection (VAD) is a technology that works with the human nature of voice conversations, mainly that one person listens while the other talks. VAD classifies traffic as speech, unknown, and silence. Speech and unknown payloads are transported, but silence is dropped. This accounts for approximately 30 percent savings in bandwidth over time.

VAD can significantly reduce the amount of bandwidth required by a media stream. However, VAD has a few negative attributes that need to be considered. Because no packets are sent during silence, the listener can get the impression that the talker has been disconnected. Another characteristic is that it takes a moment for VAD to recognize the speech as having started again, and as a result, the first part of the sentence can be clipped. This can be annoying to the listening party. Music on Hold (MoH) and fax can also cause VAD to become ineffective because the media stream is constant.

VAD is enabled by default in CUBE dial peers as long as the codec selected supports it. VAD can be disabled at the VoIP dial peer using the `no vad` command. Some codecs, such as G.729b and G.729ab, support Comfort Noise Generation (CNG). When VAD is enabled, white noise is played to the listener during times when no packets are received. This leads the listener to believe that background noise is being heard. Cisco IP Phones and most gateways support CNG.

G.729 Annex-B and G.723.1 Annex-A include an integrated VAD function, but otherwise performs the same as G.729 and G.723.1, respectively.

**VoIP Bandwidth Requirements**

The amount of bandwidth required varies by the codec and the transmission media. Two events require bandwidth. The media stream itself requires bandwidth between 17 to 106 kbps depending on codec, header compression, and Layer 2 and 3 headers. In addition, call signaling must be taken into account. While bandwidth required by call signaling is much smaller, it can cause problems on a network due to irregular requirements.
Table 11: Protocol Header Size Assumptions

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Header size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>20 bytes</td>
</tr>
<tr>
<td>UDP</td>
<td>8 bytes</td>
</tr>
<tr>
<td>RTP</td>
<td>12 bytes</td>
</tr>
<tr>
<td>cRTP</td>
<td>Reduces size of IP, UDP, RTP to 2 or 4 bytes</td>
</tr>
</tbody>
</table>

The table below gives calculations for the default voice payload sizes in Cisco CallManager or CUBE. For additional calculations, including different voice payload sizes and other protocols, use the TAC Voice Bandwidth Codec Calculator (registered customers only). For an explanation of each of the column headings, see the table below.

Table 12: Codec and Bandwidth Information

<table>
<thead>
<tr>
<th>Codec &amp; Bit Rate (kbps)</th>
<th>Codec Sample Size (Bytes)</th>
<th>Codec Sample Interval (ms)</th>
<th>Mean Opinion Score (MOS)</th>
<th>Voice Payload Size (Bytes)</th>
<th>Voice Payload Size (ms)</th>
<th>Payload Size (ms) Packets Per Second (PPS)</th>
<th>Bandwidth MP or FRF.12 (kbps)</th>
<th>Bandwidth w/cRTP MP or FRF.12 (kbps)</th>
<th>Bandwidth Ethernet (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711 (64 kbps)</td>
<td>80</td>
<td>10</td>
<td>4.1</td>
<td>160</td>
<td>20</td>
<td>50</td>
<td>82.8</td>
<td>67.6</td>
<td>87.2</td>
</tr>
<tr>
<td>G.729 (8 kbps)</td>
<td>10</td>
<td>10</td>
<td>3.92</td>
<td>20</td>
<td>20</td>
<td>50</td>
<td>26.8</td>
<td>11.6</td>
<td>31.2</td>
</tr>
<tr>
<td>G.723.1 (6.3 kbps)</td>
<td>24</td>
<td>30</td>
<td>3.9</td>
<td>24</td>
<td>30</td>
<td>33.3</td>
<td>18.9</td>
<td>8.8</td>
<td>21.9</td>
</tr>
<tr>
<td>G.723.1 (5.3 kbps)</td>
<td>20</td>
<td>30</td>
<td>3.8</td>
<td>20</td>
<td>30</td>
<td>33.3</td>
<td>17.9</td>
<td>7.7</td>
<td>20.8</td>
</tr>
<tr>
<td>G.726 (32 kbps)</td>
<td>20</td>
<td>5</td>
<td>3.85</td>
<td>80</td>
<td>20</td>
<td>50</td>
<td>50.8</td>
<td>35.6</td>
<td>55.2</td>
</tr>
<tr>
<td>G.726 (24 kbps)</td>
<td>15</td>
<td>5</td>
<td>6.0</td>
<td>60</td>
<td>20</td>
<td>50</td>
<td>42.8</td>
<td>27.6</td>
<td>47.2</td>
</tr>
<tr>
<td>G.728 (16 kbps)</td>
<td>10</td>
<td>5</td>
<td>3.61</td>
<td>60</td>
<td>30</td>
<td>33.3</td>
<td>28.5</td>
<td>18.4</td>
<td>31.5</td>
</tr>
<tr>
<td>G722_64k(64 kbps)</td>
<td>80</td>
<td>10</td>
<td>4.13</td>
<td>160</td>
<td>20</td>
<td>50</td>
<td>82.8</td>
<td>67.6</td>
<td>87.2</td>
</tr>
<tr>
<td>ilbc_mode_20(152 kbps)</td>
<td>38</td>
<td>20</td>
<td>NA</td>
<td>38</td>
<td>20</td>
<td>50</td>
<td>34.0</td>
<td>18.8</td>
<td>38.4</td>
</tr>
<tr>
<td>ilbc_mode_30(133 kbps)</td>
<td>50</td>
<td>30</td>
<td>NA</td>
<td>50</td>
<td>30</td>
<td>33.3</td>
<td>25.867</td>
<td>15.73</td>
<td>28.8</td>
</tr>
</tbody>
</table>

Table 13: Explanation of Terms

<table>
<thead>
<tr>
<th>Codec Bit Rate (kbps)</th>
<th>Based on the codec, this is the number of bits per second that need to be transmitted to deliver a voice call. (codec bit rate = codec sample size / codec sample interval).</th>
</tr>
</thead>
</table>

Cisco Unified Border Element Configuration Guide
Codec Sample Size (Bytes) | Size (Bytes) Based on the codec, this is the number of bytes captured by the digital signal processor (DSP) at each codec sample interval. For example, the G.729 coder operates on sample intervals of 10 ms, corresponding to 10 bytes (80 bits) per sample at a bit rate of 8 kbps. (codec bit rate = codec sample size / codec sample interval).

Codec Sample Interval (ms) | This is the sample interval at which the codec operates. For example, the G.729 coder operates on sample intervals of 10 ms, corresponding to 10 bytes (80 bits) per sample at a bit rate of 8 kbps. (codec bit rate = codec sample size / codec sample interval).

MOS | MOS is a system of grading the voice quality of telephone connections. With MOS, a wide range of listeners judge the quality of a voice sample on a scale of one (bad) to five (excellent). The scores are averaged to provide the MOS for the codec.

Voice Payload Size (Bytes) | The voice payload size represents the number of bytes (or bits) that are filled into a packet. The voice payload size must be a multiple of the codec sample size. For example, G.729 packets can use 10, 20, 30, 40, 50, or 60 bytes of voice payload size.

Voice Payload Size (ms) | Payload Size (ms) The voice payload size can also be represented in terms of the codec samples. For example, a G.729 voice payload size of 20 ms (two 10 ms codec samples) represents a voice payload of 20 bytes \[ (20 \text{ bytes} * 8) / (20 \text{ ms}) = 8 \text{ kbps} \]

PPS | PPS represents the number of packets that need to be transmitted every second in order to deliver the codec bit rate. For example, for a G.729 call with voice payload size per packet of 20 bytes (160 bits), 50 packets need to be transmitted every second \[ 50 \text{ pps} = (8 \text{ kbps}) / (160 \text{ bits per packet}) \]

## Supported Audio and Video Codecs

CUBE is required to support the codec used between endpoints. g729r8 is supported by default. All other codecs have to be configured. The following codecs are supported:

### Table 14: Audio Codecs Supported on CUBE

<table>
<thead>
<tr>
<th>Codec Keyword</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>aacld</td>
<td>AAACL 90000 bps</td>
</tr>
<tr>
<td>clear-channel</td>
<td>Clear Channel 64000 bps (No voice capabilities: data transport only)</td>
</tr>
<tr>
<td>Codec Keyword</td>
<td>Codec</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>g711alaw</td>
<td>G.711 A Law 64000 bps</td>
</tr>
<tr>
<td>g711ulaw</td>
<td>G.711 u Law 64000 bps</td>
</tr>
<tr>
<td>g722-48</td>
<td>G722-48K 64000 bps - Only supported for H.320&lt;&gt;H.323 calls</td>
</tr>
<tr>
<td>g722-56</td>
<td>G722-56K 64000 bps - Only supported for H.320&lt;&gt;H.323 calls</td>
</tr>
<tr>
<td>g722-64</td>
<td>G722-64K 64000 bps</td>
</tr>
<tr>
<td>g723ar53</td>
<td>G.723.1 ANNEX-A 5300 bps (contains built-in VAD that cannot be disabled)</td>
</tr>
<tr>
<td></td>
<td>Not supported on PVDM3.</td>
</tr>
<tr>
<td>g723ar63</td>
<td>G.723.1 ANNEX-A 6300 bps (contains built-in VAD that cannot be disabled)</td>
</tr>
<tr>
<td></td>
<td>Not supported on PVDM3.</td>
</tr>
<tr>
<td>g723r53</td>
<td>G.723.1 5300 bps</td>
</tr>
<tr>
<td></td>
<td>Not supported on PVDM3.</td>
</tr>
<tr>
<td>g723r63</td>
<td>G.723.1 6300 bps</td>
</tr>
<tr>
<td></td>
<td>Not supported on PVDM3.</td>
</tr>
<tr>
<td>g726r16</td>
<td>G.726 16000 bps</td>
</tr>
<tr>
<td>g726r24</td>
<td>G.726 24000 bps</td>
</tr>
<tr>
<td>g726r32</td>
<td>G.726 32000 bps</td>
</tr>
<tr>
<td>g728</td>
<td>G.728 16000 bps</td>
</tr>
<tr>
<td>g729br8</td>
<td>G.729 ANNEX-B 8000 bps (contains built-in VAD that cannot be disabled)</td>
</tr>
<tr>
<td>g729r8</td>
<td>G.729 8000 bps</td>
</tr>
<tr>
<td>gsmamr-nb</td>
<td>GSM AMR-NB 4750 to 12200 bps (contains built-in VAD that cannot be disabled)</td>
</tr>
<tr>
<td>ilbc</td>
<td>iLBC 13330 or 15200 bps</td>
</tr>
<tr>
<td>isac</td>
<td>iSAC 10 to 32 kbps (variable bit-rate)</td>
</tr>
<tr>
<td>mp4a-latm</td>
<td>MP4A-LATM upto 128 kbps</td>
</tr>
<tr>
<td>transparent</td>
<td>Transparent; uses the endpoint codec</td>
</tr>
</tbody>
</table>
Table 15: Video Codecs Supported on CUBE

<table>
<thead>
<tr>
<th>Codec Keyword</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>h261</td>
<td>Video Codec H261</td>
</tr>
<tr>
<td>h263</td>
<td>Video Codec H263</td>
</tr>
<tr>
<td>h263+</td>
<td>Video Codec H263+</td>
</tr>
<tr>
<td>h264</td>
<td>Video Codec H264</td>
</tr>
<tr>
<td>mpeg4</td>
<td>Video Codec MPEG-4 ISO/IES 14496-2</td>
</tr>
</tbody>
</table>

How to Configure Codecs

Configuring Audio and Video Codecs at the Dial Peer Level

SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice number voip
4. Enter one of the following to configure an audio codec:
   - `codec codec [bytes payload-size fixed-bytes ]`
   - `codec isac [mode {adaptive | independent} [bit-rate value framesize { 30 | 60 } [fixed] ]`
   - `codec ilbc [mode frame-size [bytes payload-size]]`
   - `codec mp4-latm [profile tag]`
5. Do the following to configure a video codec:
   - `video codec codec`
6. (Optional) Do one of the following to configure RTP payload type:
   - `rtppayload-type cisco-codec-isac value`
   - `rtppayload-type cisco-codec-ilbc value`
   - `rtppayload-type cisco-codec-video-h263+ value`
   - `rtppayload-type cisco-codec-video-h264 value`
7. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Device&gt; enable</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>configure terminal</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Enters global configuration mode.</strong></td>
</tr>
<tr>
<td><strong>Device&gt; configure terminal</strong></td>
<td><strong>Step 2</strong></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>dial-peer voice number voip</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Enters dial peer configuration mode for the specified VoIP dial peer.</strong></td>
</tr>
<tr>
<td><strong>Device(config)# dial-peer voice 1 voip</strong></td>
<td><strong>Step 3</strong></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>Enter one of the following to configure an audio codec:</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>codec codec [bytes payload-size fixed-bytes]</strong></td>
</tr>
<tr>
<td></td>
<td>- **codec isac [mode {adaptive</td>
</tr>
<tr>
<td></td>
<td>- <strong>codec ilbc [mode frame-size [bytes payload-size]]</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>codec mp4-latm [profile tag]</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Configures an audio codec at the dial peer level.</strong></td>
</tr>
<tr>
<td><strong>For g711alaw Codec</strong></td>
<td><strong>• g729r8, 20-byte payload is configured by default.</strong></td>
</tr>
<tr>
<td><strong>Device(config-dial-peer)# codec g711alaw</strong></td>
<td><strong>Step 4</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Configures a video codec at the dial peer level.</strong></td>
</tr>
<tr>
<td><strong>For ISAC Codec</strong></td>
<td><strong>Step 5</strong></td>
</tr>
<tr>
<td><strong>Device(config-dial-peer)# codec isac mode independent</strong></td>
<td><strong>Do the following to configure a video codec:</strong></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>• video codec codec</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Configures the RTP payload type.</strong></td>
</tr>
<tr>
<td><strong>For Video Codec</strong></td>
<td><strong>Step 6</strong></td>
</tr>
<tr>
<td><strong>Device(config-dial-peer)# video codec h261</strong></td>
<td><strong>(Optional) Do one of the following to configure RTP payload type:</strong></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>• rtp payload-type cisco-codec-isac value</strong></td>
</tr>
<tr>
<td></td>
<td><strong>• rtp payload-type cisco-codec-ilbc value</strong></td>
</tr>
<tr>
<td></td>
<td><strong>• rtp payload-type cisco-codec-video-h263+ value</strong></td>
</tr>
<tr>
<td></td>
<td><strong>• rtp payload-type cisco-codec-video-h264 value</strong></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Returns to privileged EXEC mode.</strong></td>
</tr>
<tr>
<td><strong>Device(config-dial-peer)# end</strong></td>
<td><strong>Step 7</strong></td>
</tr>
</tbody>
</table>
Configuring Audio Codecs Using a Codec Voice Class and Preference Lists

Preferences can be used to determine which codecs will be selected over others.

A codec voice class is a construct within which a codec preference order can be defined. A codec voice class can then be applied to a dial peer, which then follows the preference order defined in the codec voice class.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice class codec tag
4. Do the following for each audio codec you want to configure in the voice class:
   - codec preference value codec-type[bytes payload-size fixed-bytes]
   - codec preference value isac [mode {adaptive | independent}] [bit-rate value framesize {30 | 60} [fixed]]
   - codec preference value ilbc [mode frame-size [bytes payload-size]]
   - codec preference value mp4-latm [profile tag]
5. exit
6. dial-peer voice number voip
7. voice-class codec tag offer-all
8. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2</th>
<th>configure terminal</th>
<th>Enters global configuration mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example:</td>
<td>Device&gt; configure terminal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3</th>
<th>voice class codec tag</th>
<th>Enters voice-class configuration mode for the specified codec voice class.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example:</td>
<td>Device(config)# voice class codec 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 4</th>
<th>Do the following for each audio codec you want to configure in the voice class:</th>
<th>Configure a codec within the voice class and specifies a preference for the codec. This becomes part of a preference list</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• codec preference value codec-type[bytes payload-size fixed-bytes]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• codec preference value isac [mode {adaptive</td>
<td>independent}] [bit-rate value framesize {30</td>
</tr>
<tr>
<td></td>
<td>• codec preference value ilbc [mode frame-size [bytes payload-size]]</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose
- **Command or Action**: `• codec preference value mp4-latm [profile tag]`

### Step 5
- **Command or Action**: `exit`
- **Example**: Device(config-class)# exit
- **Purpose**: Exits the current mode.
  - Enter your password if prompted.

### Step 6
- **Command or Action**: `dial-peer voice number voip`
- **Example**: Device(config)# dial-peer voice 1 voip
- **Purpose**: Enters dial peer configuration mode for the specified VoIP dial peer.

### Step 7
- **Command or Action**: `voice-class codec tag offer-all`
- **Example**: Device(config-dial-peer)# voice-class codec 10
- **Purpose**: Applies the previously configured voice class and associated codecs to a dial peer.
  - The `offer-all` keyword allows the device to offer all codecs configured in a codec voice class.

### Step 8
- **Command or Action**: `end`
- **Example**: Device(config-dial-peer)# end
- **Purpose**: Returns to privileged EXEC mode.

---

### Configuring Video Codecs Using Codec Voice Class

**SUMMARY STEPS**
1. enable
2. configure terminal
3. voice class codec tag
4. video codec codec
5. exit
6. dial-peer voice number voip
7. voice-class codec tag offer-all
8. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example</strong>: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example</strong>: Device&gt; configure terminal</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 3</strong> voice class codec tag</td>
<td>Enters voice-class configuration mode for the specified codec voice class.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice class codec 10</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> video codec codec</td>
<td>Configures a video codec within the voice class.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>video codec h261</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# exit</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 6</strong> dial-peer voice number voip</td>
<td>Enters dial peer configuration mode for the specified VoIP dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 1 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> voice-class codec tag offer-all</td>
<td>Applies the previously configured codec voice class and associated codecs to a dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class codec 10</td>
<td>• The offer-all keyword allows the device to offer all codecs configured in the codec voice class.</td>
</tr>
<tr>
<td><strong>Step 8</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

## Verifying an Audio Call

### SUMMARY STEPS

1. **show call active voice [compact]**

### DETAILED STEPS

**show call active voice [compact]**

Displays a compact version of call information for voice calls in progress.

**Example:**

Device# show call active voice compact

```
<callID>    A/O FAX T<sec> Codec    type   Peer Address     IP R<ip>:<udp>
Total call-legs: 2

  23 ANS       T3    mp4a-latm    VOIP       Pepp   9.45.33.11:57210
  24 ORG       T3    mp4a-latm    VOIP       P123   9.45.33.11:57210
```

**Example:**
Device# show call active voice compact

<table>
<thead>
<tr>
<th>&lt;callID&gt;</th>
<th>A/O FAX</th>
<th>Codec</th>
<th>type</th>
<th>Peer Address</th>
<th>IP R&lt;ip&gt;:&lt;udp&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>58 ANS</td>
<td>T11</td>
<td>g711ulaw</td>
<td>VOIP</td>
<td>Psipp 2001:...:230A:6080</td>
<td></td>
</tr>
<tr>
<td>59 ORG</td>
<td>T11</td>
<td>g711ulaw</td>
<td>VOIP</td>
<td>P50001100111</td>
<td>10.13.37.150:6090</td>
</tr>
</tbody>
</table>

---

Configuration Examples for Codecs

**Example: Configuring a Codec at Dial-Peer Level**

```
Device(config)# dial-peer voice 5550199 voip
Device(config-dial-peer)# incoming called-number 5550199
Device(config-dial-peer)# codec g711ulaw
Device(config-dial-peer)# end
```

**Example: Configuring a Codec Preference List and Applying it to a Dial Peer**

```
Device(config)# voice class codec 100
Device(config-dial-peer)# codec preference 1 g711ulaw
Device(config-dial-peer)# exit
Device(config)# dial-peer voice 10 voip
Device(config-dial-peer)# voice-class codec 100
Device(config-dial-peer)# end
```
Call Admission Control

The call admission control feature enables you to control the audio quality and video quality of calls over a wide-area (IP WAN) link by limiting the number of calls that are allowed on that link at the same time. Audio and video quality can begin to degrade when too many active calls exist on a link and the amount of bandwidth is oversubscribed. Call admission control regulates audio and video quality by limiting the number of calls that can be active on a particular link at the same time.

The Call Admission Control feature controls number of calls based on resources and bandwidth, proactively reserve resources for good quality video calls, ensures that traffic adheres to QoS policies within each network.

CUBE provides different CAC mechanisms that are based on:

- Total Calls, CPU, or Memory
- Call Spike Detection
- Maximum Calls per Destination
- Dial-peer or Interface Bandwidth

- Configuring CAC Based on Total Calls, CPU or Memory, on page 63
- Configuring CAC Based on Call Spike Detection, on page 64
- Configuring CAC Based on Maximum Calls per Destination, on page 66
- Bandwidth-Based Call Admission Control, on page 67

Configuring CAC Based on Total Calls, CPU or Memory

The Call Admission Control (CAC) based on CPU Utilization feature permits the Cisco Voice Gateways to deny incoming calls exceeding a pre-configured threshold, permitting the selection of a system CPU load level value.

The ‘Call Threshold’ command allows you to configure two thresholds, high and low. The ‘Call Treatment’ is triggered when the current value of a resource goes beyond the configured high value. The ‘Call Treatment’ remains in effect until the current resource value falls below the configured low value.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. **call threshold global** [cpu-5sec | cpu-avg | io-mem | proc-mem | total-calls | total-mem] *low*
   low-threshold **high** high-threshold
4. **call treatment on**
5. **end**

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt;.enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> call threshold global [cpu-5sec</td>
<td>cpu-avg</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# call threshold global total-calls low 1 high 1</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>Device(config)# call threshold global cpu-avg low 75 high 85</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>Device(config)# call threshold global total-mem low 75 high 85</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> call treatment on</td>
<td>Enables the call treatment feature.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# call treatment on</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits global configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# end</td>
<td></td>
</tr>
</tbody>
</table>

**Configuring CAC Based on Call Spike Detection**

The Call Admission Control (CAC) based on Call Spike Detection feature permits the Cisco Voice Gateways to monitor call arrival rate over a moving window of time. Calls exceeding the configured rate threshold are
rejected. This feature helps in protecting against unexpected high call volumes, and INVITE-based DOS attacks.

You can configure this feature globally or on a per dial-peer level. An error code is sent when a call spike occurs, the error code is configurable globally or on a per dial-peer level.

**SUMMARY STEPS**

1. enable  
2. configure terminal  
3. call spike threshold  
   call number <1-2147483647> steps<3-10> size<100-250>  
4. call treatment on  
5. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  Example:  
  Device>enable |  
| **Step 2** configure terminal | Enters global configuration mode.  
  Example:  
  Device# configure terminal |  
| **Step 3** call spike threshold  
  call number <1-2147483647> steps<3-10> size<100-250> | Configures the Call Spike Call Admission Control feature at the device level to reject SIP calls when the call spike is detected as per the configuration (10 incoming call requests per 300 milliseconds)  
  Example:  
  Device(config)# call spike 10 steps 3 size 100  
  Device(config)# call spike 12 |  
| **Step 4** call treatment on | Enables the call treatment feature.  
  Example:  
  Device(config)# call treatment on |  
| **Step 5** end | Exits global configuration mode and enters privileged EXEC mode.  
  Example:  
  Device(config)# end |
Configuring CAC Based on Maximum Calls per Destination

The Call Admission Control (CAC) based on Maximum Calls per Destination feature permits the Cisco Voice Gateways to restricting the number of concurrent calls that can be active on a VoIP dial peer. Maximum connections work on individual dial-peers and does not provide CAC for the entire gateway.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `dial-peer voice tag voip`
4. `session protocol sipv2`
5. `max-conn`
6. `end`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>dial-peer voice tag voip</code></td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# dial-peer voice 10 voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>session protocol sipv2</code></td>
<td>Configures SIP as the session protocol type.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# session protocol sipv2</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> <code>max-conn</code></td>
<td>Configures the Maximum Calls per Destination Call Admission Control feature at the device level to allow only 2 long-distance calls.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# max-conn &lt;1-214748364&gt;</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> <code>end</code></td>
<td>Exits global configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device# end</code></td>
<td></td>
</tr>
</tbody>
</table>
Bandwidth-Based Call Admission Control

The Bandwidth-Based Call Admission Control (CAC) feature provides the functionality to reject SIP calls when the bandwidth accounted by the SIP signaling layer exceeds the aggregate bandwidth threshold for VoIP media traffic—voice, video, and fax. This functionality helps you prevent Quality of Service (QoS) degradation of VoIP media traffic for existing calls when the bandwidth allocated for VoIP traffic is fully utilized. The Bandwidth-Based Call Admission Control feature is supported on Session Initiation Protocol (SIP) trunks of the Time Division Multiplexing (TDM) SIP gateway and the Cisco Unified Border Element (Cisco UBE).

Midcall media renegotiation can also be rejected if the configured maximum bandwidth threshold for the VoIP media traffic is exceeded. The call continues as per the previously negotiated media codecs if midcall media renegotiation is rejected.

The excess subscription of the bandwidth allocated for VoIP traffic results in VoIP media packets being dropped or delayed, irrespective of the VoIP call to which they belong. Under such circumstances, it is better to deny new calls to prevent QoS deterioration for existing VoIP call traffic. The existing traffic congestion resolution mechanisms do not differentiate between media packets of existing calls (admitted) and new calls (oversubscribed). Similarly, existing call signaling is unaware of the media traffic congestion. The Bandwidth-Based Call Admission Control feature fills this gap by rejecting new SIP calls when the bandwidth allocated for VoIP traffic is fully utilized. The actual bandwidth usage is not measured and policed. The lower-level QoS policies control the traffic characteristics for the specified traffic class.

**Note**
The Bandwidth-Based Call Admission Control feature is applicable only to VoIP traffic.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

Restrictions for Bandwidth-Based Call Admission Control

- Cisco UBE, configured with the Bandwidth-Based Call Admission Control feature, will not reject the call if the bandwidth of the SDP answer is greater than the bandwidth of the SDP offer.
- Layer 2 overhead is not included in the bandwidth calculation.
- A midcall delayed-offer (DO) to DO call is disconnected if the bandwidth requested in an offer message (200 OK) exceeds the threshold bandwidth.
- Real Time Transport Control Protocol (RTCP) and RTP Named Telephone Event (RTP-NTE) bandwidth requirement is not computed.
- The Bandwidth-Based Call Admission Control feature does not support:
  - Cisco fax relay.
Information About Bandwidth-Based Call Admission Control

Maximum Bandwidth Calculation

The bandwidth requirement for each SIP call leg is calculated using the codec information available in the SDP. Here, the actual media bandwidth used is not measured.

\[
\text{Bandwidth in Kbps (Kilo bits per second)} = \left[ \text{codec bytes} + \text{RTP header (12)} + \text{UDP (8)} + \text{IP Header (20 or 40)} \right] \times \text{Packets per second} \times 8/1000
\]

Where, codec bytes = Codec payload size, in bytes, for a given packetization interval.

RTP header = Size of the RTP header, in bytes.

UDP = Size of the UDP header, in bytes.

IP Header = Size of the IP header, in bytes. The IPV4 header is 20 bytes and the IPV6 header is 40 bytes.

Packets per second = Number of RTP packets sent or received per second. This value is as per the negotiated packetization interval. The SDP media attribute "ptime" indicates the number of packets per second.

Bandwidth Tables

This section provides the sample maximum bandwidth calculation for audio and fax calls.

Table 16: Audio Bandwidth Table

<table>
<thead>
<tr>
<th>Codec and Bit Rate (Kbps)</th>
<th>Codec Sample Size in Bytes</th>
<th>Voice Payload Size in Bytes</th>
<th>Voice Payload Size in Milliseconds</th>
<th>Packets Per Second</th>
<th>Bandwidth for IPv4 (excluding Layer 2) in Kbps</th>
<th>Bandwidth for IPv6 (excluding Layer 2) in Kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711 (64 Kbps)</td>
<td>80</td>
<td>160</td>
<td>20</td>
<td>50</td>
<td>80</td>
<td>88</td>
</tr>
<tr>
<td>G.729 (8 Kbps)</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>50</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>G.723.1 (6.3 Kbps)</td>
<td>24</td>
<td>24</td>
<td>30</td>
<td>33.3</td>
<td>17</td>
<td>22</td>
</tr>
<tr>
<td>G.723.1 (5.3 Kbps)</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>33.3</td>
<td>16</td>
<td>21</td>
</tr>
<tr>
<td>G.726 (32 Kbps)</td>
<td>20</td>
<td>80</td>
<td>20</td>
<td>50</td>
<td>48</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Maximum Bandwidth in Kbps</td>
<td>Redundancy</td>
<td>T.38 Fax Bit Rate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
<td>------------</td>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>None</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2400 (default)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Redundancy</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>None</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Redundancy</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9600 (default)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9600 (default)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 17: Fax Bandwidth Table
How to Configure Bandwidth-Based Call Admission Control

Configuring Bandwidth-Based Call Admission Control at the Interface Level

You can configure the Bandwidth-Based Call Admission Control feature at the interface level to reject SIP calls when the bandwidth required for the call exceeds the aggregate bandwidth threshold.

You can configure the Bandwidth-Based Call Admission Control feature for the following interfaces:

- ATM
- Ethernet (Fast Ethernet, Gigabit Ethernet)
- Loopback
- Serial

Cisco recommends that you configure a bind media to associate a specific interface for SIP calls. Otherwise, the interface used for the calls will be determined based on the best local address that can access the remote media source address (for early offer calls) or the remote signaling source address (for delayed offer calls). When you use a Loopback interface to configure CAC, you must configure an additional bind-to-bind media with the Loopback interface at the global level or the dial peer level. Configure the bind media source-interface loopback number command in service SIP configuration mode to configure a bind media.

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `call threshold interface type number int-bandwidth [class-map name [l2-overhead percentage] | low low-threshold high high-threshold] [midcall-exceed]`
4. `end`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>
Purpose

Enters global configuration mode.

**Step 2**

configure terminal

Example:

Device# configure terminal

**Step 3**

call threshold interface type number int-bandwidth
{class-map name [l2-overhead percentage] | low
low-threshold high high-threshold} [midcall-exceed]

Example:

Device(config)# call threshold interface
GigabitEthernet 0/0 int-bandwidth low 1000 high 20000 midcall-exceed

or

Device(config)# call threshold interface
GigabitEthernet 0/0 int-bandwidth class-map
voip-traffic l2-overhead 20 midcall-exceed

**Step 4**

device

Example:

Device(config)# device

**Purpose**

Enters global configuration mode.

**Example:**

Device(config)# device

**Configuring Bandwidth-Based Call Admission Control at the Dial Peer Level**

You can configure the Bandwidth-Based Call Admission Control feature at the dial peer level to reject SIP calls when the bandwidth required for the calls exceeds the aggregate bandwidth threshold.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice tag voip
4. session protocol sipv2
5. max-bandwidth bandwidth-value [midcall-exceed]
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice tag voip</td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 44 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> session protocol sipv2</td>
<td>Configures the Bandwidth-Based Call Admission Control feature for SIP dial peers only.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> max-bandwidth bandwidth-value [midcall-exceed]</td>
<td>Configures the Bandwidth-Based Call Admission Control feature at the dial peer level to reject SIP calls when the bandwidth required for the calls exceed the aggregate bandwidth threshold.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Configuring the midcall-exceed keyword allows exceeding the bandwidth threshold during mid-call media renegotiation. Media renegotiation exceeding the bandwidth threshold is rejected by default.</td>
</tr>
<tr>
<td>Device(config-dial-peer)# max-bandwidth 24 midcall-exceed</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Exits dial peer configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

Configuring the Bandwidth-Based Call Admission Control SIP Error Response Code Mapping

Mapping of the call rejection cause code to a specific SIP error response code is known as error response code mapping. The cause code for the call rejected because of the bandwidth-based CAC can be mapped to a SIP error response code between 400 to 600. The default SIP error response code is 488.

You can configure SIP error response codes for calls rejected by the Bandwidth-Based Call Admission Control feature at the global level, dial peer level, or both.
## Configuring Bandwidth-Based Call Admission Control SIP Error Response Code Mapping at the Global Level

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `voice service voip`
4. `sip`
5. `error-code-override cac-bandwidth failure sip-status-code-number`
6. `end`

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice-service configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters service SIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> error-code-override cac-bandwidth failure sip-status-code-number</td>
<td>Configures bandwidth-based CAC SIP error response code mapping at the global level.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# error-code-override cac-bandwidth failure 500</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Exits service SIP configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>
Configuring Bandwidth-Based Call Admission Control SIP Error Response Code Mapping at the Dial Peer Level

SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice tag {pots | voatm | vofr | voip}
4. voice-class sip error-code-override cac-bandswidth failure {sip-status-code-number | system}
5. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice tag {pots</td>
<td>voatm</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 88 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class sip error-code-override cac-bandswidth failure {sip-status-code-number</td>
<td>system}</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip error-code-override cac-bandswidth failure 500</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits dial peer configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

Verifying Bandwidth-Based Call Admission Control

Perform this task to verify the configuration for the Bandwidth-Based Call Admission Control feature on Cisco UBE. The show commands need not be entered in any specific order.

SUMMARY STEPS

1. enable
2. show call threshold config
3. show call threshold status
4. show call threshold stats
5. show dial-peer voice

DETAILED STEPS

**Step 1**

**enable**

_example:

Device>enable

Enables privileged EXEC mode.

**Step 2**

**show call threshold config**

_example:

Device# show call threshold config

Some resource polling interval:
  CPU_AVG interval: 60
  Memory interval: 5

<table>
<thead>
<tr>
<th>IF</th>
<th>Type</th>
<th>Value</th>
<th>Low</th>
<th>High</th>
<th>Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigabitEthernet0/0</td>
<td>int-bandwidth</td>
<td>0</td>
<td>100</td>
<td>400</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Displays the current call threshold configuration at the interface level for all resources.

**Step 3**

**show call threshold status**

_example:

Device# show call threshold status

<table>
<thead>
<tr>
<th>Status</th>
<th>IF</th>
<th>Type</th>
<th>Value</th>
<th>Low</th>
<th>High</th>
<th>Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avail</td>
<td>GigabitEthernet0/0</td>
<td>int-bandwidth</td>
<td>0</td>
<td>100</td>
<td>400</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Displays the availability status of resources that are configured when the Bandwidth-Based Call Admission Control feature is enabled at an interface level.

**Step 4**

**show call threshold stats**

_example:

Device# show call threshold stats

Total resource check: 2
successful: 1
failed:  1

1: ------------------------
Failed resources: int-bandwidth,
related interface: GigabitEthernet0/0; related option:N/A
Recorded time: 04:49:39 UTC Wed Dec 8 2010

2: ------------------------
Successful
Displays the statistics of resources that are configured when the Bandwidth-Based Call Admission Control feature is enabled at an interface level.

**Step 5**

**show dial-peer voice**

**Example:**

```
Device# show dial-peer voice

incoming called-number = `2000', connections/maximum = 0/unlimited, bandwidth/maximum = 0/400,
......
Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
Accepted Calls = 3, Refused Calls = 0,
Bandwidth CAC Accepted Calls = 3, Bandwidth CAC Refused Calls = 0
```

Displays information for the voice dial peer.

---

**Troubleshooting Tips**

The following commands can help troubleshoot the Bandwidth-Based Call Admission Control feature:

- `debug csip all`
- `debug voice ccapi all`

**Configuration Examples for Bandwidth-Based Call Admission Control**

**Example: Configuring Bandwidth-Based Call Admission Control at the Interface Level**

The following example shows how to configure Cisco UBE to reject new SIP calls if the accounted VoIP media bandwidth on Gigabit Ethernet interface 0/0 exceeds 400 Kbps of bandwidth and continues to have a bandwidth above 100 Kbps:

```
Device> enable
Device# configure terminal
Device(config)# call threshold interface GigabitEthernet 0/0 int-bandwidth low 100 high 400
```

The following example shows how to configure Cisco UBE to reject new SIP calls if the VoIP media bandwidth on Gigabit Ethernet interface 0/0 exceeds the configured bandwidth for priority traffic in the “voip_traffic” class:

```
Device>enable
Device# configure terminal
Device(config)# class-map match-all voip-traffic

Device(config-cmap)# policy-map voip-policy
Device(config-pmap)# class voip-traffic
Device(config-pmap-c)# priority 440
Device(config-pmap-c)# end
```
Device# enable
Device# configure terminal
Device(config)# call threshold interface GigabitEthernet 0/0 int-bandwidth class-map
voip-traffic l2-overhead 10

**Note**
Layer 2 overhead of 10 percent in the `call threshold` command indicates that the IP bandwidth, excluding Layer 2, is 90 percent of the configured priority bandwidth.

### Example: Configuring Bandwidth-Based Call Admission Control at the Dial Peer Level

The following example shows how to configure Cisco UBE to reject calls once the accounted aggregate bandwidth of active calls exceeds 400 Kbps for a SIP dial peer:

```
Device> enable
Device# configure terminal
Device(config)# dial-peer voice 2000 voip
Device(config)# session protocol sipv2
Device(config-dial-peer)# max-bandwidth 400
```

### Example: Configuring the Bandwidth-Based Call Admission Control SIP Error Response Code Mapping at the Global Level

The following example shows how to configure Cisco UBE for bandwidth-based CAC SIP error response code mapping at the global level:

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(config-voi-serv)# sip
Device(config-serv-sip)# error-code-override cac-bandwidth 500
```

### Example: Configuring the Bandwidth-Based Call Admission Control SIP Error Response Code Mapping at the Dial Peer Level

The following example shows how to configure Cisco UBE for bandwidth-based CAC SIP error response code mapping at the dial peer level:

```
Device> enable
Device# configure terminal
Device(config)# dial-peer voice 88 voip
Device(config-dial-peer)# voice-class sip error-code-override cac-bandwidth failure 500
```
Feature Information for Bandwidth-Based Call Admission Control

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

### Table 18: Feature Information for Bandwidth-Based Call Admission Control

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth-Based Call Admission Control</td>
<td>15.2(2)T</td>
<td>The Bandwidth-Based Call Admission Control feature provides the functionality to reject SIP calls when the bandwidth accounted by the SIP signaling layer exceeds the aggregate bandwidth threshold for VoIP media traffic—voice, video, and fax. This functionality helps prevent QoS degradation of VoIP media traffic for existing calls when the bandwidth allocated for VoIP traffic is fully utilized. The following commands were introduced or modified: call threshold interface, error-code-override, max-bandwidth, show call threshold, voice-class sip</td>
</tr>
<tr>
<td>Bandwidth-Based Call Admission Control</td>
<td>Cisco IOS XE Release 3.7S</td>
<td>The Bandwidth-Based Call Admission Control feature provides the functionality to reject SIP calls when the bandwidth accounted by the SIP signaling layer exceeds the aggregate bandwidth threshold for VoIP media traffic—voice, video, and fax. This functionality helps prevent QoS degradation of VoIP media traffic for existing calls when the bandwidth allocated for VoIP traffic is fully utilized. The following commands were introduced or modified: call threshold interface, error-code-override, max-bandwidth, show call threshold, voice-class sip</td>
</tr>
</tbody>
</table>
This chapter provides basic configuration information for the following features:

- SIP Register Support
- SIP Redirect Processing Enhancement
- SIP 300 Multiple Choice Messages
- SIP implementation enhancements:
  - Interaction with Forking Proxies
  - SIP Intra-Gateway Hairpinning

Finding Support Information for Platforms and Cisco Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

- Prerequisites for Basic SIP Configuration, on page 79
- Restrictions for Basic SIP Configuration, on page 79
- Information About Basic SIP Configuration, on page 80
- How to Perform Basic SIP Configuration, on page 81
- Configuration Examples for Basic SIP Configuration, on page 97
- Toll Fraud Prevention, on page 104

Prerequisites for Basic SIP Configuration

SIP Redirect Processing Enhancement Feature

- Ensure that your SIP gateway supports 300 or 302 Redirect messages.

Restrictions for Basic SIP Configuration

- If Hot Standby Router Protocol (HSRP) is configured on the Cisco IOS Gateway, IP-TDM calls are not supported.
Information About Basic SIP Configuration

SIP Register Support

With H.323, Cisco IOS gateways can register E.164 numbers of a POTS dial peer with a gatekeeper, which informs the gatekeeper of a user’s contact information. Session Initiation Protocol (SIP) gateways allow the same functionality, but with the registration taking place with a SIP proxy or registrar. SIP gateways allow registration of E.164 numbers to a SIP proxy or registrant on behalf of analog telephone voice ports (FXS), IP phone virtual voice ports (EFXS), and local SCCP phones.

When registering dial peers with an external registrar, you can also register with a secondary SIP proxy or registrar to provide redundancy. The secondary registration can be used if the primary registrar fails.

SIP gateways allow registration of E.164 numbers to a SIP proxy or registrar server on behalf of analog telephone voice ports (FXS), IP phone virtual voice ports (EFXS), and local SCCP phones. By default, SIP gateways do not generate SIP Register messages. The following tasks set up the gateway to register E.164 telephone numbers with an external SIP registrar.

Note

There are no commands that allow registration between the H.323 and SIP protocols.

SIP Redirect Processing Enhancement

SIP Redirect Processing allows flexibility in the handling of incoming redirect or 3xx class of responses. Redirect responses can be enabled or disabled through the command-line interface, providing a benefit to service providers who deploy Cisco SIP gateways. Redirect processing is active by default, which means that SIP gateways handle incoming 3xx messages in compliance with RFC 2543. RFC 2543 states that redirect response messages are used by SIP user agents to initiate a new Invite when a user agent learns that a user has moved from a previously known location.

In accordance with RFC 2543-bis-04, the processing of 3xx redirection is as follows:

- The uniform resource identifier (URI) of the redirected INVITE is updated to contain the new contact information provided by the 3xx redirect message.
- The transmitted CSeq number found in the CSeq header is increased by one. The new INVITE includes the updated CSeq.
- The To, From, and Call ID headers that identify the call leg remain the same. The same Call ID gives consistency when capturing billing history.
- The UAC retries the request at the new address given by the 3xx Contact header field.

Redirect handling can be disabled by using the no redirection command in SIP user-agent configuration mode. In this case, the user agent treats incoming 3xx responses as 4xx error class responses. The call is not redirected, and is instead released with the appropriate PSTN cause-code message. The table below shows the mapping of 3xx responses to 4xx responses.
Table 19: Mapping of 3xx Responses to 4xx Responses

<table>
<thead>
<tr>
<th>Redirection (3xx) Response Message</th>
<th>Mapping to 4xx (Client Error) Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>300 Multiple choices</td>
<td>410 Gone</td>
</tr>
<tr>
<td>301 Moved Permanently</td>
<td>410 Gone</td>
</tr>
<tr>
<td>302 Moved Temporarily</td>
<td>480 Temporarily Unavailable</td>
</tr>
<tr>
<td>305 Use Proxy</td>
<td>410 Gone</td>
</tr>
<tr>
<td>380 Alternative Service</td>
<td>410 Gone</td>
</tr>
<tr>
<td>&lt;any other 3xx response&gt;</td>
<td>410 Gone</td>
</tr>
</tbody>
</table>

SIP Redirect Processing generates call history information with appropriate release cause codes that may be used for accounting or statistics purposes. When a 3xx response is mapped to a 4xx class of response, the cause code stored in call history is based on the mapped 4xx response code.

Call redirection must be enabled on the gateway for SIP call transfer involving redirect servers to be successful. The Cisco IOS voice gateway can also use call redirection if an incoming VoIP call matches an outbound VoIP dial peer. The gateway sends a 300 or 302 Redirect message to the call originator, allowing the originator to reestablish the call. Two commands allow you to enable the redirect functionality, globally or on a specific inbound dial peer: `redirect ip2ip (dial-peer)` and `redirect ip2ip (voice service)`.

### Sending SIP 300 Multiple Choice Messages

Originally, when a call was redirected, the SIP gateway would send a 302 Moved Temporarily message. The first longest match route on a gateway (dial-peer destination pattern) was used in the Contact header of the 302 message. Now, if multiple routes to a destination exist for a redirected number (multiple dial peers are matched), the SIP gateway sends a 300 Multiple Choice message, and the multiple routes in the Contact header are listed.

The `redirect contact order` command gives you the flexibility to choose the order in which routes appear in the Contact header.

### How to Perform Basic SIP Configuration

**Note**

For help with a procedure, see the verification and troubleshooting sections listed above.
Configuring SIP VoIP Services on a Cisco Gateway

Shut Down or Enable VoIP Service on Cisco Gateways

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. [no] shutdown [forced]
5. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3 voice service voip</td>
<td>Enters voice-service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Step 4 [no] shutdown [forced]</td>
<td>Shuts down or enables VoIP call services.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-voi-serv)# shutdown forced</td>
<td></td>
</tr>
<tr>
<td>Step 5 exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-voi-serv)# exit</td>
<td></td>
</tr>
</tbody>
</table>

Shut Down or Enable VoIP Submodes on Cisco Gateways

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. [no] call service stop [forced] [maintain-registration]
6. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice-service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters SIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> [no] call service stop [forced] [maintain-registration]</td>
<td>Shuts down or enables VoIP call services for the selected submode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-serv-sip)# call service stop maintain-registration</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-serv-sip)# exit</td>
<td></td>
</tr>
</tbody>
</table>

Configuring SIP Register Support

SUMMARY STEPS

1. enable
2. configure terminal
3. sip-ua
4. registrar {dns: address | ipv4: destination-address} expires seconds [tcp] [secondary]
5. retry register number
6. `timers register milliseconds`
7. `exit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | `enable`  
Example: `Router> enable` | Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted. |
| Step 2 | `configure terminal`  
Example: `Router# configure terminal` | Enters global configuration mode. |
| Step 3 | `sip-ua`  
Example: `Router(config)# sip-ua` | Enters SIP user-agent configuration mode. |
| Step 4 | `registrar [dns: address | ipv4: destination-address] expires seconds [tcp] [secondary]`  
Example: `Router(config-sip-ua)# registrar ipv4:10.8.17.40 expires 3600 secondary` | Registers E.164 numbers on behalf of analog telephone voice ports (FXS) and IP phone virtual voice ports (EFXS) with an external SIP proxy or SIP registrar server. Keywords and arguments are as follows:  
- `dns: address` --Domain-name server that resolves the name of the dial peer to receive calls.  
- `ipv4: destination-address` --IP address of the dial peer to receive calls.  
- `expires seconds` --Default registration time, in seconds.  
- `tcp` --Sets transport layer protocol to TCP. UDP is the default.  
- `secondary` --Specifies registration with a secondary SIP proxy or registrar for redundancy purposes. Optional. |
| Step 5 | `retry register number`  
Example: `Router(config-sip-ua)# retry register 6` | Use this command to set the total number of SIP Register messages that the gateway should send. The argument is as follows:  
| Step 6 | `timers register milliseconds`  
Example: | Use this command to set how long the SIP user agent waits before sending register requests. The argument is as follows: |
### Configuring SIP Redirect Processing Enhancement

#### Configure Call-Redirect Processing Enhancement

Redirect processing using the `redirection` command is enabled by default. To disable and then reset redirect processing, perform the steps listed in this section:

IP-to-IP call redirection can be enabled globally or on a dial-peer basis. To configure, perform the steps listed in these sections:

#### Configuring Call-Redirect Processing Enhancement

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `sip-ua`
4. `no redirection`
5. `redirection`
6. `exit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted.</td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><code>sip-ua</code></td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td><code>no redirection</code></td>
<td></td>
</tr>
<tr>
<td><code>redirection</code></td>
<td></td>
</tr>
<tr>
<td><code>exit</code></td>
<td></td>
</tr>
</tbody>
</table>
Configuring Call Redirect to Support Calls Globally

To configure call redirect to support calls globally, perform the following steps.

### Note
To enable global IP-to-IP call redirection for all VoIP dial peers, use voice-service configuration mode. The default SIP application supports IP-to-IP redirection.

### SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. redirect ip2ip
5. exit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

#### Command or Action

| Step 3 | voice service voip  
| Example: | Enters voice-service VoIP configuration mode. |

| Step 4 | redirect ip2ip  
| Example: | Redirect SIP phone calls to SIP phone calls globally on a gateway using the Cisco IOS voice gateway. |

| Step 5 | exit  
| Example: | Exits the current mode. |

#### Configuring Call Redirect to Support Calls on a Specific VoIP Dial Peer

**Note**

To specify IP-to-IP call redirection for a specific VoIP dial peer, configure it on an inbound dial peer in dial-peer configuration mode. The default application on SIP SRST supports IP-to-IP redirection.

- When IP-to-IP redirection is configured in dial-peer configuration mode, the configuration on the specific inbound dial peer takes precedence over the global configuration entered under voice service configuration.

#### SUMMARY STEPS

1. enable  
2. configure terminal  
3. dial-peer voice tag voip  
4. application application-name  
5. redirect ip2ip  
6. exit

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**  
| enable  
| Example: | Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted. |
| **Step 2**  
| configure terminal  
<p>| Example: | Enters global configuration mode. |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 3** | **Use this command to enter dial-peer configuration mode.**  The argument is as follows:  
   - `tag` --Digits that define a particular dial peer. Range: 1 to 2,147,483,647 (enter without commas). |
| dial-peer voice  tag  voip | Example:  
   Router(config)# dial-peer voice 29 voip |
| **Step 4** | Enables a specific application on a dial peer. The argument is as follows:  
   - `application-name` --Name of the predefined application you wish to enable on the dial peer. For SIP, the default Tel application (from the Cisco IOS image) is session and can be applied to both VoIP and POTS dial peers. The application must support IP-to-IP redirection |
| application  application-name | Example:  
   Router(config-dial-peer)# application session |
| **Step 5** | Redirects SIP phone calls to SIP phone calls on a specific VoIP dial peer using the Cisco IOS voice gateway. |
| redirect ip2ip | Example:  
   Router(conf-dial-peer)# redirect ip2ip |
| **Step 6** | Exits the current mode. |
| exit | Example:  
   Router(conf-dial-peer)# exit |

### Configuring SIP 300 Multiple Choice Messages

#### Configuring Sending of SIP 300 Multiple Choice Messages

| Note | If multiple routes to a destination exist for a redirected number (multiple dial peers are matched), the SIP gateway sends a 300 Multiple Choice message and the multiple routes in the Contact header are listed. This configuration allows users to choose the order in which the routes appear in the Contact header. |

### SUMMARY STEPS

1. enable  
2. configure terminal  
3. voice service voip  
4. sip  
5. redirect contact order [best-match | longest-match]  
6. exit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Router&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Router# configure terminal</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>voice service voip</td>
<td>Enters voice-service VoIP configuration mode.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 4</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip</td>
<td>Enters SIP configuration mode.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-voi-serv)# sip</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 5</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>redirect contact order [best-match</td>
<td>longest-match]</td>
<td>Sets the order of contacts in the 300 Multiple Choice Message. Keywords are as follows:</td>
</tr>
<tr>
<td>Example:</td>
<td>Router(conf-serv-sip)# redirect contact order best-match</td>
<td></td>
</tr>
<tr>
<td>• best-match</td>
<td>--Use the current system configuration to set the order of contacts.</td>
<td></td>
</tr>
<tr>
<td>• longest-match</td>
<td>--Set the contact order by using the destination pattern longest match first, and then the second longest match, the third longest match, and so on. This is the default.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 6</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td>Exits the current mode.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Router(conf-serv-sip)# exit</td>
<td></td>
</tr>
</tbody>
</table>

## Configuring SIP Implementation Enhancements

Minor underlying or minimally configurable features are described in the following sections:

For additional information on SIP implementation enhancements, see “Achieving SIP RFC Compliance.”

### Interaction with Forking Proxies

Call forking enables the terminating gateway to handle multiple requests and the originating gateway to handle multiple provisional responses for the same call. Call forking is required for the deployment of the find me/follow me type of services.
Support for call forking enables the terminating gateway to handle multiple requests and the originating gateway to handle multiple provisional responses for the same call. Interaction with forking proxies applies to gateways acting as a UAC, and takes place when a user is registered to several different locations. When the UAC sends an INVITE message to a proxy, the proxy forks the request and sends it to multiple user agents. The SIP gateway processes multiple 18X responses by treating them as independent transactions under the same call ID. When the relevant dial peers are configured for QoS, the gateway maintains state and initiates RSVP reservations for each of these independent transactions. When it receives an acknowledgment, such as a 200 OK, the gateway accepts the successful acknowledgment and destroys state for all other transactions.

The forking feature sets up RSVP for each transaction only if the dial peers are configured for QoS. If not, the calls proceed as best-effort.

Support for interaction with forking proxies applies only to gateways acting as UACs. It does not apply when the gateway acts as a UAS. In that case, the proxy forks multiple INVITES with the same call ID to the same gateway but with different request URLs.

Also, the forking feature sets up RSVP for each transaction only if the dial peers are configured for QoS. If not, the calls proceed as best-effort.

**SIP Intra-Gateway Hairpinning**

SIP hairpinning is a call routing capability in which an incoming call on a specific gateway is signaled through the IP network and back out the same gateway. This can be a PSTN call routed into the IP network and back out to the PSTN over the same gateway (see the figure below).

*Figure 15: PSTN Hairpinning Example*

![PSTN Hairpinning Example](image)

Similarly, SIP hairpinning can be a call signaled from a line (for example, a telephone line) to the IP network and back out to a line on the same access gateway (see the figure below).

*Figure 16: Telephone Line Hairpinning Example*

![Telephone Line Hairpinning Example](image)

With SIP hairpinning, unique gateways for ingress and egress are unnecessary.

SIP supports plain old telephone service (POTS)-to-POTS hairpinning (which means that the call comes in one voice port and is routed out another voice port). It also supports POTS-to-IP call legs and IP-to-POTS call legs. However, it does not support IP-to-IP hairpinning. This means that the SIP gateway cannot take an inbound SIP call and reroute it back to another SIP device using the VoIP dial peers.

Only minimal configuration is required for this feature. To enable hairpinning on the SIP gateway, see the following configuration example for dial peers. Note that:

- The POTS dial peer must have preference 2 defined, and the VoIP dial peer must have preference 1 defined. This ensures that the call is sent out over IP, not Plain Old Telephone Service (POTS).
- The session target is the same gateway because the call is being redirected to it.
Verifying SIP Gateway Status

To verify SIP gateway status and configuration, perform the following steps as appropriate (commands are listed in alphabetical order).

**SUMMARY STEPS**

1. `show sip service`  
2. `show sip-ua register status`  
3. `show sip-ua statistics`  
4. `show sip-ua status`  
5. `show sip-ua timers`

**DETAILED STEPS**

**Step 1**  
`show sip service`  
Use this command to display the status of SIP call service on a SIP gateway.

The following sample output shows that SIP call service is enabled:

**Example:**

```
Router# show sip service  
SIP Service is up
```

The following sample output shows that SIP call service was shut down with the `shutdown` command:

**Example:**

```
Router# show sip service  
SIP Service is shutdown
```
Verifying SIP Gateway Status

Router# show sip service
SIP service is shut globally
under 'voice service voip'

The following sample output shows that SIP call service was shut down with the call service stop command:

Example:

Router# show sip service
SIP service is shut
under 'voice service voip', 'sip' submode

The following sample output shows that SIP call service was shut down with the shutdown forced command:

Example:

Router# show sip service
SIP service is forced shut globally
under 'voice service voip'

The following sample output shows that SIP call service was shut down with the call service stop forced command:

Example:

Router# show sip service
SIP service is forced shut
under 'voice service voip', 'sip' submode

Step 2  show sip-ua register status

Use this command to display the status of E.164 numbers that a SIP gateway has registered with an external primary SIP registrar.

Example:

Router# show sip-ua register status
Line peer expires(sec) registered
4001 20001 596 no
4002 20002 596 no
5100 1 596 no
9998 2 596 no

Step 3  show sip-ua statistics

Use this command to display response, traffic, and retry SIP statistics, including whether call redirection is disabled.

The following sample shows that four registers were sent:

Example:

Router# show sip-ua statistics
SIP Response Statistics (Inbound/Outbound)
  Informational:  Trying 0/0, Ringing 0/0, Forwarded 0/0, Queued 0/0, SessionProgress 0/0
  Success:  OkInvite 0/0, OkBye 0/0, OkCancel 0/0, OkOptions 0/0, OkPrack 0/0, OkPreconditionMet 0/0, OkSubscribe 0/0, OkNOTIFY 0/0, OkInfo 0/0, 202Accepted 0/0
OkRegister 12/49
Redirection (Inbound only except for MovedTemp(Inbound/Outbound)) :
   MultipleChoice 0, MovedPermanently 0,
   MovedTemporarily 0/0, UseProxy 0,
   AlternateService 0
Client Error:
   BadRequest 0/0, Unauthorized 0/0,
   PaymentRequired 0/0, Forbidden 0/0,
   NotFound 0/0, MethodNotAllowed 0/0,
   NotAcceptable 0/0, ProxyAuthReqd 0/0,
   ReqTimeout 0/0, Conflict 0/0, Gone 0/0,
   ReqEntityTooLarge 0/0, ReqURITooLarge 0/0,
   UnsupportedMediaType 0/0, BadExtension 0/0,
   TempNotAvailable 0/0, CallLegNonExistent 0/0,
   LoopDetected 0/0, TooManyHops 0/0,
   AddrIncomplete 0/0, Ambiguous 0/0,
   BusyHere 0/0, RequestCancel 0/0,
   NotAcceptableMedia 0/0, BadEvent 0/0,
   SETooSmall 0/0
Server Error:
   InternalError 0/0, NotImplemented 0/0,
   BadGateway 0/0, ServiceUnavail 0/0,
   GatewayTimeout 0/0, BadSipVer 0/0,
   PreCondFailure 0/0
Global Failure:
   BusyEverywhere 0/0, Decline 0/0,
   NotExistAnywhere 0/0, NotAcceptable 0/0
Miscellaneous counters:
   RedirectRspMappedToClientErr 0

SIP Total Traffic Statistics (Inbound/Outbound)
   Invite 0/0, Ack 0/0, Bye 0/0,
   Cancel 0/0, Options 0/0,
   Prack 0/0, Comet 0/0,
   Subscribe 0/0, NOTIFY 0/0,
   Refer 0/0, Info 0/0
   Register 49/16

Retry Statistics
   Invite 0, Bye 0, Cancel 0, Response 0,
   Prack 0, Comet 0, Reliable1xx 0, NOTIFY 0

Register 4

SDP application statistics:
   Parses: 0, Builds 0
   Invalid token order: 0, Invalid param: 0
   Not SDP desc: 0, No resource: 0
   Last time SIP Statistics were cleared: <never>

The following sample output shows the RedirectResponseMappedToClientError status message. An incremented number indicates that 3xx responses are to be treated as 4xx responses. When call redirection is enabled (default), the RedirectResponseMappedToClientError status message is not incremented.

Example:

Router# show sip-ua statistics
SIP Response Statistics (Inbound/Outbound)
   Informational:
      Trying 0/0, Ringing 0/0,
      Forwarded 0/0, Queued 0/0,
      SessionProgress 0/0
   Success:
      OKInvite 0/0, OKBye 0/0,
      OKCancel 0/0, OKOptions 0/0,
      OKPrack 0/0, OKPreconditionMet 0/0,
      OKSubscribe 0/0, OKNotify 0/0,
      202Accepted 0/0
Redirection (Inbound only):
  MultipleChoice 0, MovedPermanently 0,
  MovedTemporarily 0, UseProxy 0,
  AlternateService 0
Client Error:
  BadRequest 0/0, Unauthorized 0/0,
  PaymentRequired 0/0, Forbidden 0/0,
  NotFound 0/0, MethodNotAllowed 0/0,
  NotAcceptable 0/0, ProxyAuthReqd 0/0,
  ReqTimeout 0/0, Conflict 0/0, Gone 0/0,
  ReqEntityTooLarge 0/0, ReqURITooLarge 0/0,
  UnsupportedMediaType 0/0, BadExtension 0/0,
  TempNotAvailable 0/0, CallLegNonExistent 0/0,
  LoopDetected 0/0, TooManyHops 0/0,
  AddrIncomplete 0/0, Ambiguous 0/0,
  BusyHere 0/0, RequestCancel 0/0
  NotAcceptableMedia 0/0, BadEvent 0/0
Server Error:
  InternalError 0/0, NotImplemented 0/0,
  BadGateway 0/0, ServiceUnavail 0/0,
  GatewayTimeout 0/0, BadSipVer 0/0,
  PreCondFailure 0/0
Global Failure:
  BusyEverywhere 0/0, Decline 0/0,
  NotExistAnywhere 0/0, NotAcceptable 0/0
Miscellaneous counters:
  RedirectResponseMappedToClientError 1,
SIP Total Traffic Statistics (Inbound/Outbound)
  Invite 0/0, Ack 0/0, Bye 0/0,
  Cancel 0/0, Options 0/0,
  Prack 0/0, Comet 0/0,
  Subscribe 0/0, Notify 0/0,
  Refer 0/0
Retry Statistics
  Invite 0, Bye 0, Cancel 0, Response 0,
  Prack 0, Comet 0, Reliable1xx 0, Notify 0
SDP application statistics:
  Parses: 0, Builds 0
  Invalid token order: 0, Invalid param: 0
  Not SDP desc: 0, No resource: 0

Step 4  show sip-ua status

Use this command to display status for the SIP user agent (UA), including whether call redirection is enabled or disabled.

Example:

Router# show sip-ua status
SIP User Agent Status
SIP User Agent for UDP : ENABLED
SIP User Agent for TCP : ENABLED
SIP User Agent bind status(signaling): DISABLED
SIP User Agent bind status(media): DISABLED
SIP max-forwards : 6
SIP DNS SRV version: 1 (rfc 2052)
Redirection (3xx) message handling: ENABLED

Step 5  show sip-ua timers

Use this command to display the current settings for the SIP user-agent (UA) timers.

The following sample output shows the waiting time before a register request is sent--that is, the value that is set with the timers register command:
Example:

Router# show sip-ua timers
SIP UA Timer Values (milliseconds)
trying 500, expires 180000, connect 500, disconnect 500
comet 500, prack 500, rel1xx 500, notify 500
refer 500, register 500

General Troubleshooting Tips

For more information on troubleshooting, see the following references:

- "Cisco IOS Voice Troubleshooting and Monitoring Guide"
- Cisco IOS Debug Command Reference
- Cisco IOS Voice, Video, and Fax Configuration Guide
- Troubleshooting and Debugging VoIP Call Basics
- VoIP Debug Commands

Note

Commands are listed in alphabetical order.

- Make sure that VoIP is working.

- Make sure that you can make a voice call.

- Verify that SIP-supported codecs are used. Support for codecs varies on different platforms; use the `codec ?` command to determine the codecs available on a specific platform.

- Use the `debug aaa authentication` command to display high-level diagnostics related to AAA logins.

- Use the `debug asnl events` command to verify that the SIP subscription server is up. The output displays a pending message if, for example, the client is unsuccessful in communicating with the server.

- Use the `debug call fallback family of command` to display details of VoIP call fallback.

- Use the `debug cch323` family of commands to provide debugging output for various components within an H.323 subsystem.

- Use the `debug ccsip` family of commands for general SIP debugging, including viewing direction-attribute settings and port and network address-translation traces. Use any of the following related commands:
  - `debug ccsip all`—Enables all SIP-related debugging
  - `debug ccsip calls`—Enables tracing of all SIP service-provider interface (SPI) calls
  - `debug ccsip error`—Enables tracing of SIP SPI errors
  - `debug ccsip events`—Enables tracing of all SIP SPI events
  - `debug ccsip info`—Enables tracing of general SIP SPI information, including verification that call redirection is disabled
• **debug ccsip media**—Enables tracing of SIP media streams
• **debug ccsip messages**—Enables all SIP SPI message tracing, such as those that are exchanged between the SIP user-agent client (UAC) and the access server
• **debug ccsip preauth**—Enables diagnostic reporting of authentication, authorization, and accounting (AAA) preauthentication for SIP calls
• **debug ccsip states**—Enables tracing of all SIP SPI state tracing
• **debug ccsip transport**—Enables tracing of the SIP transport handler and the TCP or User Datagram Protocol (UDP) process

• Use the **debug isdn q931** command to display information about call setup and teardown of ISDN network connections (layer 3) between the local router (user side) and the network.

• Use the **debug kpml** command to enable debug tracing of KeyPad Markup Language (KPML) parser and builder errors.

• Use the **debug radius** command to enable debug tracing of RADIUS attributes.

• Use the **debug rpms-proc preauth** command to enable debug tracing on the RPMS process for H.323 calls, SIP calls, or both H.323 and SIP calls.

• Use the **debug rtr trace** command to trace the execution of an SAA operation.

• Use the **debug voip** family of commands, including the following:
  • **debug voip ccap protoheaders**—Displays messages sent between the originating and terminating gateways. If no headers are being received by the terminating gateway, verify that the **header-passing** command is enabled on the originating gateway.
  • **debug voip ivr script**—Displays any errors that might occur when the Tcl script is run
  • **debug voip rtp session named-event 101**—Displays information important to DTMF-relay debugging, if you are using codec types g726r16 or g726r24. Be sure to append the argument 101 to the command to prevent the console screen from flooding with messages and all calls from failing.

Sample output for some of these commands follows:

**Sample Output for the debug ccsip events Command**

- The example shows how the Proxy-Authorization header is broken down into a decoded username and password.

```
Router# debug ccsip events
CCSIP SPI: SIP Call Events tracing is enabled
21:03:21: sippmh_parse_proxy_auth: Challenge is 'Basic'.
21:03:21: sippmh_parse_proxy_auth: Base64 user-pass string is 'MTIzNDU2Nzg5MDEyMzQ1Njou'.
21:03:21: sip_process_proxy_auth: Decoded user-pass string is '1234567890123456:'.
21:03:21: sip_process_proxy_auth: Username is '1234567890123456'.
21:03:21: sip_process_proxy_auth: Pass is '. '.
```

**Sample Output for the debug ccsip info Command**

This example shows only the portion of the debug output that shows that call redirection is disabled. When call redirection is enabled (default), there are no debug line changes.

```
Router# debug ccsip info
```
00:20:32: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 172.18.207.10 :5060
00:20:32: CCSIP-SPI-CONTROL: act_sentinvite_new_message
00:20:32: CCSIP-SPI-CONTROL: sipSPICheckResponse
00:20:32: sip_stats_status_code
00:20:32: ccsip_get_code_class: !!Call Redirection feature is disabled on the GW
00:20:32: ccsip_map_call_redirect_responses: !!Mapping 302 response to 480
00:20:32: Roundtrip delay 4 milliseconds for method INVITE

Configuration Examples for Basic SIP Configuration

SIP Register Support Example

Current configuration : 3394 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
!
memory-size iomem 15
ip subnet-zero
!
no ip domain lookup
!
voice service voip
redirect ip2ip
sip
redirect contact order best-match
ip dhcp pool vespa
  network 192.168.0.0 255.255.255.0
  option 150 ip 192.168.0.1
  default-router 192.168.0.1
!
voice call carrier capacity active
!
voice class codec 1
  codec preference 2 g711ulaw
!
no voice hpi capture buffer
no voice hpi capture destination
!
fax interface-type fax-mail
mta receive maximum-recipients 0
!
interface Ethernet0/0
  ip address 10.8.17.22 255.255.0.0
  half-duplex
!
interface FastEthernet0/0
  ip address 192.168.0.1 255.255.255.0
  speed auto
  no cdp enable
h323-gateway voip interface
h323-gateway voip id vespa2 ipaddr 10.8.15.4 1718
!
routing rip
network 10.0.0.0
network 192.168.0.0
!
ip default-gateway 10.8.0.1
ip classless
ip route 0.0.0.0 0.0.0.0 10.8.0.1
no ip http server
ip pim bidir-enable
!
tftp-server flash:SEPDEFAULT.cnf
tftp-server flash:P005B302.bin
call fallback active
!
call application global default.new
call rsip-sync
!
voice-port 1/0
!
voice-port 1/1
!
mgcp profile default
!
dial-peer voice 1 pots
  destination-pattern 5100
  port 1/0
!
dial-peer voice 2 pots
  destination-pattern 9998
  port 1/1
!
dial-peer voice 123 voip
  destination-pattern [12]...
  session protocol sipv2
  session target ipv4:10.8.17.42
  dtmf-relay sip-notify
!
gateway
!
sip-ua
  retry invite 3
  retry register 3
  timers register 150
  registrar dns:myhost3.example.com expires 3600
  registrar ipv4:10.8.17.40 expires 3600 secondary
!
telephony-service
  max-dn 10
  max-conferences 4
!
ephone-dn 1
  number 4001
!
ephone-dn 2
  number 4002
!
line con 0
  exec-timeout 0 0
line aux 0
line vty 0 4
login
line vty 5 15
  login
!
no scheduler allocate
end
SIP Redirect Processing Enhancement Examples

This section provides configuration examples to match the identified configuration tasks in the previous sections.

Note
IP addresses and hostnames in examples are fictitious.

Call Redirection Disabled

This example shows that call redirection is disabled on the gateway.

Router# show running-config
Building configuration...
Current configuration : 2791 bytes
!
version 12.2
service config
no service single-slot-reload-enable
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
service udp-small-servers
!
interface FastEthernet2/0
ip address 172.18.200.24 255.255.255.0
duplex auto
no shut
speed 10
ip rsvp bandwidth 7500 7500
!
voice-port 1/1/1
no supervisory disconnect lcfo
!
dial-peer voice 1 pots
application session
destination-pattern 8183821111
port 1/1/1
!
dial-peer voice 3 voip
application session
destination-pattern 7173721111
session protocol sipv2
session target ipv4:172.18.200.36
codec g711ulaw
!
dial-peer voice 4 voip
application session
destination-pattern 6163621111
session protocol sipv2
session target ipv4:172.18.200.33
codec g711ulaw
!
gateway
!
sip-ua
no redirection
retry invite 1
retry bye 1
!
line con 0
line aux 0
line vty 0 4
login
!
end

Call Redirection Enabled

This example shows that call redirection is enabled on the gateway (the default). When call redirection is enabled, the output shows no redirection.

Router# show running-config
Building configuration...
Current configuration : 2791 bytes
!
version 12.2
service config
no service single-slot-reload-enable
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
service udp-small-servers
!
interface FastEthernet2/0
ip address 172.18.200.24 255.255.255.0
duplex auto
no shut
speed 10
ip rsvp bandwidth 7500 7500
!
voice-port 1/1/1
no supervisory disconnect lcfo
!
dial-peer voice 1 pots
application session
destination-pattern 8183821111
port 1/1/1
!
dial-peer voice 3 voip
application session
destination-pattern 7173721111
session protocol sipv2
session target ipv4:172.18.200.36
codec g711ulaw
!
dial-peer voice 4 voip
application session
destination-pattern 6163621111
session protocol sipv2
session target ipv4:172.18.200.33
codec g711ulaw
!
gateway
!
sip-ua
retry invite 1
retry bye 1
Call REDIRECTION USING IP-TO-IP REDIRECTION

This example shows that redirection was set globally on the router.

Current configuration : 3394 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
!
memory-size iomem 15
ip subnet-zero
!
no ip domain lookup
!
voice service voip
redirect ip2ip
sip
redirect contact order best-match
ip dhcp pool vespa
network 192.168.0.0 255.255.255.0
option 150 ip 192.168.0.1
default-router 192.168.0.1
!
voice call carrier capacity active
!
voice class codec 1
codec preference 2 g711ulaw
!
no voice hpi capture buffer
no voice hpi capture destination
!
fax interface-type fax-mail
mta receive maximum-recipients 0
!
interface Ethernet0/0
ip address 10.8.17.22 255.255.0.0
half-duplex
!
interface FastEthernet0/0
ip address 192.168.0.1 255.255.255.0
speed auto
no cdp enable
h323-gateway voip interface
h323-gateway voip id vespa2 ipaddr 10.8.15.4 1718
!
routing rip
network 10.0.0.0
network 192.168.0.0
!
ip default-gateway 10.8.0.1
ip classless
ip route 0.0.0.0 0.0.0.0 10.8.0.1
go ip http server
ip pim bidir-enable
!tftp-server flash:SEPDEFAULT.cnf
tftp-server flash:P005B302.bin
call fallback active
!
call application global default.new
call rsvp-sync
!
voice-port 1/0
!
voice-port 1/1
!
mgcp profile default
!
dial-peer voice 1 pots
   destination-pattern 5100
   port 1/0
!
dial-peer voice 2 pots
   destination-pattern 9998
   port 1/1
!
dial-peer voice 123 voip
   destination-pattern [12]...
   session protocol sipv2
   session target ipv4:10.8.17.42
   dtmf-relay sip-notify
!
gateway
!
sip-ua
   retry invite 3
   retry register 3
   timers register 150
   registrar dns:myhost3.example.com expires 3600
   registrar ipv4:10.8.17.40 expires 3600 secondary
!
telephony-service
   max-dn 10
   max-conferences 4
!
ephone-dn 1
   number 4001
!
ephone-dn 2
   number 4002
!
line con 0
   exec-timeout 0 0
line aux 0
line vty 0 4
login
line vty 5 15
    login
!
no scheduler allocate
end
SIP 300 Multiple Choice Messages Example

This section provides a configuration example showing redirect contact order set to best match.

Current configuration : 3394 bytes
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
!
memory-size iomem 15
ip subnet-zero
!
no ip domain lookup
!
voice service voip
redirect ip2ip
sip
redirect contact order best-match
ip dhcp pool vespa
network 192.168.0.0 255.255.255.0
option 150 ip 192.168.0.1
default-router 192.168.0.1
!
voice call carrier capacity active
!
voice class codec 1
  codec preference 2 g711ulaw
!
no voice hpi capture buffer
no voice hpi capture destination
!
fax interface-type fax-mail
mta receive maximum-recipients 0
!
interface Ethernet0/0
  ip address 10.8.17.22 255.255.0.0
  half-duplex
!
interface FastEthernet0/0
  ip address 192.168.0.1 255.255.255.0
  speed auto
  no cdp enable
  h323-gateway voip interface
  h323-gateway voip id vespa2 ipaddr 10.8.15.4 1718
!
router rip
  network 10.0.0.0
  network 192.168.0.0
!
ip default-gateway 10.8.0.1
ip classless
ip route 0.0.0.0 0.0.0.0 10.8.0.1
no ip http server
ip pim bidir-enable
!
tftp-server flash:SEPDEFAULT.cnf
tftp-server flash:P005B302.bin
call fallback active
!
call application global default.new
call rsvp-sync
!
voice-port 1/0
!
voice-port 1/1
!
mgcp profile default
!
dial-peer voice 1 pots
  destination-pattern 5100
  port 1/0
!
dial-peer voice 2 pots
  destination-pattern 9998
  port 1/1
!
dial-peer voice 123 voip
  destination-pattern [12]...
  session protocol sipv2
  session target ipv4:10.8.17.42
  dtmf-relay sip-notify
!
gateway
!
sip-ua
  retry invite 3
  retry register 3
  timers register 150
  registrar dns:myhost3.example.com expires 3600
  registrar ipv4:10.8.17.40 expires 3600 secondary
!
telephony-service
  max-dn 10
  max-conferences 4
!
ephone-dn 1
  number 4001
!
ephone-dn 2
  number 4002
!
line con 0
  exec-timeout 0 0
line aux 0
line vty 0 4
login
line vty 5 15
login
!
no scheduler allocate
end

Toll Fraud Prevention

When a Cisco router platform is installed with a voice-capable Cisco IOS software image, appropriate features must be enabled on the platform to prevent potential toll fraud exploitation by unauthorized users. Deploy these features on all Cisco router Unified Communications applications that process voice calls, such as Cisco Unified Communications Manager Express (Cisco Unified CME), Cisco Survivable Remote Site Telephony (SRST), Cisco Unified Border Element (Cisco UBE), Cisco IOS-based router and standalone analog and
digital PBX and public-switched telephone network (PSTN) gateways, and Cisco contact-center VoiceXML gateways. These features include, but are not limited to, the following:

- Disable secondary dial tone on voice ports—By default, secondary dial tone is presented on voice ports on Cisco router gateways. Use private line automatic ringdown (PLAR) for foreign exchange office (FXO) ports and direct-inward-dial (DID) for T1/E1 ports to prevent secondary dial tone from being presented to inbound callers.

- Cisco router access control lists (ACLs)—Define ACLs to allow only explicitly valid sources of calls to the router or gateway, and therefore to prevent unauthorized SIP or H.323 calls from unknown parties to be processed and connected by the router or gateway.

- Close unused SIP and H.323 ports—If either the SIP or H.323 protocol is not used in your deployment, close the associated protocol ports. If a Cisco voice gateway has dial peers configured to route calls outbound to the PSTN using either time division multiplexing (TDM) trunks or IP, close the unused H.323 or SIP ports so that calls from unauthorized endpoints cannot connect calls. If the protocols are used and the ports must remain open, use ACLs to limit access to legitimate sources.

- Change SIP port 5060—If SIP is actively used, consider changing the port to something other than well-known port 5060.

- SIP registration—If SIP registration is available on SIP trunks, turn on this feature because it provides an extra level of authentication and validation that only legitimate sources can connect calls. If it is not available, ensure that the appropriate ACLs are in place.

- SIP Digest Authentication—If the SIP Digest Authentication feature is available for either registrations or invites, turn this feature on because it provides an extra level of authentication and validation that only legitimate sources can connect calls.

- Explicit incoming and outgoing dial peers—Use explicit dial peers to control the types and parameters of calls allowed by the router, especially in IP-to-IP connections on Cisco Unified CME, SRST, and Cisco UBE. Incoming dial peers offer additional control on the sources of calls, and outgoing dial peers on the destinations. Incoming dial peers are always used for calls. If a dial peer is not explicitly defined, the implicit dial peer 0 is used to allow all calls.

- Explicit destination patterns—Use dial peers with more granularity than .T for destination patterns to block disallowed off-net call destinations. Use class of restriction (COR) on dial peers with specific destination patterns to allow even more granular control of calls to different destinations on the PSTN.

- Translation rules—Use translation rules to manipulate dialed digits before calls connect to the PSTN to provide better control over who may dial PSTN destinations. Legitimate users dial an access code and an augmented number for PSTN for certain PSTN (for example, international) locations.

- Tcl and VoiceXML scripts—Attach a Tcl/VoiceXML script to dial peers to do database lookups or additional off-router authorization checks to allow or deny call flows based on origination or destination numbers. Tcl/VoiceXML scripts can also be used to add a prefix to inbound DID calls. If the prefix plus DID matches internal extensions, then the call is completed. Otherwise, a prompt can be played to the caller that an invalid number has been dialed.

- Host name validation—Use the “permit hostname” feature to validate initial SIP Invites that contain a fully qualified domain name (FQDN) host name in the Request Uniform Resource Identifier (Request URI) against a configured list of legitimate source hostnames.

- Dynamic Domain Name Service (DNS)—If you are using DNS as the “session target” on dial peers, the actual IP address destination of call connections can vary from one call to the next. Use voice source
groups and ACLs to restrict the valid address ranges expected in DNS responses (which are used subsequently for call setup destinations).

For more configuration guidance, see the “Cisco IOS Unified Communications Manager Express Toll Fraud Prevention” paper.
CHAPTER 11

SIP Binding

The SIP Binding feature enables you to configure a source IP address for signaling packets and media packets.

- Feature Information for SIP Binding, on page 107
- Information About SIP Binding, on page 108
- Configuring SIP Binding, on page 114
- Verifying SIP Binding, on page 116

Feature Information for SIP Binding

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 20: Feature Information for SIP Binding

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| SIP Gateway Support for the bind Command | Cisco IOS 12.2(2)XB, 12.2(2)XB2, 12.2(8)T, 12.2(11)T, and 12.3(4)T Cisco IOS XE 3.1.0S | The SIP Gateway Support for the bind Command feature allows you to configure the source IP address of signaling packets and media packets.  
In 12.2(2)XB, this feature was introduced.  
In 12.3(4)T, this feature was expanded to provide the flexibility to specify different source interfaces for signaling and media, and allow network administrators a finer granularity of control on the network interfaces used for voice traffic.  
The following commands were introduced or modified: bind, show dial-peer voice, show ip sockets, show sip-ua connections, and show sip-ua status. |
Information About SIP Binding

When you configure SIP on a router, the ports on all its interfaces are open by default. This makes the router vulnerable to malicious attackers who can execute toll fraud across the gateway if the router has a public IP address and a public switched telephone network (PSTN) connection. To eliminate the threat, you should bind an interface to an IP address so that only those ports are open to the outside world. In addition, you should protect any public or untrusted interface by configuring a firewall or an access control list (ACL) to prevent unwanted traffic from traversing the router.

Note

All CUBE Enterprise deployments must have signaling and media bind statements specified at the dial-peer or voice class tenant level. For voice call tenants, you must apply tenants to dial-peers used for CUBE call flows if these dial-peers do not have bind statements specified.

Benefits of SIP Binding

- SIP signaling and media paths can advertise the same source IP address on the gateway for certain applications, even if the paths used different addresses to reach the source. This eliminates confusion for firewall applications that may have taken action on source address packets before the use of binding.

- Firewalls filter messages based on variables such as the message source, the target address, and available ports. Normally a firewall opens only certain addresses or port combination to the outside world and those addresses can change dynamically. Because VoIP technology requires the use of more than one address or port combination, the `bind` command adds flexibility by assigning a gateway to a specific interface (and therefore the associated address) for the signaling or media application.

- You can obtain a predefined and separate interface for both signaling and media traffic. After a `bind` command is in effect, the interface it limits is bound solely to that purpose. Administrators can therefore dictate the use of one network to transport the signaling and another network to transport the media. The benefits of administrator control are:
  - Administrators know the traffic that runs on specific networks, thereby making debugging easier.
  - Administrators know the capacity of the network and the target traffic, thereby making engineering and planning easier.
  - Traffic is controlled, allowing Quality of Service (QoS) to be monitored.
- The **bind media** command relaxes the constraints imposed by the **bind control** and **bind all** commands, which cannot be set during an active call. The **bind media** command works with active calls.

### Source Address

In early releases of Cisco IOS software with SIP functionality, the source address of a packet going out of the gateway was never deterministic. That is, the session protocols and VoIP layers always depended on the IP layer to give the *best local address*. The best local address was then used as the source address (the address showing where the SIP request came from) for signaling and media packets. Using this non-deterministic address occasionally caused confusion for firewall applications, because a firewall could not be configured with an exact address and would take action on several different source address packets.

However, the **bind** command enables you to configure the source IP address of signaling and media packets to a specific interface’s IP address. Thus, the address that goes out on the packet is bound to the IP address of the interface specified with the **bind** command. Packets that are not destined to the bound address are discarded.

When you do not want to specify a bind address or if the interface is down, the IP layer still provides the best local address.

The Support Ability to Configure Source IP Address for Signaling and Media per SIP Trunk feature extends the global bind functionality to support the SIP signaling Transport Layer Socket (TLS) with UDP and TCP. The source address at the dial peer is the source address in all the signaling and media packets between the gateway and the remote SIP entity for calls using the dial-peer. Multiple SIP listen sockets with specific source address handle the incoming SIP traffic from each selected SIP entity. The order of preference for retrieving the SIP signalling and media source address for inbound and outbound calls is as follows:

- Bind configuration at dial peer level
- Bind configuration at global level
- Best local IP address to reach the destination

The table below describes the state of the system when the **bind** command is applied in the global or dial peer level:

**Table 21: State of the System for the bind Address**

<table>
<thead>
<tr>
<th>Bind State</th>
<th>System Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>No global bind</td>
<td>The best local address is used in all outbound SIP messages.</td>
</tr>
<tr>
<td></td>
<td>Only one SIP listen socket with a wildcard source address.</td>
</tr>
<tr>
<td>Global bind</td>
<td>Global bind address used in all outbound SIP messages.</td>
</tr>
<tr>
<td></td>
<td>Only one SIP listen socket with global bind address.</td>
</tr>
<tr>
<td>No global bind</td>
<td>Dial peer bind address is used in outbound SIP messages of this dial peer.</td>
</tr>
<tr>
<td>Dial peer bind</td>
<td>The remaining SIP messages use the best local address.</td>
</tr>
<tr>
<td></td>
<td>One SIP listen socket with a wildcard source address.</td>
</tr>
<tr>
<td></td>
<td>Additional SIP listen socket for each different dial peer bind listening on</td>
</tr>
<tr>
<td></td>
<td>the specific dial peer bind address.</td>
</tr>
</tbody>
</table>
**Bind State** | **System Status**
--- | ---
Global bind | Dial peer bind address is used in outbound SIP messages of this dial peer. The remaining SIP messages use the global bind address.
Dial peer bind | One SIP listen socket with global bind address.
 | Additional SIP listen socket for each different dial peer bind command listening on the specific dial peer bind address.

The `bind` command performs different functions based on the state of the interface (see the table below).

**Table 22: State of the Interface for the bind Command**

<table>
<thead>
<tr>
<th>Interface State</th>
<th>Result Using Bind Command</th>
</tr>
</thead>
</table>
| Shut down       | TCP, TLS, and User Datagram Protocol (UDP) socket listeners are initially closed. (Socket listeners receive datagrams addressed to the socket.)  
With or without active calls | Then the sockets are opened to listen to any IP address.  
If the outgoing gateway has the `bind` command enabled and has an active call, the call becomes a one-way call with media flowing from the outgoing gateway to the terminating gateway.  
The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages. |
| No shut down    | TCP, TLS, and UDP socket listeners are initially closed. (Socket listeners receive datagrams addressed to the socket.)  
No active calls | Then the sockets are opened and bound to the IP address set by the `bind` command.  
The sockets accept packets destined for the bound address only.  
The dial peer bind socket listeners of the interface are reopened and the configuration turns active for all subsequent SIP messages. |
| No shut down    | TCP, TLS, and UDP socket listeners are initially closed.  
Active calls    | Then the sockets are opened to listen to any IP address.  
The dial peer bind socket listeners of the interface are reopened and the configuration turns active for all subsequent SIP messages. |
<table>
<thead>
<tr>
<th>Interface State</th>
<th>Result Using Bind Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bound-interface IP address is removed.</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. Then the sockets are opened to listen to any address, because the IP address has been removed. This happens even when SIP was never bound to an IP address. A message stating that the IP address has been deleted from the SIP bound interface is printed. If the outgoing gateway has the <code>bind</code> command enabled and has an active call, the call becomes a one-way call with media flowing from the outgoing gateway to the terminating gateway. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
<tr>
<td>The physical cable is pulled on the bound port or the interface layer is down.</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. Then the sockets are opened and bound to listen to any address. When the pulled cable is replaced, the result is as documented for no shutdown interfaces. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
<tr>
<td>A bind interface is shut down or its IP address is changed or the physical cable is pulled while SIP calls are active.</td>
<td>The call becomes a one-way call with media flowing in only one direction. It flows from the gateway where the change or shutdown took place, to the gateway where no change occurred. Thus, the gateway with the status change no longer receives media. The call is then disconnected, but the disconnected message is not understood by the gateway with the status change, and the call is still assumed to be active. If the bind interface is shutdown, the dial peer bind socket listeners of the interface are closed. If the IP address of the interface is changed, the socket listeners representing the bind command is opened with the available IP address of the interface and the configuration turns active for all subsequent SIP messages.</td>
</tr>
</tbody>
</table>

**Note**

If there are active calls, the `bind` command does not take effect if it is issued for the first time or if another `bind` command is in effect. A message reminds you that there are active calls and that the change cannot take effect.
The `bind` command applied at the dial peer level can be modified only in the following situations:

- Dial peer bind is disabled in the supported IOS configuration options.
- Dial peer bind is removed when the bound interface is removed.
- Dial peer bind is removed when the dial peer is removed.

**Voice Media Stream Processing**

The SIP Gateway Support Enhancements to the `bind` Command feature extends the capabilities of the `bind` command by supporting a deterministic network interface for the voice media stream. Before the voice media stream addition, the `bind` command supported a deterministic network interface for control (signaling) traffic or all traffic. With the SIP Gateway Support Enhancements to the `bind` Command feature, a finer granularity of control is achieved on the network interfaces used for voice traffic.

If multiple `bind` commands are issued in sequence—that is, if one `bind` command is configured and then another `bind` command is configured—a set interaction happens between the commands. The table below describes the expected command behavior.

### Table 23: Interaction Between Previously Set and New `bind` Commands

<table>
<thead>
<tr>
<th>Interface State</th>
<th><code>bind</code> Command</th>
<th>Result Using <code>bind</code> Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without active calls</td>
<td><code>bind all</code></td>
<td>Generated <code>bind control</code> and <code>bind media</code> commands to override existing <code>bind control</code> and <code>bind media</code> commands.</td>
</tr>
<tr>
<td></td>
<td><code>bind control</code></td>
<td>Overrides existing <code>bind control</code> command.</td>
</tr>
<tr>
<td></td>
<td><code>bind media</code></td>
<td>Overrides existing <code>bind media</code> command.</td>
</tr>
<tr>
<td>With active calls</td>
<td><code>bind all</code> or <code>bind control</code></td>
<td>Blocks the command, and the following messages are displayed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 00:16:39: There are active calls</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 00:16:39: configure_sip_bind_command: The bind command change will not take effect</td>
</tr>
<tr>
<td></td>
<td><code>bind media</code></td>
<td>Succeeds and overrides any existing <code>bind media</code> command.</td>
</tr>
</tbody>
</table>

The `bind all` and `bind control` commands perform different functions based on the state of the interface.

---

**Note**

The `bind all` command only applies to global level, whereas the `bind control` and `bind media` command apply to global and dial peer. The table below applies to `bind media` only if the media interface is the same as the `bind control` interface. If the two interfaces are different, media behavior is independent of the interface state.
### Table 24: bind all and bind control Functions, Based on Interface State

<table>
<thead>
<tr>
<th>Interface State</th>
<th>Result Using bind all or bind control Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. (Socket listeners receive datagrams addressed to the socket.) Then the sockets are opened to listen to any IP address. If the outgoing gateway has the <strong>bind</strong> command enabled and has an active call, the call becomes a one-way call with media flowing from the outgoing gateway to the terminating gateway. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
<tr>
<td>With or without active calls</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. (Socket listeners receive datagrams addressed to the socket.) Then the sockets are opened and bound to the IP address set by the <strong>bind</strong> command. The sockets accept packets destined for the bound address only. The dial peer bind socket listeners of the interface are reopened and the configuration turns active for all subsequent SIP messages.</td>
</tr>
<tr>
<td>Not shut down</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. (Socket listeners receive datagrams addressed to the socket.) Then the sockets are opened to listen to any IP address. The dial peer bind socket listeners of the interface are reopened and the configuration turns active for all subsequent SIP messages.</td>
</tr>
<tr>
<td>Without active calls</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. Then the sockets are opened to listen to any address because the IP address has been removed. A message is printed that states the IP address has been deleted from the bound SIP interface. If the outgoing gateway has the <strong>bind</strong> command enabled and has an active call, the call becomes a one-way call with media flowing from the outgoing gateway to the terminating gateway. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
<tr>
<td>Not shut down</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. Then the sockets are opened to listen to any address because the IP address has been removed. A message is printed that states the IP address has been deleted from the bound SIP interface. If the outgoing gateway has the <strong>bind</strong> command enabled and has an active call, the call becomes a one-way call with media flowing from the outgoing gateway to the terminating gateway. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
<tr>
<td>With active calls</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. Then the sockets are opened and bound to any address. When the pulled cable is replaced, the result is as documented for interfaces that are not shut down. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
<tr>
<td>Bound interface’s IP address is removed.</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. Then the sockets are opened and bound to any address because the IP address has been removed. A message is printed that states the IP address has been deleted from the bound SIP interface. If the outgoing gateway has the <strong>bind</strong> command enabled and has an active call, the call becomes a one-way call with media flowing from the outgoing gateway to the terminating gateway. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
<tr>
<td>The physical cable is pulled on the bound port, or the interface layer goes down.</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. Then the sockets are opened and bound to listen to any address. When the pulled cable is replaced, the result is as documented for interfaces that are not shut down. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.</td>
</tr>
</tbody>
</table>
Configuring SIP Binding

<table>
<thead>
<tr>
<th>Interface State</th>
<th>Result Using bind all or bind control Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>A bind interface is shut down, or its IP address is changed, or the physical cable is pulled while SIP calls are active.</td>
<td>The call becomes a one-way call with media flowing in only one direction. The media flows from the gateway where the change or shutdown took place to the gateway where no change occurred. Thus, the gateway with the status change no longer receives media. The call is then disconnected, but the disconnected message is not understood by the gateway with the status change, and the call is still assumed to be active. If the bind interface is shutdown, the dial peer bind socket listeners of the interface are closed. If the IP address of the interface is changed, the socket listeners representing the bind command is opened with the available IP address of the interface and the configuration turns active for all subsequent SIP messages.</td>
</tr>
</tbody>
</table>

### Configuring SIP Binding

#### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface type number**
4. **ip address ip-addressmask [secondary]**
5. **exit**
6. Use one of the following commands to configure SIP binding:
   - **bind {control | all} source-interface interface-id [ipv6-address ipv6-address]** in SIP configuration mode.
   - **bind media [source-address ipv4 ipv4-address | source-interface interface-id [ipv6-address ipv6-address]]** in SIP configuration mode.
   - **voice-class sip bind control source interface interface-id [ipv6-address ipv6-address]** in dial-peer configuration mode.
   - **voice-class sip bind media {source-address ipv4 ipv4-address | source-interface interface-id [ipv6-address ipv6-address]}** in dial-peer configuration mode.
7. **end**

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

| Step 3 | `interface type number` | Configures an interface type and enters the interface configuration mode.  
  |       |                      | - `type number` — Type of interface to be configured and the port, connector, or interface card number. |
|        | Example:                | |
|        | `Router(config)# interface fastethernet0/0` |  
| Step 4 | `ip address ip-addressmask [secondary]` | Configures a primary or secondary IP address for an interface.  
  |        |                      | **Note** Secondary IP address on an interface with SIP binding is not supported for CUBE. |
|        | Example:                | |
|        | `Router(config-if)# ip address 192.168.200.33 255.255.255.0` |  
| Step 5 | `<exit>`                | Exits the current mode. |
|        | Example:                | |
|        | `Router(config-if)# exit` |  
| Step 6 | Use one of the following commands to configure SIP binding:  
  |        | - `bind {control | all} source-interface interface-id [ipv6-address ipv6-address]` in SIP configuration mode.  
  |        | - `bind media {source-address ipv4 ipv4-address | source-interface interface-id [ipv6-address ipv6-address]}` in SIP configuration mode.  
  |        | - `voice-class sip bind control source interface interface-id [ipv6-address ipv6-address]` in dial-peer configuration mode.  
  |        | - `voice-class sip bind media {source-address ipv4 ipv4-address | source-interface interface-id [ipv6-address ipv6-address]}` in dial-peer configuration mode. |
|        | Example:                | |
|        | SIP binding in SIP configuration mode:  
  |        | `Device(config)# voice service voip`  
  |        | `Device(config-voi-serv)# sip`  
  |        | `Device(config-serv-sip)# bind control source-interface FastEthernet0/0`  
  |        | `Device(config-serv-sip)# exit`  
  |        | `Device(config)# voice service voip`  
  |        | `Device(config-voi-serv)# sip`  
  |        | `Device(config-serv-sip)# bind media source-address ipv4 172.18.192.204`  
<p>|        | <code>Device(config-serv-sip)# exit</code> |<br />
|        | Example:                | |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP binding in dial-peer configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 100 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip bind</td>
<td></td>
</tr>
<tr>
<td>control source-interface fastethernet0/0</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# exit</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 100 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip bind</td>
<td></td>
</tr>
<tr>
<td>media source-address ipv4 172.18.192.204</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>

Step 7  end  Exits to privileged EXEC mode.

Verifying SIP Binding

**SUMMARY STEPS**

1. show ip sockets  
2. show sip-ua status  
3. show sip-ua connections {tcp | tls | udp} {brief | detail}  
4. show dial-peer voice  
5. show running-config  

**DETAILED STEPS**

**Step 1**  show ip sockets  
Use this command to display IP socket information and indicate whether the bind address of the receiving gateway is set.

The following sample output indicates that the bind address of the receiving gateway is set:

**Example:**

Device# show ip sockets

Proto Remote Port Local Port In Out Stat TTY OutputIF  
17 0.0.0.0 0--any-- 2517 0 0 9 0  
17 --listen-- 172.18.192.204 1698 0 0 1 0  
17 0.0.0.0 0 172.18.192.204 67 0 0 489 0  
17 0.0.0.0 0 172.18.192.204 5060 0 0 A1 0

**Example:**

**Step 2**  show sip-ua status  
Use this command to display SIP user-agent status and indicate whether bind is enabled.
The following sample output indicates that signaling is disabled and media on 172.18.192.204 is enabled:

**Example:**

```
Device# show sip-ua status
SIP User Agent Status
SIP User Agent for UDP : ENABLED
SIP User Agent for TCP : ENABLED
SIP User Agent for TLS over TCP : ENABLED
SIP User Agent bind status(signaling): DISABLED
SIP User Agent bind status(media): ENABLED 172.18.192.204
SIP early-media for 180 responses with SDP: ENABLED
SIP max-forwards : 70
SIP DNS SRV version: 2 (rfc 2782)
Role in SDP: NONE
Check media source packets: DISABLED
Maximum duration for a telephone-event in NOTIFYs: 2000 ms
SIP support for ISDN SUSPEND/RESUME: ENABLED
Redirection (3xx) message handling: ENABLED
Reason Header will override Response/Request Codes: DISABLED
Out-of-dialog Refer: DISABLED
Presence support is DISABLED
Protocol mode is ipv4
SDP application configuration:
  Version line (v=) required
  Owner line (o=) required
  Timespec line (t=) required
  Media supported: audio video image
  Network types supported: IN
  Address types supported: IPv4 IPv6
  Transport types supported: RTP/AVP udptl
```

**Step 3**  
**show sip-ua connections {tcp [tls] | udp} {brief | detail}**

Use this command to display the connection details for the UDP transport protocol. The command output looks identical for TCP and TLS.

**Example:**

```
Device# show sip-ua connections udp detail
Total active connections : 0
No. of send failures : 0
No. of remote closures : 0
No. of conn. failures : 0
No. of inactive conn. ageouts : 10
--------Printing Detailed Connection Report--------
Note:
  ** Tuples with no matching socket entry
    - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port>'
      to overcome this error condition
  ++ Tuples with mismatched address/port entry
    - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port> id <connid>'
      to overcome this error condition
No Active Connections Found
---------- SIP Transport Layer Listen Sockets ----------
Conn-Id     Local-Address
------------ ---------------------------
2           [9.42.28.29]:5060
```

**Step 4**  
**show dial-peer voice**
Use this command, for each dial peer configured, to verify that the dial-peer configuration is correct. The following is sample output from this command for a VoIP dial peer:

**Example:**

```
Device# show dial-peer voice 101

VoiceOverIpPeer1234
  peer type = voice, system default peer = FALSE, information type = voice,
  description = ' ',
  tag = 1234, destination-pattern = ' ',
  voice reg type = 0, corresponding tag = 0,
  allow watch = FALSE
  answer-address = ' ', preference=0,
  CLID Restriction = None
  CLID Network Number = ' '
  CLID Second Number sent
  CLID Override RDNIS = disabled,
  rtp-ssrc mux = system
  source carrier-id = '', target carrier-id = '',
  source trunk-group-label = '', target trunk-group-label = '',
  numbering Type = 'unknown'
  group = 1234, Admin state is up, Operation state is down,
  incoming called-number = '', connections/maximum = 0/unlimited,
  DTMF Relay = disabled,
  modem transport = system,
  URI classes:
    Incoming (Request) =
    Incoming (Via) =
    Incoming (To) =
    Incoming (From) =
    Destination =
  huntstop = disabled,
  in bound application associated: 'DEFAULT'
  out bound application associated: ''
  dnis-map =
  permission :both
  incoming COR list:maximum capability
  outgoing COR list:minimum requirement
  outgoing LPCOR:
  Translation profile (Incoming):
  Translation profile (Outgoing):
  incoming call blocking:
    translation-profile = '
  disconnect-cause = 'no-service'
  advertise 0x40 capacity_update_timer 25 addrFamily 4 oldAddrFamily 4
  mailbox selection policy: none
  type = voip, session-target = '',
  technology prefix:
  settle-call = disabled
  ip media DSCP = ef, ip media rsvp-pass DSCP = ef
  ip video rsvp-fail DSCP = af31, ip video rsvp-none DSCP = af41
  ip video rsvp-pass DSCP = af41
  ip defending Priority = 0, ip preemption priority = 0
  ip policy locator voice:
  ip policy locator video:
  UDP checksum = disabled,
  session-protocol = sipv2, session-transport = system,
  req-qos video = best-effort, acc-qos video = best-effort,
  req-qos video bandwidth = 64, req-qos audio max bandwidth = 0,
  req-qos video def bandwidth = 384, req-qos video max bandwidth = 0,
```

Cisco Unified Border Element Configuration Guide
RTP dynamic payload type values: NTE = 101
Cisco: NSE=100, fax=96, fax-ack=97, dtmf=121, fax-relay=122
CAS=123, TTY=119, ClearChan=125, PCM switch over u-law=0,
A-law=8, GSMAMR-NB=117 ilBC=116, AAC-id=114, iSAC=124
lmr_tone=0, nte_tone=0
G726r16 using static payload
G726r24 using static payload
RTP comfort noise payload type = 19
fax rate = voice, payload size = 20 bytes
fax protocol = system
fax-relay ecm enable
Fax Relay ans enabled
Fax Relay SG3-to-G3 Enabled (by system configuration)
fax NSF = 0xAD0051 (default)
codec = g729r8, payload size = 20 bytes,
video codec = None
voice class codec = `'
voice class sip session refresh system
voice class sip rsvp-fail-policy voice post-alert mandatory keep-alive interval 30
voice class sip rsvp-fail-policy voice post-alert optional keep-alive interval 30
voice class sip rsvp-fail-policy video post-alert mandatory keep-alive interval 30
voice class sip rsvp-fail-policy video post-alert optional keep-alive interval 30
text relay = disabled
Media Setting = forking (disabled) flow-through (global)
Expect factor = 10, Icpif = 20,
Playout Mode is set to adaptive,
Initial 60 ms, Max 1000 ms
Playout-delay Minimum mode is set to default, value 40 ms
Fax nominal 300 ms
Max Redirects = 1, signaling-type = cas,
VAD = enabled, Poor QOV Trap = disabled,
Source Interface = NONE
voice class sip url = system,
voice class sip tel-config url = system,
voice class sip relxx = system,
voice class sip anat = system,
voice class sip outbound-proxy = "system",
voice class sip associate registered-number =
  system,
voice class sip asserted-id system,
voice class sip privacy system
voice class sip e911 = system,
voice class sip history-info = system,
voice class sip reset timer expires 183 = system,
voice class sip pass-thru headers = system,
voice class sip pass-thru content unsupp = system,
voice class sip pass-thru content sdp = system,
voice class sip copy-list = system,
voice class sip g729 annexb-all = system,
voice class sip early-offer forced = system,
voice class sip negotiate cisco = system,
voice class sip block 180 = system,
voice class sip block 183 = system,
voice class sip block 181 = system,
voice class sip preloaded-route = system,
voice class sip random-contact = system,
voice class sip random-request-uri validate = system,
voice class sip call-route p-called-party-id = system,
voice class sip call-route history-info = system,
voice class sip privacy-policy send-always = system,
voice class sip privacy-policy passethru = system,
voice class sip privacy-policy strip history-info = system,
voice class sip privacy-policy strip diversion = system,
voice class sip map resp-code 181 = system,
voice class sip bind control = enabled, 9.42.28.29,
voice class sip bind media = enabled, 9.42.28.29,
voice class sip bandwidth audio = system,
voice class sip bandwidth video = system,
voice class sip encap clear-channel = system,
voice class sip error-code-override options-keepalive failure = system,
voice class sip calltype-video = false
voice class sip registration passthrough = System
voice class sip authenticate redirecting-number = system,
redirect ip2ip = disabled
local peer = false
probe disabled,
Secure RTP: system (use the global setting)
voice class perms tag = `'
Time elapsed since last clearing of voice call statistics never
Connect Time = 0, Charged Units = 0,
Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
Accepted Calls = 0, Refused Calls = 0,
Last Disconnect Cause is "",
Last Disconnect Text is "",
Last Setup Time = 0.
Last Disconnect Time = 0.

Note If the bind address is not configured at the dial-peer, the output of the show dial-peer voice command remains the same except for the values of the voice class sip bind control and voice class sip bind media, which display “system,” indicating that the bind is configured at the global level.

Step 5 show running-config

Although the bind all command is an accepted configuration, it does not appear in show running-config command output. Because the bind all command is equivalent to issuing the commands bind control and bind media, those are the commands that appear in the show running-config command output.

Example:
The following sample output shows that bind is enabled on router 172.18.192.204:

Building configuration...
Current configuration : 2791 bytes
!
version 12.2
service config
no service single-slot-reload-enable
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
service udp-small-servers
!
ip subnet-zero
ip ftp source-interface Ethernet0
!
voice service voip
 sip
 bind control source-interface FastEthernet0
!
interface FastEthernet0
 ip address 172.18.192.204 255.255.255.0
duplex
speed auto
fair-queue 64 256 1000
ip rsip bandwidth 75000 100
voice-port 1/1/1
no supervisory disconnect lcfo
!
dial-peer voice 1 pots
application session
destination-pattern 5550111
port 1/1/1
!
dial-peer voice 29 voip
application session
destination-pattern 5550133
session protocol sipv2
session target ipv4:172.18.200.33
codec g711ulaw
!
gateway
!
line con 0
line aux 0
line vty 0 4
login
!
end
Verifying SIP Binding
Media Path

The Media Path feature allows you to configure the path taken by media after a call is established. You can configure media path in the following modes:

- Media flow-through
- Media flow-around
- Media anti-trombone
- Feature Information for Media Path, on page 123
- Media Flow-Through, on page 124
- Media Flow-Around, on page 125
- Media Anti-Trombone, on page 126

Feature Information for Media Path

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 25: Feature Information for Configuring Path of Media

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring Media Path</td>
<td>12.4(3), 12.4(24)T, 15.0(1)M</td>
<td>The Media Path feature allows you to configure the path taken by media after a call is established. The following commands were introduced by this feature: media-flow around, media flow-through, media anti-trombone.</td>
</tr>
</tbody>
</table>
Media Flow-Through

Media Flow-Through is a media path mode where media and signaling packets terminate and originate on CUBE. As CUBE is an active participant of the call, this mode is recommended when connected outside an enterprise (untrusted endpoints).

Figure 17: Media Flow-Through Mode

Restrictions for Media Flow-Through

- Video codecs are not supported for Media Flow-Through.
- Media flow-around for Delayed-Offer to Early-Offer audio and video calls is not supported.

Configuring Media Flow-Through

SUMMARY STEPS

1. enable
2. configure terminal
3. Use one of the following commands to configure media flow-through:
   - media flow-through in dial-peer configuration mode
   - media flow-through in global VoIP configuration mode
4. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
</tbody>
</table>
### Media Flow-Around

Media Flow-Around is a media path mode where signaling packets terminate and originate on CUBE. As media bypasses CUBE and flows directly between endpoints, this mode is recommended when connected within an enterprise (trusted endpoints). Media Flow-Around is supported for both audio and video calls.

#### Configuring Media Flow-Around

**SUMMARY STEPS**

1. **enable**
2. **configure terminal**
3. Use one of the following commands to configure media flow-around:
   - **media flow-around** in dial-peer configuration mode

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong></td>
<td>Use one of the following commands to configure media flow-through:</td>
</tr>
<tr>
<td>• <strong>media flow-through</strong> in dial-peer configuration mode</td>
<td>Enables media packets to pass through the endpoints, without the intervention of the CUBE.</td>
</tr>
<tr>
<td>• <strong>media flow-through</strong> in global VoIP configuration mode</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>! Applying SIP profiles to one dial peer only</td>
<td></td>
</tr>
<tr>
<td>Device (config) <strong>dial-peer voice 10 voip</strong></td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer) <strong>media flow-through</strong></td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer) <strong>end</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In global VoIP SIP mode</td>
<td></td>
</tr>
<tr>
<td>! Applying SIP profiles globally</td>
<td></td>
</tr>
<tr>
<td>Device(config)# <strong>voice service voip</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# <strong>media flow-through</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# <strong>end</strong></td>
<td></td>
</tr>
</tbody>
</table>

| **Step 4** | **end** | Exits to privileged EXEC mode. |
**Media Anti-Trombone**

Media Anti-Tromboning is a media path mode that allows CUBE to detect and avoid loops created by call transfers or call forwards. Loops are restricted to the SIP signaling path and removed from the RTP media path.

The user agent may initiate call forwards and call transfers that are sent towards CUBE as a new SIP INVITE dialog. CUBE considers the original call and the forwarded call as separate unrelated calls. Media

---

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | **enable**  
Example:  
Device> enable | Enables privileged EXEC mode. Enter your password if prompted. |
| Step 2 | **configure terminal**  
Example:  
Device# configure terminal | Enters global configuration mode. |
| Step 3 | Use one of the following commands to configure media flow-around:  
- **media flow-around** in dial-peer configuration mode  
- **media flow-around** in global VoIP configuration mode  
Example:  
In dial-peer configuration mode  

```
Device (config-dial-peer)# dial-peer voice 10 voip  
Device (config-dial-peer)# media flow-around  
Device (config-dial-peer)# end
```

Example:  
In global VoIP SIP mode  

```
Device (config)# voice service voip  
Device (config-voi-serv)# media flow-around  
Device (config-voi-serv)# end
```
| Step 4 | **end** | Exits to privileged EXEC mode. |

---

*media flow-around* in global VoIP configuration mode

4. **end**

---

---

---

---
anti-tromboning allows CUBE to detect the relation between the calls and resolve the media loop by sending SDP packets back to the sender.

The figure below illustrates how CUBE needlessly loops RTP packets towards the User Agent because it fails to detect the loop.

*Figure 19: Tromboning - Needless looping of Media Packets*

The figure below illustrates how CUBE detects and avoids the loop with the anti-tromboning feature.

*Figure 20: Anti-Tromboning - Avoiding Media Loops*

**Prerequisites**

**Cisco Unified Border Element**

- Cisco IOS Release 15.1(3)T or a later release must be installed and running on your Cisco Unified Border Element.

**Cisco Unified Border Element (Enterprise)**

- Cisco IOS XE Release 15.1(3)T or a later release must be installed and running on your Cisco ASR 1000 Series Router.

**Restrictions for Media Anti-Tromboning**

- When Media Anti-Tromboning media path mode is activated, CUBE does not perform supplementary services such as handling REFER-based call transfers or media services such as Secure Real-Time Transport Protocol (SRTP) and SNR.

- Anti-Tromboning does not work if one call leg is media flow-through and the other call leg is Media Flow-Around. Similarly, anti-tromboning does not work if one call leg is Session Description Protocol (SDP) passthrough and another call leg is SDP normal.

- H.323 is not supported.
Configuring Media Anti-Tromboning

Before you begin
Configure mode border-element command under voice service voip, global VoIP configuration mode.

SUMMARY STEPS

1. enable
2. configure terminal
3. Enter one of the following commands to configure media anti-tromboning:
   - media anti-trombone in dial-peer configuration mode
   - media anti-trombone in global VoIP configuration mode
4. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> Enter one of the following commands to configure media anti-tromboning:</td>
<td>Enables media anti-trombone for all calls.</td>
</tr>
<tr>
<td>• media anti-trombone in dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>• media anti-trombone in global VoIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>Example: In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>! Applying SIP profiles to one dial peer only Device (config)# dial-peer voice 10 voip Device (config-dial-peer)# media anti-trombone Device (config-dial-peer)# end</td>
<td></td>
</tr>
<tr>
<td>Example: In global VoIP SIP mode</td>
<td></td>
</tr>
<tr>
<td>! Applying SIP profiles globally Device (config)# voice service voip Device (config-voi-serv)# media anti-trombone Device (config-voi-serv)# end</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Step 4 end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
Configuring Media Anti-Tromboning
SIP Profiles

Session Initiation Protocol (SIP) profiles change SIP incoming or outgoing messages so that interoperability between incompatible devices can be ensured.

You can configure SIP profiles with rules to add, remove, copy, or modify the SIP, Session Description Protocol (SDP), and peer headers that enter or leave CUBE. The rules in a SIP profile configuration can be also tagged with a unique number. Tagging the rules allows you to insert or delete rules at any position of the existing SIP profile configuration without deleting and reconfiguring the entire voice-class sip profile.

You can use the following tool to test your SIP profile on an incoming message: https://cway.cisco.com/tools/SipProfileTest/

- Feature Information for SIP Profiles, on page 131
- Information About SIP Profiles, on page 132
- Restrictions for SIP Profiles, on page 134
- How to Configure SIP Profiles, on page 135

Feature Information for SIP Profiles

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
### Table 26: Feature Information for SIP Profiles

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Profiles (for inbound messages)</td>
<td>Cisco IOS 15.4(2)T Cisco IOS XE 3.12S</td>
<td>This feature extends support to inbound messages. This feature modifies the following commands: The <strong>inbound</strong> keyword was added to the <strong>sip-profiles</strong> and <strong>voice-class sip profiles</strong> commands.</td>
</tr>
<tr>
<td>Support for Rotary calls and Media Forking</td>
<td>Cisco IOS 15.3(1)T</td>
<td>With CSCty41575, this feature was enhanced to support forked and rotary calls.</td>
</tr>
<tr>
<td>Configuring SIP Profile (Add, Delete or Modify)</td>
<td>Cisco IOS 12.4(15)XZ Cisco IOS 12.4(20)T Cisco IOS XE 2.5</td>
<td>This feature allows users to change (add, delete, or modify) the standard SIP messages that are sent or received for better interworking with different SIP entities. This feature introduces the following commands: <strong>voice class sip-profiles</strong>, <strong>response</strong>, <strong>request</strong>.</td>
</tr>
<tr>
<td>Support for Non-Standard SIP Headers</td>
<td>Cisco IOS 15.5(2)T</td>
<td>This feature allows users to add, copy, delete, or modify non-standard (for example, X-Cisco-Recording-Participant) using SIP profiles. The <strong>word</strong> keyword was added to the <strong>sip-profiles</strong> command to allow the user to configure any non-standard SIP header.</td>
</tr>
<tr>
<td>Support for tagging rules in a SIP profile configuration</td>
<td>Cisco IOS 15.5(2)T Cisco IOS XE 3.15S</td>
<td>This feature allows users to tag the rules in a SIP profile configuration. Tagging the rules allows users to insert or delete rules at any position of the existing SIP profile configuration without deleting and reconfiguring the entire voice-class sip profile. The following command is introduced in voice class sip profiles configuration mode to tag and insert rules: <strong>rule</strong>. This feature also allows users to upgrade or downgrade all the existing SIP profile configurations to rule-format and non-rule format. The following commands are introduced in global configuration mode: <strong>voice sip sip-profiles upgrade</strong>, <strong>voice sip sip-profiles downgrade</strong>.</td>
</tr>
<tr>
<td>Support for Copying Unsupported SDP Headers</td>
<td>Cisco IOS 15.6(1)T Cisco IOS XE 3.17S</td>
<td>This feature allows for unsupported SDP headers to be copied into a SIP Profile and traverse through CUBE, for all m-lines. The feature introduces the following command: <strong>pass-thru content custom-sdp</strong>.</td>
</tr>
</tbody>
</table>

### Information About SIP Profiles

Protocol translation and repair is a key Cisco Unified Border Element (CUBE) function. CUBE can be deployed between two devices that support the same VoIP protocol (for example, SIP), but do not interwork because...
of differences in how the protocol is implemented or interpreted. CUBE can customize the SIP messaging on either side to what the devices in that segment of the network expects to see by normalizing the SIP messaging on the network border, or between two non-interoperable devices within the network.

Service providers may have policies for which SIP messaging fields should be present (or what constitutes valid values for the header fields) before a SIP call enters their network. Similarly, enterprises and small businesses may have policies for the information that can enter or exit their networks for policy or security reasons from a service provider SIP trunk.

Figure 22: SIP Profile

In order to customize SIP messaging in both directions, you can place and configure a CUBE with a SIP profile at the boundary of these networks.

In addition to network policy compliance, the CUBE SIP profiles can be used to resolve incompatibilities between SIP devices inside the enterprise network. These are the situations in which incompatibilities can arise:

- A device rejects an unknown header (value or parameter) instead of ignoring it
- A device sends incorrect data in a SIP message
- A device does not implement (or implements incorrectly) protocol procedures
- A device expects an optional header value or parameter, or an optional protocol procedure that can be implemented in multiple ways
- A device sends a value or parameter that must be changed or suppressed before it leaves or enters the network
- Variations in the SIP standards on how to achieve certain functions

The SIP profiles feature on CUBE provides a solution to these incompatibilities and customization issues. SIP profiles can also be used to change a header name from the long form to the compact form. For example, From to f. This can be used as a way to reduce the length of a SIP message. By default, the device never sends the compact form of the SIP messages although it receives either the long or the short form.

Important Characteristics of SIP Profiles

Given below are a few important notes for SIP Profiles:

- Copy Variables u01 to u99 are shared by inbound and outbound SIP Profiles.
- Session Initiation Protocol (SIP) and Session Description Protocol (SDP) headers are supported. SDP can be either a standalone body or part of a Multipurpose Internet Mail Extensions (MIME) message.
• The rules configured for an INVITE message are applied only to the first INVITE of a call. A special REINVITE keyword is used to manipulate subsequent INVITEs of a CALL.
• Manipulation of SIP headers by outbound SIP profiles occurs as the last step before the message leaves the CUBE device; that is, after destination dial-peer matching has taken place. Changes to the SIP messages are not remembered or acted on by the CUBE application. The Content-length field is recalculated after the SIP Profiles rules are applied to the outgoing message.
• If the ANY keyword is used in place of a header, it indicates that a rule must be applied to any message within the specified category.
• SIP header modification can be cryptic. It is easier to remove a header and add it back (with the new value), rather than modifying it.
• To include “?” (question-mark) character as part of match-pattern or replace-pattern, you need to press "Ctrl+v" keys and then type "?". This is needed to treat ‘?’ as an input character itself instead of usual device help prompt.
• For header values used to add, modify or copy a header:
  • If a whitespace occurs, the entire value must be included between double quotes. For example, “User-Agent: CISCO CUBE”
  • If double quotes occurs, a back slash must prefix the double quotes. For example, “User-Agent: \"CISCO\" CUBE”
  • Regular expressions are supported.
• If an incoming SIP message contains certain proprietary attributes, CUBE can copy these unsupported SDP attributes or lines from incoming leg to outgoing leg using a SIP profile rule.
• The copy variable can be used in outbound profile to add or modify the outgoing message.

Inbound SIP Profile:
• If the incoming message contains multiple instances of same header, the header values are stored as a comma separated list, and this needs to be considered while modifying it.
• Modification by an inbound SIP profile takes place before regular SIP call processing happens so that behavior of CUBE would be as if it received the message directly without modification.

If inbound dial peer matching fails as required information could not be extracted from headers (like Request-URI, Via, From or To) due to issues in them, global dial peers are applied. An example is a request with invalid SIP-Req-URI.
• After modification by inbound SIP Profiles, the parameters in SIP message might change, which might change the inbound dial-peer matched when actual dial-peer lookup is done.
• In the register pass-through feature, there is only one dial-peer for register and response. So both register from phone and response from registrar would go through the same inbound sip profile under the dial-peer if any.

Restrictions for SIP Profiles
• Removal or addition of mandatory headers is not supported. You can only modify mandatory headers Mandatory SIP headers include To, From, Via, CSeq, Call-Id, and Max-Forwards. Mandatory SDP headers include v, o, s, t, c, and m.
• Addition or removal of entire Multipurpose Internet Mail Extensions (MIME) or (Session Description Protocol) SDP bodies from SIP messages is not supported.
Syntax checking is not performed on SIP messages after SIP profile rules have been applied. Changes specified in the SIP profile should result in valid SIP protocol exchanges.

The header length (including header name) after modification should not exceed 300 characters. Max header length for add value is approximately 220 characters. Max SDP length is 2048 characters. If any header length exceeds this maximum value after applying SIP profiles, then the profile is not applied.

If a header-name is changed to its compact form, SIP profile rules cannot be applied on that header. Thus a SIP profile rule modifying a header name to its compact form must be the last rule on that header.

We cannot modify the "image" m-line attributes (m=image 16850 udptl t38) using SIP profiles. SIP profiles can be applied only on audio and video m-lines in SDP.

In a high-availability (HA) scenario, SIP profiles copy variable data is not check-pointed to standby.

Existing limitations and restrictions of outbound SIP profiles apply to inbound SIP profiles as well.

You cannot configure more than 99 variables for the SIP profiles copy option.

Once a SIP profile is configured using rule tag, you cannot add rules without tags in the same profile and vice-versa.

How to Configure SIP Profiles

To configure SIP Profiles, you must first configure the SIP Profile globally, and apply it at either to all dial peers (globally) or to a single dial peer (dial-peer level). After a SIP profile is configured, it can be applied as an inbound or outbound profile.

Configuring a SIP Profile to Manipulate SIP Request or Response Headers

SUMMARY STEPS

1. enable
2. configure terminal
3. voice class sip-profiles profile-id
4. Enter one of the following to add, remove, modify SIP headers:
   • request message [sip-header | sdp-header] header-to-add add header-value-to-add
   • request message [sip-header | sdp-header] header-to-remove remove
   • request message [sip-header | sdp-header] header-to-modify modify header-value-to-match header-value-to-replace
5. Enter one of the following to add, remove, or modify SIP response headers:
   • response message [method method-type] [sip-header | sdp-header] header-to-add add header-value-to-add
   • response message [method method-type] [sip-header | sdp-header] header-to-remove remove
   • response message [method method-type] [sip-header | sdp-header] header-to-modify modify header-value-to-match header-value-to-replace
6. end
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 1    | enable            | Enables privileged EXEC mode.  
  - Enter your password if prompted. |
| 2    | configure terminal | Enters global configuration mode. |
| 3    | voice class sip-profiles profile-id  
  **Example:**  
  Device(config)# voice class sip-profiles 10 | Creates a SIP Profiles and enters voice class configuration mode. |
| 4    | Enter one of the following to add, remove, modify SIP headers:  
  - request message {sip-header | sdp-header}  
    header-to-add add header-value-to-add  
  - request message {sip-header | sdp-header}  
    header-to-remove remove  
  - request message {sip-header | sdp-header}  
    header-to-modify modify header-value-to-match header-value-to-replace | According to your choice, this step does one of the following:  
  - Adds a SIP or SDP header to a SIP request.  
  - Removes a SIP or SDP header to a SIP request.  
  - Modifies a SIP or SDP header to a SIP request.  
  - If ANY is used in place of a header, it indicates that a rule must be applied to any message within the specified category.  
  - For header-value-to-add used to add a header, header-value-to-match or header-value-to-replace used to modify a header:  
    - If a whitespace occurs, the entire value must be included between double quotes. For example, “User-Agent: CISCO CUBE”  
    - If double quotes occurs, a back slash must prefix the double quotes. For example, “User-Agent: \"CISCO\" CUBE”  
    - Regular expressions are supported. |
| 5    | Enter one of the following to add, remove, or modify SIP response headers:  
  - response message [method method-type] {sip-header | sdp-header}  
    header-to-add add header-value-to-add  
  - response message [method method-type] {sip-header | sdp-header}  
    header-to-remove remove  
  - response message [method method-type] {sip-header | sdp-header}  
    header-to-modify modify header-value-to-match header-value-to-replace | According to your choice, this step does one of the following:  
  - Adds a SIP or SDP header to a SIP response.  
  - Removes a SIP or SDP header to a SIP response.  
  - Modifies a SIP or SDP header to a SIP response.  
  - All notes from the previous step are applicable here. |
| 6    | end               | Exits to privileged EXEC mode |
## Configuring SIP Profiles for Copying Unsupported SDP Headers

CUBE can pass across SDP attributes by defining SIP profile rules. The following steps are involved:

1. Configure CUBE to pass-through custom SDP on in-leg.
2. Define rule to *Copy* relevant attributes from peer SDP on out-leg.
3. Define rule to *Add* or *Modify* attributes in outbound SDP with copied data.

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. To enable copying of unsupported SDP attribute from incoming leg to outbound leg, you need to enable one of the following commands:
   - In Global VoIP SIP configuration mode
     ```yaml
pass-thru content custom-sdp
     ```
   - In dial-peer configuration mode (The configuration is applied on the incoming dial-peer)
     ```yaml
     voice-class sip pass-thru content custom-sdp
     ```
4. `voice class sip-profiles profile-id`
5. Enter one of the following to copy an unsupported SDP line or attribute from peer leg's SDP and add, modify, or remove in the outgoing SDP:
   - `request/response ANY peer-header mline-index index COPY match-pattern copy-variable`
   - `request/response ANY sd-header mline-index indexheader-name ADD copy-variable`
   - `request/response ANY sd-header mline-index indexheader-name MODIFY copy-variable + replace-pattern`
   - `request/response ANY sd-header mline-index indexheader-name REMOVE`
6. `end`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; enable</td>
<td>– Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>To enable copying of unsupported SDP attribute from incoming leg to outbound leg, you need to enable one of the following commands:</td>
<td>Enables copying of unsupported SDP attributes per m-line to the peer leg so that it can be used in outgoing SIP messages.</td>
</tr>
<tr>
<td>– In Global VoIP SIP configuration mode</td>
<td><strong>Note</strong> Enabling this command does not enable the SDP Passthrough feature.</td>
<td></td>
</tr>
</tbody>
</table>
**Example: Configuring SIP Profile Rules (Attribute Passing)**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pass-thru content custom-sdp</code></td>
<td>In dial-peer configuration mode (The configuration is applied on the incoming dial-peer)</td>
</tr>
<tr>
<td><code>voice-class sip pass-thru content custom-sdp</code></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

In Global VoIP SIP configuration mode:

```
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# pass-thru content custom-sdp
```

**Example:**

In Dial-peer configuration mode:

```
Device(config)# dial-peer voice 2 voip
Device(config-dial-peer)# voice-class sip pass-thru content custom-sdp
```

**Step 4** `voice class sip-profiles profile-id`

**Example:**

```
Device(config)# voice class sip-profiles 10
```

- Voice class sip-profile is configured on the outbound dial-peer or as a global configuration.
- Creates a SIP Profile and enters voice class configuration mode.

**Step 5** Enter one of the following to copy an unsupported SDP line or attribute from peer leg's SDP and add, modify, or remove in the outgoing SDP:

- `request/response ANY peer-header sdp mline-index index COPY match-pattern copy-variable`
- `request/response ANY sdp-header mline-index indexheader-name ADD copy-variable`
- `request/response ANY sdp-header mline-index indexheader-name MODIFY copy-variable + replace-pattern`
- `request/response ANY sdp-header mline-index indexheader-name REMOVE`

- M-line Index values:
  - 0 - A value of zero represents the session level.
  - 1 to 6 - A value in the range of one to six represents the m-line number in SDP.
  
- Copy: Enables copying of SDP line or attribute from peer leg SDP.
- Add: Enables adding the copied SDP line or attribute in the outgoing SDP.
- Modify: Enables modifying SDP line or attribute in the outgoing SDP.
- Remove: Enables removing SDP line or attribute in the outgoing SDP.

**Step 6** `end`

- Exits to privileged EXEC mode.

**Example: Configuring SIP Profile Rules (Attribute Passing)**

```
response ANY peer-header sdp mline-index 4 copy "(a=ixmap:0.*)" u01
response ANY sdp-header mline-index 4 a=ixmap add "\u01"
```
Configuring SIP Profile Rules (Parameter Passing)

Example: Configuring SIP Profile Rules (Parameter Passing)

```plaintext
response ANY peer-header sdp mline-index 2 copy "a=fmtp:126 .*(max-fps=....)" u04
response ANY sdp-header mline-index 2 a=fmtp:126 modify ";" ";\u04;"
```

Example: Configuration to Remove an Attribute

```plaintext
response ANY sdp-header mline-index 4 a=test REMOVE
```

Configuring SIP Profile Using Rule Tag

Configure SIP profile rules using the rule tag, enables you to performing the following tasks:

- Add SIP profile request and response headers with a rule tag.
- Modify the existing SIP profile configurations by inserting a rule at any position of the SIP profile without deleting and reconfiguring the entire SIP profile.
- Remove a rule by specifying only rule tag.

Below are the rule tag behaviors that needs to be considered while using rule tag in SIP profile configurations:

- If a rule is added with the tag of an existing rule, then the existing rule is overwritten with the new rule.
- For inserting a rule at the desired position, the SIP profile configuration should be in rule format. In case the SIP profile is in non-rule format, upgrade the SIP profiles to rule format before inserting a rule.
- If a new rule is inserted, the new rule takes the position specified in `before` tag. The subsequent rules are incremented sequentially.
- Once the rule is removed, the tag belonging to the removed rule remains vacant. The tags associated with the subsequent rules remain unchanged.
- If a rule is added to a vacant tag, the new rule gets associated with the vacant tag and the subsequent rules remain unchanged.

SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `voice class sip-profiles profile-id`
4. Enter one of the following to add, copy, modify, or remove a SIP request or response headers to a SIP profile configuration:
   - `rule tag request method sdp-header | sip-header header-name add | copy | modify | remove string`
   - `rule tag response method sdp-header | sip-header header-name add | copy | modify | remove string`
5. Enter one of the following to insert a rule in between the existing set of rules to add, remove, or modify SIP request or response headers:
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 1    | enable            | Enables privileged EXEC mode.  
|      |                   | • Enter your password if prompted. |
| 2    | configure terminal | Enters global configuration mode. |
| 3    | voice class sip-profiles profile-id | Creates a SIP Profile and enters voice class configuration |
|      | Example:          |         |
|      | Device(config)# voice class sip-profiles 10 |         |
| 4    | Enter one of the following to add, copy, modify, or remove a SIP request or response headers to a SIP profile configuration:  
|      | • rule tag request method sd-header | sip-header header-name add | copy | modify | remove string  
|      | • rule tag response method sd-header | sip-header header-name add | copy | modify | remove string |  
|      | According to your choice, this step tags the SIP request or response header with a unique number. |
| 5    | Enter one of the following to insert a rule in between the existing set of rules to add, remove, or modify SIP request or response headers:  
|      | • rule before tag request method sd-header | sip-header header-name add | copy | modify | remove string  
|      | • rule before tag response method sd-header | sip-header header-name add | copy | modify | remove string |  
|      | According to your choice this steps inserts the rule at the position specified in the before tag. The subsequent rules in the existing SIP profile configuration is incremented sequentially. |
| 6    | Enter the following to delete a rule:  
|      | • no rule tag | According to your choice, this step tags the SIP request or response with a unique number. |
| 7    | end            | Exits voice class sip-profiles configuration mode. |
Configuring a SIP Profile for Non-standard SIP Header

SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `voice class sip-profiles profile-id`
4. Enter one of the following to add, copy, remove, or modify non-standard SIP request headers:
   - `request message {sip-header} non-standard-header-to-add add non-standard-header-value-to-add`
   - `request message {sip-header} non-standard-header-to-copy copy non-standard-header-value-to-match copy-variable`
   - `request message {sip-header} non-standard-header-to-remove remove`
   - `request message {sip-header} non-standard-header-to-modify modify non-standard-header-value-to-match non-standard-header-value-to-replace`
5. Enter one of the following to add, copy, remove, or modify non-standard SIP response headers:
   - `response message [method method-type] {sip-header} non-standard-header-to-remove remove`
6. `end`

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Step 3 voice class sip-profiles profile-id</td>
<td>Creates a SIP Profiles and enters voice class configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice class sip-profiles 10</td>
<td></td>
</tr>
<tr>
<td>Step 4 Enter one of the following to add, copy, remove, or modify non-standard SIP request headers:</td>
<td>According to your choice, this step does one of the following:</td>
</tr>
<tr>
<td>• <code>request message {sip-header} non-standard-header-to-add add non-standard-header-value-to-add</code></td>
<td>• Adds a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td>• <code>request message {sip-header} non-standard-header-to-copy copy non-standard-header-value-to-match copy-variable</code></td>
<td>• Copies contents from a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td></td>
<td>• Removes a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td></td>
<td>• Modifies a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>• request message {sip-header } non-standard-header-to-remove remove</td>
<td>• If ANY is used in place of a header, it indicates that a rule must be applied to any message within the specified category.</td>
</tr>
<tr>
<td>• request message {sip-header } non-standard-header-to-modify modify non-standard-header-value-to-match non-standard-header-value-to-replace</td>
<td>• For non-standard-header-value-to-add used to add a non-standard header, non-standard-header-value-to-match or non-standard-header-value-to-replace used to modify a non-standard header:</td>
</tr>
<tr>
<td></td>
<td>• If a whitespace occurs, the entire value must be included between double quotes. For example, “User-Agent: CISCO CUBE”</td>
</tr>
<tr>
<td></td>
<td>• If double quotes occurs, a back slash must prefix the double quotes. For example, “User-Agent: &quot;CISCO&quot; CUBE”</td>
</tr>
<tr>
<td></td>
<td>• Regular expressions are supported.</td>
</tr>
</tbody>
</table>

### Step 5

Enter one of the following to add, copy, remove, or modify non-standard SIP response headers:

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• response message [method method-type] {sip-header } non-standard-header-to-add add non-standard-header-value-to-add</td>
<td>• Adds a non-standard SIP to a SIP response.</td>
</tr>
<tr>
<td>• response message [method method-type] {sip-header } non-standard-header-to-copy copy non-standard-header-value-to-match copy-variable</td>
<td>• Copies contents from a non-standard SIP header to a SIP response.</td>
</tr>
<tr>
<td>• response message [method method-type] {sip-header } non-standard-header-to-remove remove</td>
<td>• Removes a non-standard header to a SIP response.</td>
</tr>
<tr>
<td>• response message [method method-type] {sip-header } non-standard-header-to-modify modify non-standard-header-value-to-match non-standard-header-value-to-replace</td>
<td>• Modifies a non-standard SIP header to a SIP response.</td>
</tr>
<tr>
<td></td>
<td>• All notes from the previous step are applicable here.</td>
</tr>
</tbody>
</table>

### Step 6

end

Exits to privileged EXEC mode

---

**Upgrading or Downgrading SIP Profile Configurations**

You can upgrade or downgrade all the SIP Profile configurations to rule-format or non-rule format automatically.

---

**Note**

We recommend that you downgrade the SIP profiles to non-rule format configuration before migrating to a version below Cisco IOS Release 15.5(2)T or Cisco IOS-XE Release 3.15S. If you migrate without downgrading the SIP profile configurations, then all the SIP profile configurations is lost after migration.
SUMMARY STEPS

1. enable
2. Enter the following to upgrade SIP profiles configurations to rule-format:
   • voice sip sip-profiles upgrade
3. Enter the following to downgrade SIP profiles configurations to non-rule format:
   • voice sip sip-profiles downgrade
4. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
   • Enter your password if prompted. |
| **Step 2** Enter the following to upgrade SIP profiles configurations to rule-format:  
   • voice sip sip-profiles upgrade | Upgrades all SIP Profiles to rule-format configurations. |
| **Example:**  
Device#voice sip sip-profiles upgrade | |
| **Step 3** Enter the following to downgrade SIP profiles configurations to non-rule format:  
   • voice sip sip-profiles downgrade | Downgrades all SIP Profiles from rule-format configurations to non-rule format configurations. |
| **Example:**  
Device#voice sip sip-profiles downgrade | |
| **Step 4** end | Exits privileged EXEC mode. |

What to do next

Now apply the SIP Profile as an inbound or outbound SIP profile.

Configuring a SIP Profile as an Outbound Profile

SUMMARY STEPS

1. enable
2. configure terminal
3. Apply the SIP profile to a dial peer:
   • voice-class sip profiles profile-id in the dial-peer configuration mode.
Configuring a SIP Profile as an Inbound Profile

You can configure a SIP profile as an inbound profile applied globally or to a single inbound dial peer. Inbound SIP profiles feature must be enabled before applying it to dial peers.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. sip
5. sip-profiles inbound
6. Apply the SIP profile to a dial peer:
   - `voice-class sip profiles profile-id inbound` in the dial-peer configuration mode.
   - `sip-profiles profile-id inbound` in the global VoIP configuration mode

7. end

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>• Enter your password if prompted.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>voice service voip</code></td>
<td>Enters global VoIP configuration mode.</td>
</tr>
<tr>
<td>Example: <code>Device(config)# voice service voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters global VoIP SIP configuration mode.</td>
</tr>
<tr>
<td>Example: <code>Device(config-voi-serv)# sip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> sip-profiles inbound</td>
<td>Enables inbound SIP profiles feature.</td>
</tr>
<tr>
<td>Example: <code>Device(config-voi-sip)# sip-profiles inbound</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> Apply the SIP profile to a dial peer:</td>
<td></td>
</tr>
<tr>
<td>• <code>voice-class sip profiles profile-id inbound</code> in the dial-peer configuration mode.</td>
<td></td>
</tr>
<tr>
<td>• <code>sip-profiles profile-id inbound</code> in the global VoIP configuration mode</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>! Applying SIP profiles to one dial peer only</td>
<td></td>
</tr>
<tr>
<td>Device (config)# <code>dial-peer voice 10 voip</code></td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer)# <code>voice-class sip profiles 30 inbound</code></td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer)# <code>end</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In global VoIP SIP mode</td>
<td></td>
</tr>
<tr>
<td>! Applying SIP profiles globally</td>
<td></td>
</tr>
<tr>
<td>Device(config)# <code>voice service voip</code></td>
<td></td>
</tr>
<tr>
<td>Device (config-voi-serv)# <code>sip</code></td>
<td></td>
</tr>
</tbody>
</table>
### Verifying SIP Profiles

**SUMMARY STEPS**

1. show dial-peer voice *id* | include profile

**DETAILED STEPS**

```
show dial-peer voice *id* | include profile
```

Displays information related to SIP profiles configured on the specified dial peer.

**Example:**

```
Device# show dial-peer voice 10 | include profile
```

| Translation profile (Incoming): |
| Translation profile (Outgoing): |
| translation-profile = `'' |
| voice class sip profiles = 11 |
| voice class sip profiles inbound = 10 |

---

### Troubleshooting SIP Profiles

**SUMMARY STEPS**

1. debug ccsip all

**DETAILED STEPS**

```
ddebug ccsip all
```

This command displays the applied SIP profiles.

**Example:**

Applied SIP profile is highlighted in the example below.

```
Device# debug ccsip all
```

Oct 12 06:51:53.619: //I/735085DC8F3D/SIP/Info/sipSPIGetShrlPeer:
Try match incoming dialpeer for Calling number:
: sippOct 12 06:51:53.619:
Examples: Adding, Modifying, Removing SIP Profiles

Example: Adding a SIP, SDP, or Peer Header

Example: Adding "b=AS:4000" SDP header to the video-media Header of the INVITE SDP Request Messages

```
Device(config)# voice class sip-profiles 10
Device(config-class)# request INVITE sdp-header Video-Bandwidth-Info add "b=AS:4000"
```

Example: Adding "b=AS:4000" SDP header to the video-media Header of the INVITE SDP Request Messages in rule format

```
Device(config)# voice class sip-profiles 10
Device(config-class)# rule 1 request INVITE sdp-header Video-Bandwidth-Info add "b=AS:4000"
```

Example: Adding the Retry-After Header to the SIP 480 Response Messages

```
Device(config)# voice class sip-profiles 20
```
Example: Adding a SIP, SDP, or Peer Header

Device(config-class)# response 480 sip-header Retry-After add "Retry-After: 60"
Device(config-class)# end

Example: Adding the Retry-After Header to the SIP 480 Response Messages in rule format

Device(config)# voice class sip-profiles 20
Device(config-class)# rule 1 response 480 sip-header Retry-After add "Retry-After: 60"
Device(config-class)# end

Example: Adding "User-Agent: SIP-GW-UA" to the User-Agent Field of the 200 Response SIP Messages

Device(config)# voice class sip-profiles 40
Device(config-class)# response 200 sip-header User-Agent add "User-Agent: SIP-GW-UA"
Device(config-class)# end

Example: Adding "User-Agent: SIP-GW-UA" to the User-Agent Field of the 200 Response SIP Messages in rule format

Device(config)# voice class sip-profiles 40
Device(config-class)# rule 1 response 200 sip-header User-Agent add "User-Agent: SIP-GW-UA"
Device(config-class)# end

Example: Adding "a=ixmap:0 ping" in M-Line number 4 of the INVITE SDP Request Messages

Device(config)# voice class sip-profiles 10
Device(config-class)# request INVITE sdp-header mline-index 4 a=ixmap add "a=ixmap:0 ping"
Device(config-class)# end

Applying the SIP Profiles

! applying SIP profiles globally
Device(config)#voice service voip
Device(config-voi-serv)#sip
Device(config-voi-sip)#sip-profiles 20
Device(config-voi-sip)#end

! applying SIP profiles to one dial peer only
Device(config) dial-peer voice 10 voip
Device(config-dial-peer)#voice-class sip profiles 30
Device(config-dial-peer)#end
Example: Modifying a SIP, SDP, or Peer Header

Example: Modifying SIP-Req-URI of the Header of the INVITE and RE-INVITE SIP Request Messages to include "user=phone"

Device(config)# voice class sip-profiles 30
Device(config-class)# request INVITE sip-header SIP-Req-URI modify "; SIP/2.0" ";user=phone SIP/2.0"
Device(config-class)# request RE-INVITE sip-header SIP-Req-URI modify "; SIP/2.0" ";user=phone SIP/2.0"
Device(config-class)# end

Example: Modifying SIP-Req-URI of the Header of the INVITE and RE-INVITE SIP Request Messages to include "user=phone" in rule format

Device(config)# voice class sip-profiles 30
Device(config-class)# rule 1 request INVITE sip-header SIP-Req-URI modify "; SIP/2.0" ";user=phone SIP/2.0"
Device(config-class)# rule 2 request RE-INVITE sip-header SIP-Req-URI modify "; SIP/2.0" ";user=phone SIP/2.0"
Device(config-class)# end

Modify the From Field of a SIP INVITE Request Messages to “gateway@gw-ip-address” Format

For example, modify 2222000020@10.13.24.7 to gateway@10.13.24.7

Device(config)# voice class sip-profiles 20
Device(config-class)# request INVITE sip-header From modify "(<.*:)(.*@)" ";gateway@gw-ip-address"

Modify the From Field of a SIP INVITE Request Messages to “gateway@gw-ip-address” Format in rule format

For example, modify 2222000020@10.13.24.7 to gateway@10.13.24.7

Device(config)# voice class sip-profiles 20
Device(config-class)# rule 1 request INVITE sip-header From modify "(<.*:)(.*@)" ";gateway@gw-ip-address"

Replace "CiscoSystems-SIP-GW-UserAgent" with "-" in the Originator Header of the SDP in INVITE Request Messages

Device(config)# voice class sip-profiles 10
Device(config-class)# request INVITE sdp-header Session-Owner modify "CiscoSystems-SIP-GW-UserAgent" ";-"

Replace "CiscoSystems-SIP-GW-UserAgent" with "-" in the Originator Header of the SDP in INVITE Request Messages in rule format

Device(config)# voice class sip-profiles 10
Device(config-class)# rule 1 request INVITE sdp-header Session-Owner modify "CiscoSystems-SIP-GW-UserAgent" ";-"
Convert "sip uri" to "tel uri" in Req-URI, From and To Headers of SIP INVITE Request Messages

For example, modify sip:2222000020@9.13.24.6:5060” to “tel:2222000020

Device(config)# voice class sip-profiles 40
Device(config-class)# request INVITE sip-header SIP-Req-URI modify "sip:(.*)@[^ \]+" "tel:\1"
Device(config-class)# request INVITE sip-header From modify "<sip:(.*)@.*>" "<tel:\1>
Device(config-class)# request INVITE sip-header To modify "<sip:(.*)@.*>" "<tel:\1>

Convert "sip uri" to "tel uri" in Req-URI, From and To Headers of SIP INVITE Request Messages in rule format

For example, modify sip:2222000020@9.13.24.6:5060” to “tel:2222000020

Device(config)# voice class sip-profiles 40
Device(config-class)# rule 1 request INVITE sip-header SIP-Req-URI modify "sip:(.*)@[^ \]+" "tel:\1"
Device(config-class)# rule 2 request INVITE sip-header From modify "<sip:(.*)@.*>" "<tel:\1>
Device(config-class)# rule 3 request INVITE sip-header To modify "<sip:(.*)@.*>" "<tel:\1>

Example: Change the Audio Attribute Ptime:20 to Ptime:30

Inbound ptime:
\texttt{a=ptime:20}

Outbound ptime:
\texttt{a=ptime:30}

Device(config)# voice class sip-profiles 103
Device(config-class)# request ANY sdp-header Audio-Attribute modify "a=ptime:20" "a=ptime:30"

Example: Modify Audio direction "Audio-Attribute"

Some service providers or customer equipment reply to delay offer invites and or re-invites that contain a=inactive with a=inactive, a=recvonly, or a=sendonly. This can create an issue when trying to transfer or retrieve a call from hold. The result is normally one-way audio after hold or resume or transfer or moh is not heard. To resolve this issue changing the audio attribute to Sendrecv prevents the provider from replaying back with a=inactive, a=recvonly, or a=sendonly.

Case 1:

Inbound Audio-Attribute
\texttt{a-inactive}

Outbound Audio-Attribute
\texttt{a-sendrecv}

Case 2:

Inbound Audio-Attribute
\texttt{a=recvonly}
Outbound Audio-Attribute
a=sendrecv

Case 3
Inbound Audio-Attribute
a=sendonly
Outbound Audio-Attribute
a=sendrecv

Device(config)# voice class sip-profiles 104
Device(config-class)# request any sdp-header Audio-Attribute modify "a=inactive" "a=sendrecv"
Device(config-class)# request any sdp-header Audio-Attribute modify "a=recvonly" "a=sendrecv"
Device(config-class)# request any sdp-header Audio-Attribute modify "a=sendonly" "a=sendrecv"
Device(config-class)# response any sdp-header Audio-Attribute modify "a=inactive" "a=sendrecv"
Device(config-class)# response any sdp-header Audio-Attribute modify "a=recvonly" "a=sendrecv"
Device(config-class)# response any sdp-header Audio-Attribute modify "a=sendonly" "a=sendrecv"

Example: Modifying Packetization Mode in a=fmtp line of M-line number 2 of the INVITE SDP Request Messages

Device(config)# voice class sip-profiles 10
Device(config-class)# request INVITE sdp-header mline-index 2 a=fmtp modify "packetization-mode=1" "packetization-mode=0"
Device(config-class)# end

Applying the SIP Profiles to Dial Peers

! Applying SIP Profiles globally
Device(config)# voice service voip
Device(config-voipl-serv)# sip-profiles 20
Device(config-voipl-serv)# sip-profiles 10
Device(config-voipl-serv)# sip-profiles 40
Device(config-voipl-serv)# sip-profiles 103
Device(config-voipl-serv)# sip-profiles 104
Device(config-voipl-serv)# exit

! Applying SIP Profiles to one dial peer only
Device(config)# dial-peer voice 90 voip
Device(config-dial-peer)# voice-class sip profiles 30

Example: Remove a SIP, SDP, or Peer Header

Remove Cisco-Guid SIP header from all Requests and Responses

Device(config)# voice class sip-profiles 20
Device(config-class)# request ANY sip-header Cisco-Guid remove
Device(config-class)# response ANY sip-header Cisco-Guid remove
Device(config-class)# end

Remove Server Header from 100 and 180 SIP Response Messages

Device(config)# voice class sip-profiles 20
Device(config-class)# response 100 sip-header Server remove
Device(config-class)# response 180 sip-header Server remove
Device(config-class)# end

Removing a SIP Profile rule in rule format configuration

SIP Profile configuration in rule format

Device(config)# voice class sip-profiles 10
Device(config-class)# rule 1 request any sdp-header Audio-Attribute modify "a=inactive"
"a=sendrecv"
Device(config-class)# rule 2 request any sdp-header Audio-Attribute modify "a=recvonly"
"a=sendrecv"
Device(config-class)# end

Removing the rule using rule tag

Device(config)# voice class sip-profiles 10
Device(config-class)# no rule 1
Device(config)# end

Once the rule is removed, the tag belonging to the removed rule remains vacant. The tags associated with the subsequent rules are unchanged.

The SIP Profile configuration after removing the rule

Device(config)# voice class sip-profiles 10
Device(config-class)# rule 2 request any sdp-header Audio-Attribute modify "a=recvonly"
"a=sendrecv"
Device(config-class)# end

Example: Removing “a=ixmap” in M-Line number 4 of the INVITE SDP Request Messages

Device(config)# voice class sip-profiles 10
Device(config-class)# response ANY sdp-header mline-index 4 a=ixmap REMOVE
Device(config-class)# end

Example: Inserting SIP Profile Rules

Example: Inserting a SIP Profile Rule

Inserting a SIP profile rule to a SIP Profile
Example: Upgrading and Downgrading SIP Profiles automatically

Upgrading SIP Profiles to rule-format

The following is a snippet from `show running-config` command showing the SIP profiles in non-rule format:

Device#show running-config
!
request INVITE sip-header Contact Modify "(.*)" "\1;temp=xyz"
request INVITE sip-header Supported Add "Supported: "
!

Execute the following command in EXEC(#) mode to upgrade the SIP Profiles to rule-format:
Device#voice sip sip-profiles upgrade
!

Downgrading SIP Profiles to non-rule format

The following is a snippet from `show running-config` command showing SIP profiles in rule-format:

Device#show running-config
!
rule 1 request INVITE sip-header Contact Modify "(.*)" "\1;temp=xyz"
rule 2 request INVITE sip-header Supported Add "Supported: "
!

Execute the following command in EXEC(##) mode to downgrade SIP Profiles to non-rule format:
Device# voice sip sip-profiles downgrade

The following is a snippet from `show running-config` command showing SIP profiles after downgrading to non-rule format:
Example: Modifying Diversion Headers

Example: Modify Diversion Headers from Three-Digit Extensions to Ten Digits.

Most service providers require a ten digit diversion header. Prior to Call manager 8.6, Call manager would only send the extension in the diversion header. A SIP profile can be used to make the diversion header ten digits.

Call manager version 8.6 and above has the field “Redirecting Party Transformation CSS” which lets you expand the diversion header on the call manager.

The SIP profile will look for a diversion header containing "<sip:5...", where ... stands for the three-digit extension and then concatenates 9789365 with these three digits.

Original Diversion Header:

```
Diversion:<sip:5100@161.44.77.193>;privacy=off;reason=unconditional;counter=1;screen=no
```

Modified Diversion Header:

```
Diversion: <sip:9789365100@10.86.176.19>;privacy=off;reason=unconditional;counter=1;screen=no
```

Example: Create a Diversion header depending on the area code in the From field

Most service providers require a redirected call to have a diversion header that contains a full 10 digit number that is associated with a SIP trunk group. Sometimes, a SIP trunk may cover several different area codes, states, and geographic locations. In this scenario, the service provider may require a specific number to be placed in the diversion header depending on the calling party number.

In the below example, if the From field has an area code of 978 "<sip:978", the SIP profile leaves the From field as is and adds a diversion header.

```
Device(config)# voice class sip-profiles 102
Device(config-class)# request INVITE sip-header From modify "From:(.*)<sip:978(.*)@(.*)" "<sip:9789365000@10.86.176.19:5060;privacy=off;reason=unconditional;counter=1;screen=no"
```

The below diversion header is added. There was no diversion header before this was added:

```
Diversion: <sip:9789365000@10.86.176.19:5060;transport=udp>"
```
Example: Sample SIP Profile Application on SIP Invite Message

The SIP profile configured is below:

```plaintext
voice class sip-profiles 1
  request INVITE sdp-header Audio-Bandwidth-Info add "b=AS:1600"
  request ANY sip-header Cisco-Guid remove
  request INVITE sdp-header Session-Owner modify "CiscoSystems-SIP-GW-UserAgent" "-
```

The SIP INVITE message before the SIP profile has been applied is shown below:

```
INVITE sip:2222000020@9.13.40.250:5060 SIP/2.0
Via: SIP/2.0/UDP 9.13.40.249:5060;branch=z9hG4bK1A203F
From: "sipp " <sip:11110000109@9.13.40.249>;tag=F11AE0-1D8D
To: <sip:2222000020@9.13.40.250>
Date: Mon, 29 Oct 2007 19:02:04 GMT
Call-ID: 4561B116-858811DC-804DEF2E-4CF2D71B@9.13.40.249
Cisco-Guid: 1163870326-2240287196-2152197934-1290983195
Content-Length: 290
```

The SIP INVITE message after the SIP profile has been applied is shown below:

- The Cisco-Guid has been removed.
- CiscoSystemsSIP-GW-UserAgent has been replaced with -.
- The Audio-Bandwidth SDP header has been added with the value b=AS:1600.

```
INVITE sip:2222000020@9.13.40.250:5060 SIP/2.0
Via: SIP/2.0/UDP 9.13.40.249:5060;branch=z9hG4bK1A203F
From: "sipp " <sip:11110000109@9.13.40.249>;tag=F11AE0-1D8D
To: <sip:2222000020@9.13.40.250>
Date: Mon, 29 Oct 2007 19:02:04 GMT
Call-ID: 4561B116-858811DC-804DEF2E-4CF2D71B@9.13.40.249
Content-Length: 279
```

Example: Sample SIP Profile for Non-Standard SIP Headers

Before Cisco IOS Release 15.5(2)T, there was no method to add, copy, delete, or modify any non-standard SIP headers like 'X-Cisco-Recording-Participant' using SIP profiles. The SIP profile will look for the new option "WORD" that allows the user to change any non-standard SIP header.
Example: Sample SIP Profile for Non-Standard SIP Headers

voice class sip-profiles 1
request INVITE sip-header X-Cisco-Recording-Participant copy "sip:(.*)@" u01
request INVITE sip-header X-Cisco-Recording-Participant modify "sip:sipp@" "sip:1000@"
request INVITE sip-header My-Info add "My-Info: MF Call"
request INVITE sip-header My-Info remove
CHAPTER 14

SIP Out-of-Dialog OPTIONS Ping Group

This feature groups the monitoring of SIP dial-peers endpoints and servers by consolidating dial peers with the same SIP Out-of-Dialog (OOD) OPTIONS ping setup.

• Finding Feature Information, on page 157
• Information About SIP Out-Of-dialog OPTIONS Ping Group, on page 157
• How to Configure SIP Out-Of-dialog OPTIONS Ping Group, on page 158
• Configuration Examples For SIP Out-of-Dialog OPTIONS Ping Group , on page 161
• Additional References, on page 162
• Feature Information for SIP Out-of-dialog OPTIONS Ping Group , on page 163

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About SIP Out-Of-dialog OPTIONS Ping Group

SIP Out-of-Dialog OPTIONS Ping Group Overview

The SIP Out-Of-Dialog OPTIONS (OODO) Ping Group feature is an existing mechanism that is used by CUBE to monitor the status of a single SIP dial-peer destination (keepalive). A generic heartbeat mechanism allows you to monitor the status of SIP servers or endpoints and provide the option of marking a dial peer as inactive (busyout) upon total heartbeat failure.

You can now consolidate the sending of SIP OODO ping packets by grouping dial peers with the same SIP OODO ping setup. A keepalive profile is created and referenced by different SIP dial peers. An OODO Options ping heartbeat keepalive connection is set up for each dial-peer destination of a keepalive profile. If a heartbeat failure occurs for any of the dial peers of the profile, the status of the respective dial peer is changed to inactive (busyout) by CUBE.
Configuring the same OPTIONS KEEPALIVE profile on two or more dial-peers with different bind interfaces configured is not supported. This leads to a scenario wherein the OPTIONS SIP message is not sent from all bind interfaces expect the first configured one. But the dial-peer is always marked as ACTIVE.

You can use the `shutdown` command to suspend monitoring of all dial peers associated with a keepalive profile.

The new command `voice-class sip options-keepalive profile tag` is used to monitor a group of SIP servers or endpoints and the existing `voice-class sip options-keepalive` command is used to monitor a single SIP endpoint or server.

You can configure a server group to be a part of a SIP OODO ping group. A SIP dial peer is updated to BUSY state only if all targets of its server group does not response to the OODO ping.

**How to Configure SIP Out-Of-dialog OPTIONS Ping Group**

**Configuring SIP Out-of-Dialog OPTIONS Ping Group**

**Before you begin**
Configure SIP profiles and server groups.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `voice class sip-options-keepalive keepalive-group-profile-id`
4. `description text`
5. `transport {tcp [tls] | udp | system}`
6. `sip-profiles profile-number`
7. `down-interval down-interval`
8. `up-interval up-interval`
9. `retry retry-interval`
10. `exit`
11. `dial-peer voice dial-peer-id voip`
12. `session protocol sipv2`
13. `voice-class sip options-keepalive profile keepalive-group-profile-id`
14. `session server-group server-group-id`
15. `end`
16. `show voice class sip-options-keepalive keepalive-group-profile-id`
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>enable</td>
<td>Enters privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>voice class sip-options-keepalive keepalive-group-profile-id</td>
<td>Configures a keepalive profile and enters voice class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>- You can use the <strong>shutdown</strong> command to suspend keepalive activity for all dial peers associated with the keepalive profile.</td>
</tr>
<tr>
<td></td>
<td>Device(config)# voice class sip-options-keepalive 171</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>description text</td>
<td>Configures a textual description for the keepalive heartbeat connection.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# description Target Boston</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>transport {tcp [tls]</td>
<td>udp</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>- The default value is system.</td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# transport tcp</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>sip-profiles profile-number</td>
<td>Specifies the SIP profile that is to be used to send this message.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>- To configure a SIP profile, refer to “Configuring SIP Parameter Modification”.</td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# sip-profiles 100</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>down-interval down-interval</td>
<td>Configures the time (in seconds) at which an SIP OODO ping is sent to the dial-peer endpoint when the heartbeat connection to the endpoint is in Down status.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>- The default value is 30.</td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# down-interval 35</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>up-interval up-interval</td>
<td>Configures the time (in seconds) at which an SIP OODO ping is sent to the dial-peer endpoint when the heartbeat connection to the endpoint is in Up status.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>- The default value is 60.</td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# up-interval 65</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>retry retry-interval</td>
<td>Configures the maximum number of OODO ping retrials permitted for a dial-peer destination. After receiving failed responses for the configured number of OODO ping, the</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>10</td>
<td>exit</td>
<td>Exits voice class configuration mode and enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# exit</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>dial-peer voice</td>
<td>Defines a local dial peer and enters dial peer configuration mode.</td>
</tr>
<tr>
<td></td>
<td>dial-peer-id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 123 voip</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>session protocol</td>
<td>Specifies SIP version 2 as the session protocol for calls between local and remote routers using the packet network.</td>
</tr>
<tr>
<td></td>
<td>sipv2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>voice-class sip</td>
<td>Associates the dial peer with the specified keepalive group profile. The dial peer is monitored by CUBE according to the parameters defined by this profile.</td>
</tr>
<tr>
<td></td>
<td>options-keepalive profile</td>
<td></td>
</tr>
<tr>
<td></td>
<td>keepalive-group-profile-id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# voice-class sip options-keepalive profile 171</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>session server-group</td>
<td>Associates the dial peer with the specified keepalive group profile. The dial peer is monitored by the device according to the parameters defined by this profile.</td>
</tr>
<tr>
<td></td>
<td>server-group-id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session server-group 151</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>end</td>
<td>Exits dial peer configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>show voice class</td>
<td>Displays information about voice class server group.</td>
</tr>
<tr>
<td></td>
<td>sip-options-keepalive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>keepalive-group-profile-id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# show voice class sip-options-keepalive 171</td>
<td></td>
</tr>
</tbody>
</table>
Configuration Examples For SIP Out-of-Dialog OPTIONS Ping Group

Example: SIP Out-of-Dialog OPTIONS Ping for Group of SIP Endpoints

!Configuring the SIP profile
Device(config)# voice class sip-profiles 100
Device(config-class)# request INVITE sip-header SIP-Req-URI modify "; SIP/2.0" ";user=phone SIP/2.0"

!Configuring the SIP Keepalive Group
Device(config)# voice class sip-options-keepalive 171
Device(config-class)# transport tcp
Device(config-class)# sip-profile 100
Device(config-class)# down-interval 30
Device(config-class)# up-interval 60
Device(config-class)# retry 5
Device(config-class)# description Target New York
Device(config-class)# exit

!Configuring an outbound SIP Dial Peer
Device(config)# dial-peer voice 123 voip
Device(config-dial-peer)# session protocol sipv2
!Associating the Dial Peer with a keepalive profile group
Device(config-dial-peer)# voice-class sip options-keepalive profile 171
Device(config-dial-peer)# end

!Verifying the Keepalive group configurations
Device# show voice class sip-options-keepalive 171

Voice class sip-options-keepalive: 171 AdminStat: Up
Description: Target New York
Transport: system Sip Profiles: 100
Interval(seconds) Up: 60 Down: 30
Retry: 5

<table>
<thead>
<tr>
<th>Peer Tag</th>
<th>Server Group</th>
<th>OOD SessID</th>
<th>OOD Stat</th>
<th>IfIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: SIP Out-of-dialog OPTIONS Ping for Group of SIP Servers

!Configuring the Server Group
Device(config)# voice class server-group 151
Device(config-class)# ipv4 10.1.1.1 preference 1
Device(config-class)# ipv4 10.1.1.2 preference 2
Device(config-class)# ipv4 10.1.1.3 preference 3
Device(config-class)# hunt-scheme round-robin
Device(config-class)# description It has 3 entries
Device(config-class)# exit

!Configuring an E164 pattern map class
Device(config)# voice class e164-pattern-map 3000
Device(config-class)# e164 300
Configuring an outbound SIP dial peer.

```text
Device(config)# dial-peer voice 181 voip
Device(config-dial-peer)# destination e164-pattern-map 3000
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session server-group 151
Device(config-dial-peer)# voice-class sip options-keepalive profile 171
Device(config-dial-peer)# end
```

Verifying the Keepalive group configurations

```
Device# show voice class sip-options-keepalive 171
Voice class sip-options-keepalive: 171 AdminStat: Up
  Description: Target New York
  Transport: system Sip Profiles: 100
  Interval(seconds) Up: 60 Down: 30
  Retry: 5

  Peer Tag | Server Group | OOD SessID | OOD Stat | IfIndex
----------|-------------|-----------|--------|-------
  123      |             |           |        |      
  181      | 151         |           | Busy   | 106

  Server Group: 151 OOD Stat: Busy
    OOD SessID | OOD Stat
    ----------|--------
    1         | Busy
    2         | Busy
    3         | Busy

  OOD SessID: 1 OOD Stat: Busy
    Target: ipv4:10.1.1.1
    Transport: system Sip Profiles: 100

  OOD SessID: 2 OOD Stat: Busy
    Target: ipv4:10.1.1.2
    Transport: system Sip Profiles: 100

  OOD SessID: 3 OOD Stat: Busy
    Target: ipv4:10.5.0.1
    Transport: system Sip Profiles: 100
```

---

**Additional References**

**Related Documents**

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice commands</td>
<td>Cisco IOS Voice Command Reference</td>
</tr>
<tr>
<td>Cisco IOS Commands</td>
<td>Cisco IOS Master Command List, All Releases</td>
</tr>
<tr>
<td>SIP Configuration</td>
<td></td>
</tr>
<tr>
<td>Configuring SIP</td>
<td></td>
</tr>
</tbody>
</table>
Feature Information for SIP Out-of-dialog OPTIONS Ping Group

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 27: Feature Information for SIP Out-of-dialog OPTIONS Ping Group

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Out-of-dialog OPTIONS Ping Group</td>
<td>Cisco IOS XE Release 3.11S 15.4(1)T</td>
<td>This feature groups the monitoring of SIP dial peers endpoints and servers by consolidating SIP Out-Of-Dialog (OOD) Options of dial peers with the similar SIP OOD Options ping setup. The following commands were introduced or modified: <code>voice class sip-options-keepalive</code>, <code>description</code>, <code>transport</code>, <code>sip-profiles</code>, <code>down-interval</code>, <code>up-interval</code>, <code>voice-class sip options-keepalive profile</code>, <code>retry</code>, <code>show voice class sip-options-keepalive</code>.</td>
</tr>
</tbody>
</table>
Feature Information for SIP Out-of-dialog OPTIONS Ping Group
CHAPTER 15

Configure TCL IVR Applications

This chapter shows you how to configure Interactive Voice Response (IVR) using the Tool Command Language (TCL) scripts. The Cisco IOS Release 12.1(3)T release introduces TCL IVR Version 2.0 with several feature enhancements to the Cisco IVR functionality. This chapter contains the following sections:

- Tcl IVR Overview, on page 165
- Tcl IVR Enhancements, on page 166
- MGCP Scripting, on page 166
- RTSP Client Implementation, on page 167
- TCL IVR Prompts Played on IP Call Legs, on page 168
- TCL Verbs, on page 169
- TCL IVR Prerequisite Tasks, on page 172
- TCL IVR Configuration Tasks List, on page 172
- Configuring the Call Application for the Dial Peer, on page 173
- Configuring TCL IVR on the Inbound POTS Dial Peer, on page 175
- Configuring TCL IVR on the Inbound VoIP Dial Peer, on page 177
- Configuring MGCP Scripting, on page 179
- Verifying TCL IVR Configuration, on page 182
- TCL IVR Configuration Examples, on page 183
- TCL IVR for Gateway1 (GW1) Configuration Example, on page 183
- TCL IVR for GW2 Configuration Example, on page 186
- MGCP Scripting Configuration Example, on page 188

Tcl IVR Overview

IVR consists of simple voice prompting and digit collection to gather caller information for authenticating the user and identifying the destination. IVR applications can be assigned to specific ports or invoked on the basis of DNIS. An IP public switched telephone network gateway can have several IVR applications to accommodate many different gateway services, and you can customize the IVR applications to present different interfaces to the various callers.

IVR systems provide information in the form of recorded messages over telephone lines in response to user input in the form of spoken words, or more commonly dual tone multifrequency (DTMF) signalling. For
example, when a user makes a call with a debit card, an IVR application is used to prompt the caller to enter a specific type of information, such as an account number. After playing the voice prompt, the IVR application collects the predetermined number of touch tones and then places the call to the destination phone or system.

IVR uses TCL scripts to gather information and to process accounting and billing. For example, a TCL IVR script plays when a caller receives a voice-prompt instruction to enter a specific type of information, such as a personal identification number (PIN). After playing the voice prompt, the TCL IVR application collects the predetermined number of touch tones and sends the collected information to an external server for user authentication and authorization.

---

**Note**

Audio playback is not supported when Secure Real-Time Transport Protocol (SRTP) is used with TCL IVR applications.

---

**Tcl IVR Enhancements**

Since the introduction of the Cisco IVR technology, the software has undergone several enhancements. TCL IVR Version 2.0 is made up of separate components that are described individually in the sections that follow. The enhancements are as follows:

- Media Gateway Control Protocol (MGCP) scripting package implementation
- Real Time Streaming Protocol (RTSP) client implementation
- TCL IVR prompt playout and digit collection on IP call legs
- New TCL verbs to utilize RTSP and MGCP scripting features

The enhancements add scalability and enable the TCL IVR scripting functionality on VoIP legs. In addition, support for RTSP enables VoIP gateways to play messages from RTSP-compliant announcement servers. The addition of these enhancements also reduces the CPU load and saves memory on the gateway because no packetization is involved. Larger prompts can be played, and the use of an external audio server is allowed.

---

**Note**

TCL IVR 2.0 removed the signature locking mechanism requirement.

---

**MGCP Scripting**

TCL IVR Version 2.0 infrastructure is greatly enhanced with the addition of support for MGCP using the application package model. MGCP defines application packages to run scripts on the media gateways. These application packages initiate scripts on the gateways and receive return values after execution completes. MGCP scripting allows external call agents (CAs) to instruct a media gateway to run an TCL IVR script in order to perform a specific task and return the end result. For example, you can request and collect the PIN and account number from a caller.

Two previously released Cisco VoIP features that can be implemented are the Debit Card for Packet Telephony and TCL IVR. Both features use the TCL scripting language. The TCL scripts that run with MGCP are written
in TCL IVR API Version 2.0 and are able to receive calls through hand off. MGCP scripts can run any TCL command.

Figure 23: MGCP Control of TCL IVR Scripts, on page 167 displays the CA controlling the TCL IVR scripts. MGCP is the protocol that is running on the CA. The RTSP server is configured to interact with the gateways that have TCL IVR scripts installed and running. The RADIUS server running authentication, authorization, and accounting (AAA) also interacts with the gateways.

Figure 23: MGCP Control of TCL IVR Scripts

RTSP Client Implementation

RTSP is an application-level protocol used for control over the delivery of data that has real-time properties. Using RTSP also enables an external RTSP server to play announcements and interact with voice mail servers. It provides an extensive framework to enable control and to perform on-demand delivery of real-time data. For example, RTSP is used to control the delivery of audio streams from an audio server.

If you use an RTSP server in your network with VoIP gateways, a scripting application, (for example, an MGCP script) can run on the gateway and connect calls with audio streams from an external audio server. Using RTSP also has the following benefits:

- Reduces the CPU load
- Allows large prompts to be played that previously demanded high CPU usage from the gateway
- Saves memory on the gateway because no packetization is involved
- Allows use of an external audio server which removes the limitation on the number of prompts that can be played out and on the size of the prompt
TCL IVR Prompts Played on IP Call Legs

TCL IVR Version 2.0 scripts can be configured for incoming plain old telephone service (POTS) or VoIP call legs to play announcements to the user or collect user input (digits). With TCL IVR Version 2.0 the prompts can be triggered from both the PSTN side of the call leg and the IP side of the call leg. This feature enables the audio files (or prompts) to be played out over the IP network.

TCL IVR scripts played toward a VoIP call leg are subject to the following conditions:

• G.711 mu-law encoding must be used when prompts are played.

• G.711 mu-law encoding must also be used for the duration of these calls, even after prompt playout has completed.

• Digital signaling protocols (DSPs) cannot be on the IP call leg so the script cannot initiate a tone.

• When an TCL IVR script is used to collect digits on a VoIP call leg, one of the following DTMF relay methods must be used.
  
  • For H.323 protocol configured on the call leg, use one of the following DTMF relay methods: Cisco proprietary RTP, H.245 Alphanumeric IE, or H.245 Signal IE
  
  • For SIP protocol configured on the call leg, use Cisco proprietary RTP

---

For additional information about the `dtmf-relay` command, refer to the Cisco IOS Voice Command Reference - D through I.

IVR 2.0 enables the system to accept calls initiated from the IP side of the network using G.711, and terminate calls to the terminating gateway using the same codec. Figure 24: IVR Control of Scripts on an IP Call Leg, on page 169 displays the TCL IVR application on the gateways controlling the scripts. IP phones can also originate a call to a gateway running an TCL IVR script.
TCL Verbs

TCL IVR, Version 2.0, delivers a new set of TCL verbs and scripts that replace the previous TCL version. The new TCL verbs enable the user to:

- Utilize the RTSP audio servers
- Develop TCL scripts that interact with the IVR application
- Pass events to the Media Gateway Controller, which is a call agent

TCL IVR Version 2.0 is not backward compatible with the IVR 1.0 scripts. The MGCP scripting package can only be implemented using the new TCL verbs.

Note

For in-depth information about the TCL 2.0 verb set and how to develop scripts, refer to Cisco.com (Related Documentation index).

TCL IVR scripts use the TCL verbs to interact with the gateway during call processing in order to collect the required digits—for example, to request the PIN or account number for the caller. The TCL scripts are the default scripts for all Cisco voice features using IVR. TCL scripts are configured to control calls coming into or going out of the gateway.

Note

Ensure that you have loaded the version of TCL scripts that support IVR Version 2.
The TCL IVR scripts shown below are listed as an example of the types of scripts available to be downloaded from the cisco.com Software Center. For a complete list of scripts, it is recommended that you check the Software Center.

Cisco provides the following IVR scripts:

- **fax_hop_on_1**—Collects digits from the redialer, such as account number and destination number. When a call is placed to an H.323 network, the set of fields (configured in the call information structure) are "entered", "destination", and "account".

- **clid_authen**—Authenticates the call with automatic number identification (ANI) and DNIS numbers, collects the destination data, and makes the call.

- **clid_authen_npw**—Performs as clid_authen, but uses a null password when authenticating, rather than DNIS numbers.

- **clid_authen_collect**—Authenticates the call with ANI and DNIS numbers and collects the destination data. If authentication fails, it collects the account and password.

- **clid_authen_col_npw**—Performs as clid_authen_collect, but uses a null password and does not use or collect DNIS numbers.

- **clid_col_npw_3**—Performs as clid_authen_col_npw except with that script, if authentication with the digits collected (account and PIN) fails, the clid_authen_col_npw script just plays a failure message (auth_failed.au) and then hangs up. The clid_col_npw_3 script allows two failures, then plays the retry audio file (auth_retry.au) and collects the account and PIN again.

- The caller can interrupt the message by entering digits for the account number, triggering the prompt to tell the caller to enter the PIN. If authentication fails the third time, the script plays the audio file auth_fail_final.au, and hangs up.

**Table 28: clid_col_npw_3 Script Prompt Audio Files**, on page 170 lists the prompt audio files associated with the clid_col_npw_3 script.

**Table 28: clid_col_npw_3 Script Prompt Audio Files**

<table>
<thead>
<tr>
<th>Audio Filename</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>flash:enter_account.au</td>
<td>Asks the caller to enter an account number. Played as the first request.</td>
</tr>
<tr>
<td>flash:auth_fail_retry.au</td>
<td>Asks the caller to reenter the account number. Plays after two failures.</td>
</tr>
<tr>
<td>flash:enter_pin.au</td>
<td>Asks the caller to enter a PIN.</td>
</tr>
<tr>
<td>flash:enter_destination.au</td>
<td>Asks the caller to enter a destination phone number.</td>
</tr>
<tr>
<td>flash:auth_fail_final.au</td>
<td>Informs the caller that the account number authorization has failed three times.</td>
</tr>
</tbody>
</table>

**Table 29: Additional clid_col_npw_3 Script Audio Files**, on page 170 lists additional audio files associated with the clid_col_npw_3 script.

**Table 29: Additional clid_col_npw_3 Script Audio Files**

<table>
<thead>
<tr>
<th>Audio Filename</th>
<th>Action</th>
</tr>
</thead>
</table>
auth_fail_retry.au
Informs the caller that authorization failed. Prompts the caller to reenter the account number followed by the pound sign (#).

auth_fail_final.au
Informs the caller, "I'm sorry, your account number cannot be verified. Please hang up and try again."

- clid_col_npw_npw—Tries to authenticate by using ANI, null as the user ID, user, and user password pair. If that fails, it collects an account number and authenticates with account and null. It allows three tries for the caller to enter the account number before ending the call with the authentication failed audio file. If authentication succeeds, it plays a prompt to enter the destination number.

Table 30: clid_col_npw_npw Script Audio Files, on page 171 lists the audio files associated with the clid_col_npw_npw script.

Table 30: clid_col_npw_npw Script Audio Files

<table>
<thead>
<tr>
<th>Audio Filename</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>flash:enter_account.au</td>
<td>Asks the caller to enter the account number the first time.</td>
</tr>
<tr>
<td>flash:auth_fail_retry.au</td>
<td>Asks the caller to reenter the account number after first two failures.</td>
</tr>
<tr>
<td>flash:enter_destination.au</td>
<td>Asks the caller to enter the destination phone number.</td>
</tr>
<tr>
<td>flash:auth_fail_final.au</td>
<td>Informs the caller that the account number authorization has failed three times.</td>
</tr>
</tbody>
</table>

- clid_col_dnis_3.tcl—Authenticates the caller ID three times. First it authenticates the caller ID with DNIS. If that is not successful, it attempts to authenticate with the caller PIN up to three times.

- clid_col_npw_3.tcl—Authenticates with null. If authentication is not successful, it attempts to authenticate by using the caller PIN up to 3 times.

- clid_4digits_npw_3.tcl—Authenticates with null. If the authentication is not successful, it attempts to authenticate with the caller PIN up to 3 times using the 14-digit account number and password entered together.

- clid_4digits_npw_3_cli.tcl—Authenticates the account number and PIN respectively by using ANI and null. The number of digits allowed for the account number and password are configurable through the CLI. If the authentication fails, it allows the caller to retry. The retry number is also configured through the CLI.

- clid_authen_col_npw_cli.tcl—Authenticates the account number and PIN respectively using ANI and null. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.

- clid_authen_collect_cli.tcl—Authenticates the account number and PIN by using ANI and DNIS. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.

- clid_col_npw_3_cli.tcl—Authenticates by using ANI and null for account and PIN respectively. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI.
TCL IVR Prerequisite Tasks

Before you configure your Cisco gateway to support TCL IVR, you must perform the following prerequisite tasks:

• Configure VoIP to support H.323-compliant gateways—meaning that in addition to the basic configuration tasks, such as configuring dial peers and voice ports, you must configure specific devices in your network to act as gateways.

• Configure a TFTP server to perform storage and retrieval of the audio files, which are required by the Debit Card gateway or other features requiring TCL IVR scripts and audio files.

• Download the appropriate TCL IVR script from the Cisco.com. Use the `copy` command to copy your audio file (.au file) to your Flash memory, and the `audio-prompt load` command to read it into RAM. When you use TCL IVR applications, the gateway needs to know the URL where the TCL script can be found, as well as the URL of any audio file you want to use. Cisco IOS File System (IFS) is used to read the files, so any IFS-supported URLs can be used, which includes TFTP, FTP, or a pointer to a device on the router. During configuration of the application, you specify the URLs for the script and for the audio prompt. See the "Using URLs in IVR Scripts" chapter in the TCL IVR API Version 2.0 Programmer's Guide for more information.

• Make sure that your audio files are in the proper format. The TCL IVR prompts require audio file (.au) format of 8-bit, u-law, and 8-khz encoding. To encode your own audio files, we recommend that you use one of these two audio tools (or a tool of similar quality):
  • Cool Edit, manufactured by Syntrillium Software Corporation
  • AudioTool, manufactured by Sun Microsystems

• Make sure that your access platform has a minimum of 16 MB Flash and 128MB of DRAM memory.

• Install and configure the appropriate RADIUS security server in your network. The version of RADIUS that you are using must be able to support IETF-supported vendor specific attributes (VSAs), which are implemented by using IETF RADIUS attribute 26.

TCL IVR Configuration Tasks List

Before starting the software configuration tasks for the TCL IVR Version 2.0 features, complete the following preinstallation tasks:

• Download the TCL scripts and audio files to be used with this feature from the Cisco.com.
• Store the TCL scripts and audio files on a TFTP server configured to interact with your gateway access server.

• Create the TCL IVR application script to use with the **call application voice** command when configuring IVR using TCL scripts. You create this application first and store it on a server or location where it can be retrieved by the access server.

• Define the call flow and pass the defined parameter values to the application. Depending on the TCL script you select, these values can include the language of the audio file and the location of the audio file. **Table 28: clid_col_npw_3 Script Prompt Audio Files, on page 170** lists the TCL scripts and the parameter values they require.

• Associate the application to the incoming POTS or VoIP dial peer.

---

**Note**

When an IVR script is used to detect a "long #" from a caller connected to the H.323 call leg, the DTMF method used must either be Cisco proprietary RTP or DTMF relay using H.245 signal IE. DTMF relay using H.245 alphanumeric IE does not report the actual duration of the digit, causing long pound (#) detection to fail.

---

**Configuring the Call Application for the Dial Peer**

**Before you begin**

You must configure the application that interacts with the dial peer before you configure the dial peer. The dial peer collects digits from the caller and uses the application you have created. Use the call application voice command as shown in the table that follows. Each command line is optional depending on the type of action desired or the digits to be collected.

To configure the application, enter the following commands in global configuration mode:

**SUMMARY STEPS**

1. **call application voice name url**
2. **call application voice name language digit language**
3. **call application voice name pin-length number**
4. **call application voice name retry-count number**
5. **call application voice name uid-length number**
6. **call application voice name set-location language category location**

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>call application voice name url</strong></td>
<td>Defines the name of the application to be used with your TCL IVR script. The <em>url</em> argument specifies the location of the file and the access protocol. An example is as follows: <code>flash:scripts/session.tcl</code></td>
</tr>
</tbody>
</table>

Example:

Router(config)# **call application voice name url**
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>tftp://dirt/sarvi/scripts/session.tcl  ftp://sarvi-ultra/scripts/session.tcl slot0:scripts/tcl/session..tcl</td>
<td>You can only configure a url if the application named name has not been configured.</td>
</tr>
</tbody>
</table>

**Step 2**

**call application voice name language digit language**

**Example:**

```bash
Router(config)# call application voice name language digit language
```

Specifies the language used by the audio files. An example is: `call application voice test language 1 en`. The arguments are as follows:

- `digit`—Specifies zero (0) through 9.
- `language`—Specifies two characters that represent a language. For example, "en" for English, "sp" for Spanish, and "ch" for Mandarin. Enter `aa` to represent all.

**Step 3**

**call application voice name pin-length number**

**Example:**

```bash
Router(config)# call application voice name pin-length number
```

Defines the number of characters in the PIN for the designated application. Values are from 0 through 10.

**Step 4**

**call application voice name retry-count number**

**Example:**

```bash
Router(config)# call application voice name retry-count number
```

Defines the number of times a caller is permitted to reenter the PIN for the designated application. Values are from 1 through 5.

**Step 5**

**call application voice name uid-length number**

**Example:**

```bash
Router(config)# call application voice name uid-length number
```

Defines the number of characters allowed to be entered for the user ID for the designated application. Values are from 1 through 20.

**Step 6**

**call application voice name set-location language category location**

**Example:**

```bash
Router(config)# call application voice nameset-locationlanguage category location
```

Defines the location, language, and category of the audio files for the designated application. An example is: `set-location en 1 tftp://server dir/audio filename`.

---

**What to do next**

The following table lists TCL script names and the corresponding parameters that are required for each TCL scripts.
### Configuring TCL IVR on the Inbound POTS Dial Peer

**Before you begin**

Configuring gw-accounting and AAA are not always required for POTS dial peer configuration. It is dependent upon the type of application that is being used with TCL IVR. For example, the Pre-Paid Calling Card feature requires accounting and the authentication caller ID application does not.

<table>
<thead>
<tr>
<th>TCL Script Name</th>
<th>Description—Summary</th>
<th>Commands to Configure</th>
</tr>
</thead>
<tbody>
<tr>
<td>clid_4digits_npw_3_cli.tcl</td>
<td>Authenticates the account number and PIN using ANI and null. The allowed length of digits is configurable through the CLI. If the authentication fails, it allows the caller to retry. The retry number is also configured through the CLI.</td>
<td>call application voice uid-len min = 1, max = 20, default - 10 call application voice pin-len min = 0, max - 10, default = 4 call application voice retry-count min = 1, max = 5, default = 3</td>
</tr>
<tr>
<td>clid_authen_col_npw_cli.tcl</td>
<td>Authenticates the account number and PIN using ANI and null. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.</td>
<td>call application voice retry-count min = 1, max = 5, default = 3</td>
</tr>
<tr>
<td>clid_authen_collect_cli.tcl</td>
<td>Authenticates the account number and PIN using ANI and DNIS. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.</td>
<td>call application voice retry-count min = 1, max = 5, default = 3</td>
</tr>
<tr>
<td>clid_col_npw_3_cli.tcl</td>
<td>Authenticates using ANI and null for account and PIN. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI.</td>
<td>call application voice retry-count min = 1, max = 5, default = 3</td>
</tr>
<tr>
<td>clid_col_npw_npw_cli.tcl</td>
<td>Authenticates using ANI and null for account and PIN. If authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected together.</td>
<td>call application voice retry-count min = 1, max = 5, default = 3</td>
</tr>
</tbody>
</table>
To configure the inbound POTS dial peer, use the following commands beginning in global configuration mode:

**SUMMARY STEPS**

1. `aaa new-model`
2. `gw-accounting h323`
3. `aaa authentication login h323 radius`
4. `aaa accounting connection h323 start-stop radius`
5. `radius-server host ip-address auth-port number acct-port number`
6. `radius-server key key`
7. `dial-peer voice number pots`
8. `application name`
9. `destination-pattern string`
10. `session target`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | `aaa new-model` | (Optional) Enables AAA security and accounting services.  
Example:  
Router(config)# `aaa new-model` |
Example:  
Router(config)# `gw-accounting h323` |
| Step 3 | `aaa authentication login h323 radius` | (Optional) Defines a method list called H.323 where RADIUS is defined as the only method of login authentication.  
Example:  
Router(config)# `aaa authentication login h323 radius` |
| Step 4 | `aaa accounting connection h323 start-stop radius` | (Optional) Defines a method list called H.323 where RADIUS is used to perform connection accounting, providing start-stop records.  
Example:  
Router(config)# `aaa accounting connection h323 start-stop radius` |
| Step 5 | `radius-server host ip-address auth-port number acct-port number` | Identifies the RADIUS server and the ports that will be used for authentication and accounting services.  
Example:  
Router(config)# `radius-server host ip-address auth-port number acct-port number` |
| Step 6 | `radius-server key key` | Specifies the password used between the gateway and the RADIUS server.  
Example:  
Router(config)# `radius-server key key` |
<table>
<thead>
<tr>
<th>Step 7</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>dial-peer voice number pots</code></td>
<td>Enters dial-peer configuration mode to configure the incoming POTS dial peer. The <em>number</em> argument is a tag that uniquely identifies the dial peer.</td>
</tr>
<tr>
<td>Step 8</td>
<td>application <em>name</em></td>
<td>Associates the TCL IVR application with the incoming POTS dial peer. Enter the selected TCL IVR application name.</td>
</tr>
<tr>
<td>Step 9</td>
<td>destination-pattern <em>string</em></td>
<td>Enters the telephone number associated with this dial peer. The <em>pattern</em> argument is a series of digits that specify the E.164 or private dialing plan telephone number. Valid entries are numbers from zero (0) through nine and letters A through D. The following special characters can be entered in the string:</td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# dial-peer voice number pots</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(dial-peer)# application name</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config-dial-peer)# destination-pattern string</code></td>
<td></td>
</tr>
<tr>
<td>Step 10</td>
<td>session target</td>
<td>Specifies the session target IP address.</td>
</tr>
<tr>
<td></td>
<td><code>Router(config-dial-peer)# session target</code></td>
<td></td>
</tr>
</tbody>
</table>

### Configuring TCL IVR on the Inbound VoIP Dial Peer

#### Before you begin

To configure the inbound VoIP dial peer, use the following commands beginning in global configuration mode:

```
SUMMARY STEPS

1. dial-peer voice 4401 voip
2. application application-name
3. destination-pattern pattern
4. session protocol sipv2
5. session target
6. dtmf-relay cisco-rtp
7. codec g711ulaw

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters the dial-peer configuration mode and identifies the call leg.</td>
</tr>
<tr>
<td>dial-peer voice 4401 voip</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# dial-peer voice 4401 voip</td>
<td></td>
</tr>
</tbody>
</table>

**Step 2**

Specifies the name of the application and script to use.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>application application-name</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-dial-peer)# application application-name</td>
<td></td>
</tr>
</tbody>
</table>

**Step 3**

Enters the destination pattern.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination-pattern pattern</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-dial-peer)# destination-pattern pattern</td>
<td></td>
</tr>
</tbody>
</table>

**Step 4**

Specifies the session protocol. The default session protocol is H.323. The sipv2 argument enables SIP.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
</tbody>
</table>

**Step 5**

Specifies the session target IP address.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>session target</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-dial-peer)# session target</td>
<td></td>
</tr>
</tbody>
</table>

**Step 6**

Specifies the DTMF relay method. The keyword cisco-rtp specifies H.323 and SIP. Other keywords that are available only for H.323 are h245-alphanumeric and h245-signal.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtmf-relay cisco-rtp</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-dial-peer)# dtmf-relay cisco-rtp</td>
<td></td>
</tr>
</tbody>
</table>

**Note** If digit collection from this VoIP call leg is required, the command dtmf-relay is required. The default is no dtmf-relay.

**Step 7**

Specifies the voice codec.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>codec g711ulaw</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-dial-peer)# codec g711ulaw</td>
<td></td>
</tr>
</tbody>
</table>

**Note** If the configured application will be playing prompts to the VoIP call leg, the g711ulaw keyword is required.
Configuring MGCP Scripting

Before you begin

To perform MGCP scripting, you must enable the MGCP script package. Enable the script in global configuration mode by entering the `mgcp package-capability script package` command. The example MGCP configuration shown in this section is for DS0s on T1 lines. The configuration tasks are as follows:

- Enabling the MGCP service on the DS0 groups
- Enabling the other MGCP packages
- Configuring the call agent address and other MGCP parameters

To configure MGCP scripting, use the following commands beginning in global configuration mode:

**SUMMARY STEPS**

1. `mgcp`
2. `mgcp request timeout timeout`
3. `mgcp request retries count`
4. `mgcp call-agent { ipaddr | hostname } { port }`
5. `mgcp max-waiting-delay value`
6. `mgcp restart-delay value`
7. `mgcp vad`
8. `mgcp package-capability { as-package | dtmf-package | gm-package | rtp-package | trunk-package }
9. `mgcp default-package { as-package | dtmf-package | gm-package | rtp-package | trunk-package }
10. `mgcp quality-threshold { hwm-jitter-buffer value | hwm-latency value | hwm-packet-loss value | lwm-jitter-buffer value | lwm-latency value | lwm-packet-loss value }
11. `mgcp playout { adaptive init-value min-valuemax-value } | { fixed init-value }
12. `mgcp codec type { packetization-period value }
13. `mgcp ip-tos { high-reliability | high-throughput | low-cost | low-delay | precedence value }
14. `controller t1 slot#`
15. `framing`
16. `clock source type`
17. `linecode type`
18. `ds0-group n timeslots range type signaling-type service mgcp`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>mgcp</code></td>
<td>Starts the MGCP daemon.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router(config)# mgcp</code></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>-----------------</td>
</tr>
<tr>
<td>2</td>
<td>mgcp request timeout <em>timeout</em></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp request timeout timeout</td>
</tr>
<tr>
<td>3</td>
<td>mgcp request retries <em>count</em></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp request retries count</td>
</tr>
<tr>
<td>4</td>
<td>mgcp call-agent {ipaddr</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp call-agent {ipaddr</td>
</tr>
<tr>
<td>5</td>
<td>mgcp max-waiting-delay <em>value</em></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp max-waiting-delay value</td>
</tr>
<tr>
<td>6</td>
<td>mgcp restart-delay <em>value</em></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp restart-delay value</td>
</tr>
<tr>
<td>7</td>
<td>mgcp vad</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp vad</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp package-capability {as-package</td>
</tr>
<tr>
<td>9</td>
<td>mgcp default-package {as-package</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp default-package {as-package</td>
</tr>
<tr>
<td>10</td>
<td>mgcp quality-threshold {hwm-jitter-buffer <em>value</em></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router(config)# mgcp quality-threshold {hwm-jitter-buffer value</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>hwm-packet-loss value</td>
<td>Tunes the jitter buffer packet size used for MGCP connections.</td>
</tr>
<tr>
<td>lwm-jitter-buffer value</td>
<td></td>
</tr>
<tr>
<td>lwm-latency value</td>
<td></td>
</tr>
<tr>
<td>lwm-packet-loss value</td>
<td></td>
</tr>
</tbody>
</table>

**Step 11**

**mgcp playout { adaptive init-value min-value max-value } | { fixed init-value }**

*Example:*

Router(config)# mgcp playout { adaptive init-value min-value max-value } | { fixed init-value }

**Step 12**

**mgcp codec type [ packetization-period value ]**

*Example:*

Router(config)# mgcp codec type { packetization-period value }

**Step 13**

**mgcp ip-tos { high-reliability | high-throughput | low-cost | low-delay | precedence value }**

*Example:*

Router(config)# mgcp ip-tos { high-reliability | high-throughput | low-cost | low-delay | precedence value }

**Step 14**

**controller t1 slot#**

*Example:*

Router(config)# controller t1 slot#

**Step 15**

**framing**

*Example:*

Router(config-controller)# framing type

**Step 16**

**clock source type**

*Example:*

Router(config-controller)# clock source type

**Step 17**

**linecode type**

*Example:*

Router(config-controller)# linecode type

**Step 18**

**ds0-group n timeslots range type signaling-type service mgcp**

*Example:*

Router(config-controller)# ds0-group n timeslots range type signaling-type service mgcp

**Example:**

Router(config)# mgcp playout { adaptive init-value min-value max-value } | { fixed init-value }

Configure the default codec type.

Configure the IP type of service for MGCP connections.

Uses the controller configuration mode for the T1 controller in the specified slot.

Configures the framing type.

Configures the clock source.

Configures the line coding.

Configures the DS0s to support MGCP.
Verifying TCL IVR Configuration

Before you begin
You can verify TCL IVR configuration by performing the following tasks:

• To verify TCL IVR configuration parameters, use the show running-config command.
• To display a list of all voice applications, use the show call application summary command.
• To display a list of all voice applications, use the show call application summary command.
• To show the contents of the script configured, use the show call application voice command.
• To verify that the operational status of the dial peer, use the show dial-peer voice command.

To verify the TCL IVR configuration, perform the following steps:

Step 1
Enter the show call application voice summary command to verify that the newly created applications are listed. The example output follows

Router# show call application voice summary

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>NEW::Basic app to do DID, or supply dialtone.</td>
</tr>
<tr>
<td>fax_hop_on</td>
<td>Script to talk to a fax redialer</td>
</tr>
<tr>
<td>clid_authen</td>
<td>Authenticate with (ani, dnis)</td>
</tr>
<tr>
<td>clid_authen_collect</td>
<td>Authenticate with (ani, dnis), collect if that fails</td>
</tr>
<tr>
<td>clid_authen_npw</td>
<td>Authenticate with (ani, NULL)</td>
</tr>
<tr>
<td>clid_authen_col_npw</td>
<td>Authenticate with (ani, NULL), collect if that fails</td>
</tr>
<tr>
<td>clid_col_npw_3</td>
<td>Authenticate with (ani, NULL), and 3 tries collecting</td>
</tr>
<tr>
<td>clid_col_npw_npw</td>
<td>Authenticate with (ani, NULL) and 3 tries without pw</td>
</tr>
<tr>
<td>SESSION</td>
<td>Default system session application</td>
</tr>
<tr>
<td>hotwo</td>
<td>tftp://hostname/scripts/nb/nb_handoffTwoLegs.tcl</td>
</tr>
<tr>
<td>hoone</td>
<td>tftp://hostname/scripts/nb/nb_dohandoff.tcl</td>
</tr>
<tr>
<td>hodest</td>
<td>tftp://hostname/scripts/nb/nb_handoff.tcl</td>
</tr>
<tr>
<td>clid</td>
<td>tftp://hostname/scripts/tcl_ivr/clid_authen_collect.tcl</td>
</tr>
<tr>
<td>db102</td>
<td>tftp://hostname/scripts/1.02/debitcard.tcl</td>
</tr>
<tr>
<td>*hw</td>
<td>tftp://171.69.184.xxx/tr_hello.tcl</td>
</tr>
<tr>
<td>name</td>
<td>description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>*hw1</td>
<td>tftp://san*tr_db</td>
</tr>
</tbody>
</table>

TFTP://171.69.184.235/tr_debitcard.answer.tcl

TCL Script Version 2.0 supported.
TCL Script Version 1.1 supported.

**Note** In the output shown, an asterisk (*) in an application indicates that this application was not loaded successfully. Use the `show call application voice` command with the `name` argument to view information for a particular application.

**Step 2** Enter the `show dial-peer voice` command with the `peer tag` argument and verify that the application associated with the dial peer is correct.

**Step 3** Enter the `show running-config` command to display the entire configuration.

---

**TCL IVR Configuration Examples**

Use the `show running-config` command to display the entire gateway configuration. Figure 25: Example Configuration Topology, on page 183 shows the type of topology used in the configuration for the example.

In this example configuration, GW1 is running TCL IVR for phone A, and GW2 is running TCL IVR for phone B.

This section provides the following configuration examples:

*Figure 25: Example Configuration Topology*

---

**TCL IVR for Gateway 1 (GW1) Configuration Example**

The following output is the result of using the `show running-config` command:

GW1
Router# show running-config

Building configuration...

Current configuration:

! Last configuration change at 08:39:29 PST Mon Jan 10 2000 by lab
! version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
hostname GW1
logging buffered 100000 debugging
aaa new-model
aaa authentication login default local group radius
aaa authentication login h323 group radius
aaa authentication login con none
aaa authorization exec h323 group radius
aaa accounting connection h323 start-stop group radius
enable password xxx
username lab password 0 lab
resource-pool disable
clock timezone PST -8
ip subnet-zero
ip host baloo 1.14.124.xxx
ip host dirt 223.255.254.254
ip host rtspserver3 1.14.1xx.2
ip host rtspserver1 1.14.1xx.2
mgcp package-capability trunk-package
mgcp default-package trunk-package
isdn switch-type primary-net5
isdn voice-call-failure 0
tftp://dirt/hostname/WV/en_new/
call application voice debit_card tftp://dirt/Router/scripts.new/app_debitcard.tcl
call application voice debit_card uid-len 6
call application voice debit_card_language 1 en
call application voice debit_card_language 2 ch
call application voice debit_card_set-location ch 0 tftp://dirt/hostname/WV/ch_new/
call application voice debit_card_rtsp_set-location en 0 tftp://dirt/hostname/WV/en_new/
call application voice debit_card_rtsp tftp://IVR 2.0/scripta.new/app_debitcard.tcl
call application voice debit_card_rtsp_uid-len 6
call application voice debit_card_rtsp_language 1 en
call application voice debit_card_rtsp_language 2 ch
call application voice debit_card_rtsp_set-location ch 0 rtsp://rtspserver1:554/
call application voice debit_card_rtsp_set-location en 0 rtsp://rtspserver1:554/
mta receive maximum-recipients 0
controller E1 0
clock source line primary
pri-group timeslots 1-31
controller E1 1
controller E1 2
controller E1 3

gw-accounting h323
gw-accounting h323 vsa
gw-accounting voip

interface Ethernet0
ip address 1.14.128.35 255.255.255.xxx
no ip directed-broadcast
h323-gateway voip interface
h323-gateway voip id gkl ipaddr 1.14.128.19 1xxx
h323-gateway voip h323-id gw1@cisco.com
h323-gateway voip tech-prefix 5#
!
interface Serial0:15
  no ip address
  no ip directed-broadcast
  isdn switch-type primary-net5
  isdn incoming-voice modem
  fair-queue 64 256 0
  no cdp enable
!
interface FastEthernet0
  ip address 16.0.0.1 255.255.xxx.0
  no ip directed-broadcast
  duplex full
  speed auto
  no cdp enable
!
  ip classless
  ip route 0.0.0.0 0.0.0.0 1.14.128.33
  ip route 1.14.xxx.0 255.xxx.255.xxx 16.0.0.2
  no ip http server
!
  radius-server host 1.14.132.2 auth-port 1645 acct-port 1646
  radius-server key cisco
  radius-server vsa send accounting
  radius-server vsa send authentication
!
  voice-port 0:D
    cptone DE
!
  dial-peer voice 200 voip
    incoming called-number 53
    destination-pattern 34.....
    session target ipv4:16.0.0.2
dtmf-relay h245-alphanumeric
    codec g711ulaw
!
  dial-peer voice 102 pots
    application debit_card_rtsp
    incoming called-number 3450072
    shutdown
    destination-pattern 53.....
    port 0:D
!
  dial-peer voice 202 voip
    shutdown
    destination-pattern 34.....
    session protocol sip
    session target ipv4:16.0.0.2
dtmf-relay cisco-rtp
    codec g711ulaw
!
  dial-peer voice 101 pots
    application debit_card
    incoming called-number 3450070
    destination-pattern 53.....
    port 0:D
!
  gateway
!
  line con 0
  exec-timeout 0 0
  transport input none
line aux 0
line vty 0 4
  password xxx
!
ntp clock-period 17180740
ntp server 1.14.42.23
end

GW1#

TCL IVR for GW2 Configuration Example

The following output is the result of using the show running-config command:

GW2#
  Router# show running-config

Building configuration...

Current configuration:
!
! Last configuration change at 08:41:12 PST Mon Jan 10 2000 by lab
!
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname GW2
!
logging buffered 100000 debugging
aaa new-model
aaa authentication login default local group radius
aaa authentication login h323 group radius
aaa authorization exec h323 group radius
aaa accounting connection h323 start-stop group radius
!
username lab password xxx
username 111119 password xxx
!
resource-pool disable
!
clock timezone PST -8
ip subnet-zero
ip host radiusserver2 1.14.132.2
ip host radiusserver1 1.14.138.11
ip host baloo 1.14.124.254
ip host rtspserver2 1.14.136.2
ip host dirt 223.255.254.254
ip host rtspserver3 1.14.126.2
!
mgcp package-capability trunk-package
mgcp default-package trunk-package
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
call application voice clid_authen_sky
tftp://dirt/hostname/sky_scripts/clid_authen_collect_cli_sky.tcl

  call application voice rtsp_demo tftp://dirt/hostname/sky_scripts/rtsp_demo.tcl
tftp://dirt/hostname/WV/en_new/
call application voice debit_card tftp://dirt/IVR 2.0/scripts.new/app_debitcard.tcl
call application voice debit_card uid-len 6
call application voice debit_card language 1 en
call application voice debit_card language 2 ch
call application voice debit_card set-location ch 0 tftp://dirt/hostname/WV/ch_new/
call application voice debit_card set-location en 0 tftp://dirt/hostname/WV/en_new/
call application voice clid_authen_rtsp tftp://dirt/IVR
2.0/scripts.new/app_clid_authen_collect_cli_rtsp.tcl

call application voice clid_authen_rtsp location rtsp://rtspserver2:554/
call application voice clid_authen1 tftp://dirt/IVR
2.0/scripts.new/app_clid_authen_collect_cli_rtsp.tcl
call application voice clid_authen1 location tftp://dirt/hostname/WV/en_new/
call application voice clid_authen1 uid-len 6
call application voice clid_authen1 retry-count 4
mta receive maximum-recipients 0
!
controller T1 0
  framing esf
  clock source line primary
  linecode b8zs
  pri-group timeslots 1-24
!
controller T1 1
  clock source line secondary 1
!
controller T1 2
!
controller T1 3
!
gw-accounting h323

gw-accounting h323 vsa

gw-accounting voip
!
interface Ethernet0
  ip address 1.14.xxx.4 255.255.xxx.240
  no ip directed-broadcast
  h323-gateway voip interface
  h323-gateway voip id gk2 ipaddr 1.14.xxx.18 1719
  h323-gateway voip h323-id gw2@cisco.com
  h323-gateway voip tech-prefix 3#
!
interface Serial0:23
  no ip address
  no ip directed-broadcast
  isdn switch-type primary-5ess
  isdn incoming-voice modem
  fair-queue 64 256 0
  no cdp enable
!
interface FastEthernet0
  ip address 16.0.0.2 255.xxx.255.0
  no ip directed-broadcast
duplex full
  speed 10
  no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 1.14.xxx.5
ip route 1.14.xxx.32 255.255.xxx.240 16.0.0.1
no ip http server
!
radius-server host 1.14.132.2 auth-port 1645 acct-port 1646
radius-server key cisco  
radius-server vsa send accounting  
radius-server vsa send authentication  
!  
voice-port 0:D  
!  
dial-peer voice 100 voip  
application debit_card  
incoming called-number 34  
shutdown  
destination-pattern 53.....  
session target ras  
dtmf-relay h245-alphanumeric  
codec g711ulaw  
!  
dial-peer voice 200 pots  
incoming called-number 30001  
destination-pattern 3450070  
port 0:D  
prefix 50070  
!  
dial-peer voice 101 voip  
application debit_card  
incoming called-number 34.....  
shutdown  
session protocol sipv2  
session target ipv4:16.0.0.1  
dtmf-relay cisco-rtp  
codec g711ulaw  
!  
dial-peer voice 102 voip  
incoming called-number 34.....  
destination-pattern 53.....  
session target ipv4:16.0.0.1  
dtmf-relay h245-alphanumeric  
codec g711ulaw  
!  
gateway  
!  
line con 0  
exec-timeout 0 0  
transport input none  
line aux 0  
line vty 0 4  
password xxx  
!  
ntp clock-period 17180933  
ntp server 1.14.42.23  
end  
GW2#  

MGCP Scripting Configuration Example
The following example displays only the MGCP specific portion of the configuration:

!  
mgcp  
mgcp request timeout 10000  
mgcp request retries 1  
mgcp call-agent 1.14.138.11
mgcp restart-delay 10
mgcp codec g723ar63 packetization-period 30
mgcp vad
mgcp package-capability gm-package
mgcp package-capability dtmf-package
mgcp package-capability trunk-package
mgcp package-capability rtp-package
mgcp package-capability as-package
mgcp package-capability script-package
mgcp default-package trunk-package
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
mta receive maximum-recipients 0
!
controller T1 0
   framing esf
   clock source line primary
   linecode b8zs
   ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 1
   framing esf
   clock source line secondary 1
   linecode b8zs
   ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 2
   framing esf
   linecode b8zs
   ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 3
   framing esf linecode b8zs
   ds0-group 0 timeslots 1-24 type none service mgcp
!
end
VoIP for IPv6

This document describes VoIP in IPv6 (VoIPv6), a feature that adds IPv6 capability to existing VoIP features. This feature adds dual-stack (IPv4 and IPv6) support on voice gateways and media termination points (MTPs), IPv6 support for Session Initiation Protocol (SIP) trunks, and support for Skinny Client Control Protocol (SCCP)-controlled analog voice gateways. In addition, the Session Border Controller (SBC) functionality of connecting a SIP IPv4 or H.323 IPv4 network to a SIP IPv6 network is implemented on a Cisco UBE to facilitate migration from VoIPv4 to VoIPv6.

- Finding Feature Information, on page 191
- Prerequisites for VoIP for IPv6, on page 191
- Restrictions for Implementing VoIP for IPv6, on page 192
- Information About VoIP for IPv6, on page 193
- How to Configure VoIP for IPv6, on page 199
- Configuration Examples for VoIP over IPv6, on page 222
- Troubleshooting Tips for VoIP for IPv6, on page 223
- Verifying and Troubleshooting Tips, on page 223
- Feature Information for VoIP for IPv6, on page 241

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for VoIP for IPv6

- Cisco Express Forwarding for IPv6 must be enabled.
- Virtual routing and forwarding (VRF) is not supported in IPv6 calls.

Cisco Unified Border Element

- Cisco IOS Release 12.4(22)T or a later release must be installed and running on your Cisco UBE.
Cisco Unified Border Element (Enterprise)

- Cisco IOS XE Release 3.3S or a later release must be installed and running on your Cisco ASR 1000 Series Router.

Restrictions for Implementing VoIP for IPv6

The following are the restrictions for Cisco UBE features:

Media Flow–Through

- Video call flows with Alternative Network Address Types (ANAT) are not supported.
- WebEx call flow with ANAT are not supported (Cisco UBE does not support ANAT on Video and Application media types).

SDP Pass-Through

- Supports only Early Offer (EO)–Early Offer (EO) and Delayed Offer (DO)–Delayed Offer (DO) call flows.
- Delayed Offer–Early Offer call flow falls back to Delayed Offer–Delayed Offer call flow.
- Supplementary services are not supported on SDP Pass-Through.
- Transcoding and DTMF interworking are not supported.

Note

The above SDP Pass–Through restrictions are applicable for both IPv4 and IPv6.

- SDP Pass–Through does not support the dual-stack functionality.
- ANAT call flows does not support IPv4-to-IPv6 and IPv6-to-IPv4 Media interworking.

UDP Checksum

- CEF and process options are not supported on ASR1000 series routers.
- None option is partially supported on ISR–G2.

Media Anti–Trombone

- Media Anti–Trombone is not enabled if the initial call before tromboning is in Flow–Around (FA) mode.
- Media Anti–Trombone supports only symmetric media address type interworking (IPv4-IPv4 or IPv6-IPv6 media) with or without ANAT.
- Does not provide support for IPv4-IPv6 interworking cases with or without ANAT because Cisco UBE cannot operate in FA mode post tromboning.
Information About VoIP for IPv6

SIP Features Supported on IPv6

The Session Initiation Protocol (SIP) is an alternative protocol developed by the Internet Engineering Task Force (IETF) for multimedia conferencing over IP.

The Cisco SIP functionality enables Cisco access platforms to signal the setup of voice and multimedia calls over IP networks. SIP features also provide advantages in the following areas:

- Protocol extensibility
- System scalability
- Personal mobility services
- Interoperability with different vendors

A SIP User Agent (UA) operates in one of the following three modes:

- IPv4-only: Communication with only IPv6 UA is unavailable.
- IPv6-only: Communication with only IPv4 UA is unavailable.
- Dual-stack: Communication with only IPv4, only IPv6 and dual-stack UAs are available.

Dual-stack SIP UAs use Alternative Network Address Transport (ANAT) grouping semantics:

- Includes both IPv4 and IPv6 addresses in the Session Description Protocol (SDP).
- Is automatically enabled in dual-stack mode (can be disabled if required).
- Requires media to be bound to an interface that have both IPv4 and IPv6 addresses.
- Described in RFC 4091 and RFC 4092 (RFC 5888 describes general SDP grouping framework).

SIP UAs use “sdp-anat” option tag in the Required and Supported SIP header fields:

- Early Offer (EO) INVITE using ANAT semantics places “sdp-anat” in the Require header.
- Delayed Offer (DO) INVITE places “sdp-anat” in the Supported header.

SIP Signaling and Media Address Selection:

- Source address for SIP signaling is selected based on the destination signaling address type configured in the session-target of the outbound dial-peer:
  - If signaling bind is configured, source SIP signaling address is chosen from the bound interface.
  - If signaling bind is not configured, source SIP signaling address is chosen based on the best address in the UA to reach the destination signaling address.

SDP may or may not use ANAT semantics:
• When ANAT is used, media addresses in SDP are chosen from the interface media that is configured. When ANAT is not used, media addresses in SDP are chosen from the interface media that is configured OR based on the best address to reach the destination signaling address (when no media bind is configured).

**SIP Voice Gateways in VoIPv6**

Session Initiation Protocol (SIP) is a simple, ASCII-based protocol that uses requests and responses to establish communication among the various components in the network and to ultimately establish a conference between two or more endpoints.

In addition to the already existing features that are supported on IPv4 and IPv6, the SIP Voice Gateways support the following features:

- **History–Info**: The SIP History–info Header Support feature provides support for the history-info header in SIP INVITE messages only. The SIP gateway generates history information in the INVITE message for all forward and transferred calls. The history-info header records the call or dialog history. The receiving application uses the history-info header information to determine how and why the call has reached it.

  For more information, refer to the “SIP History INFO” section in the *Cisco Unified Border Element (Enterprise) SIP Support Configuration Guide*.

- **Handling 181/183 Responses with/without SDP**: The Handling 181/183 Responses with/without SDP feature provides support for SIP 181 (Call is Being Forwarded) and SIP 183 (Session Progress) messages either globally or on a specific dial-peer. Also, you can control when the specified SIP message is dropped based on either the absence or presence of SDP information.

  For more information, refer to “SIP–Enhanced 180 Provisional Response Handling” section in the *Cisco Unified Border Element Configuration Guide*.

- **Limiting the Rate of Incoming SIP Calls per Dial-Peer (Call Spike)**: The call rate-limiting feature for incoming SIP calls starts working after a switch over in a SIP call. The rate-limiting is done for new calls received on the new Active. The IOS timers that track the call rate limits runs on active and standby mode and does not require any checkpoint. However, some statistics for calls rejected requires to be checked for the show commands to be consistent before and after the switchover.

- **PPI/PAI/Privacy and RPID Passing**: For incoming SIP requests or response messages, when the PAI or PPI privacy header is set, the SIP gateway builds the PAI or PPI header into the common SIP stack, thereby providing support to handle the call data present in the PAI or PPI header. For outgoing SIP requests or response messages, when the PAI or PPI privacy header is set, privacy information is sent using the PAI or PPI header.

  For more information, refer to the “Support for PAID PPID Privacy PCPID and PAURI Headers on Cisco UBE” section in the *Cisco Unified Border Element SIP Support Configuration Guide*.

- **SIP VMWI for FXS phones**: SIP provides visible message waiting indication (VMWI) on FXS phones. This feature provides users with the option to enable one message waiting indication (MWI): audible, visible, or both. The VMWI mechanism uses SIP Subscribe or Notify to get MWI updates from a virtual machine (VM) system, and then forwards updates to the FXS phone on the port.

  For more information, refer to the “Configuring SIP MWI Features” section in the *SIP Configuration Guide*.

- **SIP Session timer (RFC 4028)**: This feature 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. Two header fields can be defined: Session-Expires, which conveys the lifetime of the session, and Min-SE, which conveys the minimum allowed value for the session timer.

For more information, refer to the “SIP Session Timer Support” section in the Cisco Unified Border Element SIP Support Configuration Guide.

• **SIP Media Inactivity Detection**: The SIP Media Inactivity Detection Timer feature enables Cisco gateways to monitor and disconnect VoIP calls if no Real-Time Control Protocol (RTCP) packets are received within a configurable time period.

For more information, refer to the SIP Media Inactivity Timer section.

The SIP Voice Gateways feature is supported for analog endpoints that are connected to Foreign Exchange Station (FXS) ports or a Cisco VG224 Analog Phone Gateway and controlled by a Cisco call-control system, such as a Cisco Unified Communications Manager (Cisco Unified CM) or a Cisco Unified Communications Manager Express (Cisco Unified CME).

For more information on SIP Gateway features and information about configuring the SIP voice gateway for VoIPv6, see the Configuring VoIP for IPv6.

**VoIPv6 Support on Cisco UBE**

Cisco UBE in VoIPv6 adds IPv6 capability to VoIP features. This feature adds dual-stack support on voice gateways, IPv6 support for SIP trunks, support for SCCP-controlled analog voice gateways, support for real-time control protocol (RTCP) pass-through, and support for T.38 fax over IPv6.

For more information on these features, refer to the following:

• “Configuring Cisco IOS Gateways” section in the Deploying IPv6 in Unified Communications Networks with Cisco Unified Communications Manager

• “Trunks” section in Deploying IPv6 in Unified Communications Networks with Cisco Unified Communications Manager

• “SCCP-controlled analog voice gateways” section in the SCCP Controlled Analog (FXS) Ports with Supplementary Features in Cisco IOS Gateways

• “RTCP Pass-Through” section in Cisco UBE RTCP Voice Pass-Through for IPv6

• “T.38 fax over IPv6” section in Fax, Modem, and Text Support over IP Configuration Guide

Support has been added for audio calls in media Flow–Through (FT) and Flow–Around (FA) modes, High Density (HD) transcoding, Local Transcoding Interface (LTI), along with Voice Class Codec (VCC) support, support for Hold/Resume, REFER, re-INVITE, 302 based services, and support for media anti-trombone have been added to Cisco UBE.

Cisco UBE being a signaling proxy processes all signaling messages for setting up media channels. This enables Cisco UBE to affect the flow of media packets using the media flow-through and the media flow-around modes.

• Media FT and Media FA modes support the following call flows:
  • EO–to–EO
  • DO–to–DO
  • DO–to–EO
• **Media Flow-Through (FT):** In a media flow-through mode, between two endpoints, both signaling and media flows through the IP-to-IP Gateway (IPIP GW). The IPIP GW performs both signaling and media interworking between H.323/SIP IPv4 and SIP IPv6 networks.

*Figure 26: H.323/SIP IPv4 – SIP IPv6 interworking in media flow-through mode*

• **Media Flow-Around (FA):** Media flow-around provides the ability to have a SIP video call whereby signaling passes through Cisco UBE and media pass directly between endpoints bypassing the Cisco UBE.

*Figure 27: H.323/SIP IPv4 - SIP IPv6 interworking in media flow-around mode*

• **Assisted RTCP (RTCP Keepalive):** Assisted Real-time Transport Control Protocol (RTCP) enables Cisco UBE to generate RTCP keepalive reports on behalf of endpoints; however, endpoints, such as second generation Cisco IP phones (7940/7960) and Nortel Media Gateways (MG 1000T) do not generate any RTCP keepalive reports. Assisted RTCPs enable customers to use Cisco UBE to interoperate between endpoints and call control agents, such as Microsoft OCS/Lync so that RTCP reports are generated to indicate session liveness during periods of prolonged silence, such as call hold or call on mute.

The assisted RTCP feature helps Cisco UBE to generate standard RTCP keepalive reports on behalf of endpoints. RTCP reports determine the liveness of a media session during prolonged periods of silence, such as a call on hold or a call on mute.

• **SDP Pass–Through:** SDP is configured to pass through transparently at the Cisco UBE, so that both the remote ends can negotiate media independently of the Cisco UBE.

SDP pass-through is addressed in two modes:

• **Flow-through**—Cisco UBE plays no role in the media negotiation, it blindly terminates and re-originates the RTP packets irrespective of the content type negotiated by both the ends. This supports address hiding and NAT traversal.

• **Flow-around**—Cisco UBE neither plays a part in media negotiation, nor does it terminate and re-originates media. Media negotiation and media exchange is completely end-to-end.
For more information, refer to the “Configurable Pass-through of SIP INVITE Parameters” section in the Cisco Unified Border Element SIP Support Configuration Guide.

• **UDP Checksum for IPv6**: User Datagram Protocol (UDP) checksums provide data integrity for addressing different functions at the source and destination of the datagram, when a UDP packet originates from an IPv6 node.

• **IP Toll Fraud**: The IP Toll Fraud feature checks the source IP address of the call setup before routing the call. If the source IP address does not match an explicit entry in the configuration as a trusted VoIP source, the call is rejected.

For more information, refer to the “Configuring Toll Fraud Prevention” section in the Cisco Unified Communications Manager Express System Administrator Guide.

• **RTP Port Range**: Provides the capability where the port range is managed per IP address range. This feature solves the problem of limited number of rtp ports for more than 4000 calls. It enables combination of an IP address and a port as a unique identification for each call.

• **Hold/Resume**: Cisco UBE supports supplementary services such as Call Hold and Resume. An active call can be put in hold state and later the call can be resumed.

For more information, refer to the “Configuring Call Hold/Resume for Shared Lines for Analog Ports” section in Supplementary Services Features for FXS Ports on Cisco IOS Voice Gateways Configuration Guide.

• **Call Transfer (re-INVITE, REFER)**: Call transfer is used for conference calling, where calls can transition smoothly between multiple point-to-point links and IP level multicasting.

For more information, refer to the “Configurable Pass-through of SIP INVITE Parameters” section in the Cisco Unified Border Element SIP Support Configuration Guide.

• **Call Forward (302 based)**: SIP provides a mechanism for forwarding or redirecting incoming calls. A Universal Access Servers (UAS) can redirect an incoming INVITE by responding with a 302 message (moved temporarily).
  
  • Consumption of 302 at stack level is supported for EO-EO, DO-DO and DO-EO calls for all combination of IPv4/IPv6/ANAT.

  • Consumption of 302 at stack level is supported for both FT and FA calls.

For more information, refer to the “Configuring Call Transfer and Forwarding” section in Cisco Unified Communications Manager Express System Administrator Guide.

• **Media Antitrombone**: Antitromboning is a media signaling service in SIP entity to overcome the media loops. Media Trombones are media loops in a SIP entity due to call transfer or call forward. Media loops in Cisco UBE are not detected because Cisco UBE looks at both call types as individual calls and not calls related to each other.

Antitrombone service has to be enabled only when no media interworking is required in both legs. Media antitrombone is supported only when the initial call is in IPv4 to IPv4 or IPv6 to IPv6 mode only.

For more information, refer to the “Configuring Media Antitrombone” section in the Cisco Unified Border Element Protocol-Independent Features and Setup Configuration Guide.

• **RE-INVITE Consumption**: The Re-INVITE/UPDATE consumption feature helps to avoid interoperability issues by consuming the mid-call Re-INVITEs/UPDATEs from Cisco UBE. As Cisco
UBE blocks RE-INVITE / mid-call UPDATE, remote participant is not made aware of the SDP changes, such as Call Hold, Call Resume, and Call transfer.

For more information, refer to the “Cisco UBE Mid-call Re-INVITE/UPDATE Consumption” section in the Cisco Unified Border Element Protocol-Independent Features and Setup Configuration Guide.

• **Address Hiding**: The address hiding feature ensures that the Cisco UBE is the only point of signaling and media entry/exit in all scenarios. When you configure address-hiding, signaling and media peer addresses are also hidden from the endpoints, especially for supplementary services when the Cisco UBE passes REFER/3xx messages from one leg to the other.

For more information, refer to the “Configuring Address Hiding” section in the SIP-to-SIP Connections on a Cisco Unified Border Element.

• **Header Passing**: Header Pass through enables header passing for SIP INVITE, SUBSCRIBE and NOTIFY messages; disabling header passing affects only incoming INVITE messages. Enabling header passing results in a slight increase in memory and CPU utilization.

For more information, refer to the “SIP-to-SIP Connections on a Cisco Unified Border Element” section in the SIP-to-SIPConnections on Cisco Unified Border Element.

• **Refer–To Passing**: The Refer-to Passing feature is enabled when you configure refer-to-passing in Refer Pass through mode and the supplementary service SIP Refer is already configured. This enables the received refer-to header in Refer Pass through mode to move to the outbound leg without any modification. However, when refer-to-passing is configured in Refer Consumption mode without configuring the supplementary-service SIP Refer, the received Refer-to URI is used in the request-URI of the triggered invite.

For more information, refer to the “Configuring Support for Dynamic REFER Handling on Cisco UBE” section in the Cisco Unified Border Element SIP Configuration Guide.

• **Error Pass-through**: The SIP error message pass through feature allows a received error response from one SIP leg to pass transparently over to another SIP leg. This functionality will pass SIP error responses that are not yet supported on the Cisco UBE or will preserve the Q.850 cause code across two sip call-legs.

For more information, refer to the “Configuring SIP Error Message Passthrough” section in the Cisco Unified Border Element SIP Support Configuration Guide.

• **SIP UPDATE Interworking**: The SIP UPDATE feature allows a client to update parameters of a session (such as, a set of media streams and their codecs) but has no impact on the state of a dialog. UPDATE with SDP will support SDP Pass through, media flow around and media flow through. UPDATE with SDP support for SIP to SIP call flows is supported in the following scenarios:
  • Early Dialog SIP to SIP media changes.
  • Mid Dialog SIP to SIP media changes.

For more information, refer to the “SIP UPDATE Message per RFC 3311” section in the Cisco Unified Border Element SIP Support Configuration Guide.

• **SIP OPTIONS Ping**: The OPTIONS ping mechanism monitors the status of a remote Session Initiation Protocol (SIP) server, proxy or endpoints. Cisco UBE monitors these endpoints periodically.

For more information, refer to the “Cisco UBE Out-of-dialog OPTIONS Ping for Specified SIP Servers or Endpoints” section in the Configuration of SIP Trunking for PSTN Access (SIP-to-SIP) Configuration Guide.
**Configurable Error Response Code in OPTIONS Ping:** Cisco UBE provides an option to configure the error response code when a dial peer is busyed out because of an Out-of-Dial OPTIONS ping failure.

For more information, refer to the “Configuring an Error Response Code upon an Out-of-Dialog OPTIONS Ping Failure” section in the Cisco Unified Border Element SIP Support Configuration Guide.

**SIP Profiles:** SIP profiles create a set of provisioning properties that you can apply to SIP trunk.

**Dynamic Payload Type Interworking (DTMF and Codec Packets):** The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature provides dynamic payload type interworking for dual tone multifrequency (DTMF) and codec packets for Session Initiation Protocol (SIP) to SIP calls. The Cisco UBE interworks between different dynamic payload type values across the call legs for the same codec. Also, Cisco UBE supports any payload type value for audio, video, named signaling events (NSEs), and named telephone events (NTEs) in the dynamic payload type range 96 to 127.

For more information, refer to the “Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls” section in the Cisco Unified Border Element (Enterprise) Protocol-Independent Features and Setup Configuration Guide.

**Audio Transcoding using Local Transcoding Interface (LTI):** Local Transcoding Interface (LTI) is an interface created to remove the requirement of SCCP client for Cisco UBE transcoding.

For information, refer to Cisco Unified Border Element 9.0 Local Transcoding Interface (LTI).

**Voice Class Codec (VCC) with or without Transcoding:** The Voice Class Codec feature supports basic and all Re-Invite based supplementary services like call-hold/resume, call forward, call transfer, where if any mid-call codec changes, Cisco UBE inserts/removes/modifies the transcoder as needed.

Support for negotiation of an Audio Codec on each leg of a SIP–SIP call on the Cisco UBE feature supports negotiation of an audio codec using the Voice Class Codec (VCC) infrastructure on Cisco UBE. VCC supports SIP-SIP calls on Cisco UBE and allows mid-call codec change for supplementary services.

---

**How to Configure VoIP for IPv6**

### Configuring VoIP for IPv6

SIP is a simple, ASCII-based protocol that uses requests and responses to establish communication among the various components in the network and to ultimately establish a conference between two or more endpoints.

Users in a SIP network are identified by unique SIP addresses. A SIP address is similar to an e-mail address and is in the format of sip:userID@gateway.com. The user ID can be either a username or an E.164 address. The gateway can be either a domain (with or without a hostname) or a specific Internet IPv4 or IPv6 address.

A SIP trunk can operate in one of three modes: SIP trunk in IPv4-only mode, SIP trunk in IPv6-only mode, and SIP trunk in dual-stack mode, which supports both IPv4 and IPv6.

A SIP trunk uses the Alternative Network Address Transport (ANAT) mechanism to exchange multiple IPv4 and IPv6 media addresses for the endpoints in a session. ANAT is automatically enabled on SIP trunks in dual-stack mode. The ANAT Session Description Protocol (SDP) grouping framework allows user agents (UAs) to include both IPv4 and IPv6 addresses in their SDP session descriptions. The UA is then able to use any of its media addresses to establish a media session with a remote UA.
A Cisco Unified Border Element can interoperate between H.323/SIP IPv4 and SIP IPv6 networks in media flow-through mode. In media flow-through mode, both signaling and media flows through the Cisco Unified Border Element, and the Cisco Unified Border Element performs both signaling and media interoperation between H.323/SIP IPv4 and SIP IPv6 networks (see the figure below).

**Figure 28: H.323/SIP IPv4–SIP IPv6 Interoperating in Media Flow-Through Mode**

---

**Shutting Down or Enabling VoIPv6 Service on Cisco Gateways**

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `voice service voip`
4. `shutdown [ forced]`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: <code>Device&gt; enable</code></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: <code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td><code>voice service voip</code></td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example: <code>Device(config)# voice service voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>shutdown [ forced]</code></td>
<td>Shuts down or enables VoIP call services.</td>
</tr>
<tr>
<td>Example: <code>Device(config-voi-serv)# shutdown forced</code></td>
<td></td>
</tr>
</tbody>
</table>
Shutting Down or Enabling VoIPv6 Submodes on Cisco Gateways

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `voice service voip`
4. `sip`
5. `call service stop [forced]`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt;` enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters SIP configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> call service stop [forced]</td>
<td>Shuts down or enables VoIPv6 for the selected submode.</td>
</tr>
<tr>
<td>Example: Device(config-serv-sip)# call service stop</td>
<td></td>
</tr>
</tbody>
</table>

**Configuring the Protocol Mode of the SIP Stack**

**Before you begin**

SIP service should be shut down before configuring the protocol mode. After configuring the protocol mode as IPv6, IPv4, or dual-stack, SIP service should be reenabled.

**SUMMARY STEPS**

1. `enable`
2. configure terminal
3. sip-ua
4. protocol mode ipv4 | ipv6 | dual-stack [preference ipv4 | ipv6]}

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>sip-ua</td>
<td>Enters SIP user agent configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>protocol mode ipv4</td>
<td>ipv6</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-sip-ua)# protocol mode dual-stack</td>
<td></td>
</tr>
</tbody>
</table>

Example: Configuring the SIP Trunk

This example shows how to configure the SIP trunk to use dual-stack mode, with IPv6 as the preferred mode. The SIP service must be shut down before any changes are made to protocol mode configuration.

Device(config)# sip-ua
Device(config-sip-ua)# protocol mode dual-stack preference ipv6

Disabling ANAT Mode

ANAT is automatically enabled on SIP trunks in dual-stack mode. Perform this task to disable ANAT in order to use a single-stack mode.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. no anat

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  Example: Device> enable |
| **Step 2** configure terminal | Enters global configuration mode.  
  Example: Device# configure terminal |
| **Step 3** voice service voip | Enters voice service VoIP configuration mode.  
  Example: Device(config)# voice service voip |
| **Step 4** sip | Enters SIP configuration mode.  
  Example: Device(config-voi-serv)# sip |
| **Step 5** no anat | Disables ANAT on a SIP trunk.  
  Example: Device(conf-serv-sip)# no anat |

**Verifying SIP Gateway Status**

**Before you begin**

To verify the status of SIP Gateway, use the following commands

**SUMMARY STEPS**

1. show sip-ua calls  
2. show sip-ua connections  
3. show sip-ua status

**DETAILED STEPS**

**Step 1** show sip-ua calls

The **show sip-ua calls** command displays active user agent client (UAC) and user agent server (UAS) information on SIP calls:
Device# show sip-ua calls
SIP UAC CALL INFO
Call 1
SIP Call ID : 8368ED08-1C2A11DD-80078908-BA2972D0@2001::21B:D4FF:FED7:B000
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 2000
Called Number : 1000
Bit Flags : 0xC04018 0x100 0x0
CC Call ID : 2
Source IP Address (Sig): 2001:DB8:0:ABCD::1
Destn SIP Req Addr:Port : 2001:DB8:0:FFFF:5060
Destn SIP Resp Addr:Port: 2001:DB8:0:FFFF:5060
Destination Name : 2001::21B:D5FF:FE1D:6C00
Number of Media Streams : 1
Number of Active Streams: 1
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 2
Stream Type : voice-only (0)
Stream Media Addr Type : 1709707780
Negotiated Codec : (20 bytes)
Codec Payload Type : 18
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
Media Source IP Addr:Port: [2001::21B:D4FF:FED7:B000]:16504
Media Dest IP Addr:Port : [2001::21B:D5FF:FE1D:6C00]:19548
Options-Ping ENABLED:NO ACTIVE:NO
Number of SIP User Agent Client(UAC) calls: 1
SIP UAS CALL INFO
Number of SIP User Agent Server(UAS) calls: 0

Step 2 show sip-ua connections

Use the show sip-ua connections command to display SIP UA transport connection tables:

Example:

Device# show sip-ua connections udp brief
Total active connections : 1
No. of send failures : 0
No. of remote closures : 0
No. of conn. failures : 0
No. of inactive conn. ageouts : 0

Router# show sip-ua connections udp detail
Total active connections : 1
No. of send failures : 0
No. of remote closures : 0
No. of conn. failures : 0
No. of inactive conn. ageouts : 0

Note:
** Tuples with no matching socket entry
- Do 'clear sip <tcp[tls]/udp> conn t ipv4::<addr>:<port>'
  to overcome this error condition
++ Tuples with mismatched address/port entry
- Do 'clear sip <tcp[tls]/udp> conn t ipv4::<addr>:<port> id <connid>'
  to overcome this error condition
Remote-Agent:2001::21B:D5FF:FE1D:6C00, Connections-Count:1
Remote-Port Conn-Id Conn-State WriteQ-Size
Step 3  

**show sip-ua status**

Use the **show sip-ua status** command to display the status of the SIP UA:

**Example:**

```
Device# show sip-ua status
SIP User Agent Status
SIP User Agent for UDP : ENABLED
SIP User Agent for TCP : ENABLED
SIP User Agent for TLS over TCP : ENABLED
SIP User Agent bind status(signaling): DISABLED
SIP User Agent bind status(media): DISABLED
SIP early-media for 180 responses with SDP: ENABLED
SIP max-forwards : 70
SIP DNS SRV version: 2 (rfc 2782)
NAT Settings for the SIP-UA
Role in SDP: NONE
Check media source packets: DISABLED
Maximum duration for a telephone-event in NOTIFYs: 2000 ms
SIP support for ISDN SUSPEND/RESUME: ENABLED
Redirect (3xx) message handling: ENABLED
Reason Header will override Response/Request Codes: DISABLED
Out-of-dialog Refer: DISABLED
Presence support is DISABLED
**protocol mode is ipv6**
SDP application configuration:
Version line (v=) required
Owner line (o=) required
Timespec line (t=) required
Media supported: audio video image
Network types supported: IN
Address types supported: IP4 IP6
Transport types supported: RTP/AVP udptl
```

---

**RTCP Pass-Through**

IPv4 and IPv6 addresses embedded within RTCP packets (for example, RTCP CNAME) are passed on to Cisco UBE without being masked. These addresses are masked on the Cisco UBE ASR 1000.

The Cisco UBE ASR 1000 does not support printing of RTCP debugs.

---

**Note**

RTCP is passed through by default. No configuration is required for RTCP pass-through.

---

**Configuring IPv6 Support for Cisco UBE**

In Cisco UBE, IPv4-only and IPv6-only modes are not supported when endpoints are dual-stack. In this case, Cisco UBE must also be configured in dual-stack mode.
### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `sip-ua`
4. `protocol mode {ipv4 | ipv6 | dual-stack {preference {ipv4 | ipv6}}}
5. `end`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; <code>enable</code></td>
<td></td>
</tr>
</tbody>
</table>

| **Step 2**        | Enters global configuration mode. |
| `configure terminal` | |
| **Example:**      | |
| Device# `configure terminal` | |

| **Step 3**        | Enters SIP user-agent configuration mode. |
| `sip-ua`          | |
| **Example:**      | |
| Device(config)# `sip-ua` | |

| **Step 4**        | Configures the Cisco IOS SIP stack. |
| `protocol mode {ipv4 | ipv6 | dual-stack {preference {ipv4 | ipv6}}}` | |
| **Example:**      | • `protocol mode dual-stack preference {ipv4 | ipv6}` — Sets the IP preference when the ANAT command is configured. |
| Device(config-sip-ua)# `protocol mode ipv6` | |
| **Example:**      | • `protocol mode {ipv4 | ipv6}` — Passes the IPv4 or IPv6 address in the SIP invite. |
| Device(config-sip-ua)# `protocol mode ipv6` | |
| **Example:**      | • `protocol mode dual-stack` — Passes both the IPv4 addresses and the IPv6 addresses in the SIP invite and sets priority based on the far-end IP address. |

| **Step 5**        | Exits SIP user-agent configuration mode. |
| `end`             | |
| **Example:**      | |
| Device(conf-voi-serv)# `end` | |

### Verifying RTP Pass-Through

To enable RTCP packet-related debugging, use the following command

### SUMMARY STEPS

1. `debug voip rtcp packets`
Configuring the Source IPv6 Address of Signaling and Media Packets

Users can configure the source IPv4 or IPv6 address of signaling and media packets to a specific interface’s IPv4 or IPv6 address. Thus, the address that goes out on the packet is bound to the IPv4 or IPv6 address of the interface specified with the `bind` command.

The `bind` command also can be configured with one IPv6 address to force the gateway to use the configured address when the bind interface has multiple IPv6 addresses. The bind interface should have both IPv4 and IPv6 addresses to send out ANAT.

When you do not specify a bind address or if the interface is down, the IP layer still provides the best local address.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. sip
5. bind {control | media | all} source interface interface-id [ipv6-address ipv6-address]
**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>enable</td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>configure terminal</td>
<td></td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td>voice service voip</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters SIP configuration mode.</td>
</tr>
<tr>
<td>sip</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Binds the source address for signaling and media packets to the IPv6 address of a specific interface.</td>
</tr>
<tr>
<td>bind {control</td>
<td>media</td>
</tr>
<tr>
<td>Example: Device(config-serv-sip)# bind control source-interface FastEthernet 0/0</td>
<td></td>
</tr>
</tbody>
</table>

**Example: Configuring the Source IPv6 Address of Signaling and Media Packets**

Device(config)# voice service voip
Device(config-voi-serv)# sip
Device(config-serv-sip)# bind control source-interface FastEthernet 0/0

**Configuring the SIP Server**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. sip-ua
4. sip-server \{dns: host-name\} | \ipv4: ipv4-address\ | \ipv6: ipv6-address\ \{port-nums\}\}
5. keepalive target \{\ipv4: address | ipv6: address\}\{port\} \{dns: hostname\} \{tcp | tls\} \{udp\} \{secondary\}
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>sip-ua</td>
<td>Enters SIP user agent configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>sip-server {dns: host-name}</td>
<td>Configures a network address for the SIP server interface.</td>
</tr>
<tr>
<td></td>
<td>ipv4: ipv4-address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipv6: ipv6-address [:port-nums]}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-sip-ua)# sip-server ipv6: 2001:DB8:0:0:8:800:200C:417A</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>keepalive target {ipv4: address</td>
<td>Identifies SIP servers that will receive keepalive packets from the SIP gateway.</td>
</tr>
<tr>
<td></td>
<td>ipv6: address[:port]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dns: hostname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[tcp [ts]]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[udp] [secondary]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-sip-ua)# keepalive target ipv6: 2001:DB8:0:0:8:800:200C:417A</td>
<td></td>
</tr>
</tbody>
</table>

**Example: Configuring the SIP Server**

```plaintext
Device(config)# sip-ua
Device(config-sip-ua)# sip-server ipv6: 2001:DB8:0:0:8:800:200C:417A
```

### Configuring the Session Target

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice tag {mmoip | pots | vofr | voip}
4. destination pattern [+ string T]
### Configuring SIP Register Support

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**
Enable   
Example:
Device> enable | Enables privileged EXEC mode.  
- Enter your password if prompted. |
| **Step 2**
configure terminal   
Example:
Device# configure terminal | Enters global configuration mode. |
| **Step 3**
dial-peer voice tag {mmpi|pots|vofr|voip}   
Example:
Device(config)# dial-peer voice 29 voip | Defines a particular dial peer, specifies the method of voice encapsulation, and enters dial peer configuration mode. |
| **Step 4**
destination pattern [+ string T]   
Example:
Device(config-dial-peer)# destination-pattern 7777 | Specifies either the prefix or the full E.164 telephone number to be used for a dial peer. |
| **Step 5**
session target {ipv4: destination-address} ipv6: [destination-address] | Designates a network-specific address to receive calls from a VoIP or VoIPv6 dial peer. |

**Example: Configuring the Session Target**

Device(config)# dial-peer voice 29 voip  
Device(config-dial-peer)# destination-pattern 7777  
Device(config-dial-peer)# session target ipv6:2001:DB8:0:0:8:800:200C:417A

### Configuring SIP Register Support

#### SUMMARY STEPS

1. enable
2. configure terminal
3. sip-ua
4. registrar \{dns: address | ipv4: destination-address [:port] | ipv6: destination-address : port \} aor-domain expires seconds [tcp tls] type [secondary] [scheme string]
5. retry register retries
6. timers register milliseconds

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>sip-ua</td>
<td>Enters SIP user agent configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>registrar {dns: address</td>
<td>ipv4: destination-address [:port]</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-sip-ua)# registrar ipv6: 2001:DB8::1:20F:FF00:FE0B:2972 expires 3600 secondary</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>retry register retries</td>
<td>Configures the total number of SIP register messages that the gateway should send.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-sip-ua)# retry register 10</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>timers register milliseconds</td>
<td>Configures how long the SIP UA waits before sending register requests.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-sip-ua)# timers register 500</td>
<td></td>
</tr>
</tbody>
</table>

**Example: Configuring SIP Register Support**

Device(config)# sip-ua
Configuring Outbound Proxy Server Globally on a SIP Gateway

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. outbound-proxy {ipv4: ipv4-address | ipv6: ipv6-address | dns: host : domain} [: port-number]

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable                                      | Enables privileged EXEC mode.  
   **Example:**                                                 |         |
| Device> enable                                         | • Enter your password if prompted. |
| **Step 2** configure terminal                           | Enters global configuration mode.  
   **Example:**                                                 |         |
| Device# configure terminal                              |         |
| **Step 3** voice service voip                           | Enters voice service VoIP configuration mode.  
   **Example:**                                                 |         |
| Device(config)# voice service voip                      |         |
| **Step 4** sip                                          | Enters sip configuration mode.  
   **Example:**                                                 |         |
| Device(config-voi-serv)# sip                            |         |
| **Step 5** outbound-proxy {ipv4: ipv4-address | ipv6: ipv6-address | dns: host : domain} [: port-number] | Specifies the SIP outbound proxy globally for a Cisco IOS voice gateway using an IPv6 address.  
   **Example:**                                                 |         |
| Device(config-serv-sip)#outbound-proxy ipv6: 2001:DB8:0:0:8:800:200C:417A |         |
Configuring UDP Checksum

SUMMARY STEPS

1. enable
2. configure terminal
3. ipv6 udp checksum [process | cef | none]
4. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> ipv6 udp checksum [process</td>
<td>cef</td>
</tr>
<tr>
<td>Example:</td>
<td>Use the following keywords with the <strong>ipv6 udp checksum</strong> command:</td>
</tr>
<tr>
<td>Device(config)# ipv6 udp checksum process</td>
<td>• process: Packets are punted to the process switching path for checksum validation.</td>
</tr>
<tr>
<td></td>
<td>• cef: The UDP checksum validation is done in the CEF path.</td>
</tr>
<tr>
<td></td>
<td>• none: UDP checksum validation is not done for received media packets in the CEF path and there is no UDP checksum computation for transmitted media packets.</td>
</tr>
<tr>
<td><strong>Step 4</strong> exit</td>
<td>Exits global configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# exit</td>
<td></td>
</tr>
</tbody>
</table>
Configuring IP Toll Fraud

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. ip address trusted list
5. ipv6 $X:.$X:.$X:.$X::X
6. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ip address trusted list</td>
<td>Enters IP address trusted list configuration mode. You can add unique and multiple IP addresses for incoming VoIP (H.323/SIP) calls to a list of trusted IP addresses.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-voi-serv)# ip address trusted list</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> ipv6 $X:.$X:.$X:.$X::X</td>
<td>Enters IPv6 addresses for toll fraud prevention.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(cfg-iptrust-list)# ipv6 2001:DB8::/48</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Exits trusted list configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(cfg-iptrust-list)# end</td>
<td></td>
</tr>
</tbody>
</table>
Configuring the RTP Port Range for an Interface

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. allow-connections sip to sip
5. media-address range range
6. rtp-port range range
7. exit
8. dial-peer voice tag voip
9. voice–class sip bind media source–interface interface
10. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> allow-connections sip to sip</td>
<td>Allows sip-to-sip connections under voice service voip configuration</td>
</tr>
<tr>
<td>Example:</td>
<td>mode for Cisco UBE.</td>
</tr>
<tr>
<td>Device(conf-voi-serv)# allow-connections sip to sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> media-address range range</td>
<td>Configures the media-address range, which enables the media</td>
</tr>
<tr>
<td>Example:</td>
<td>gateway to allocate the available free port for a given IP</td>
</tr>
<tr>
<td>Device(config-voi-serv)# media-address range 2001:DB8::/48</td>
<td>address within the address range.</td>
</tr>
<tr>
<td><strong>Step 6</strong> rtp-port range range</td>
<td>Configures the RTP port range.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
Configuring Message Waiting Indicator Server Address

### SUMMARY STEPS

1. enable  
2. configure terminal  
3. sip-ua  
4. mwi-server {ipv4: destination-address | ipv6: destination-address | dns: host-name} peer-tag [output-dial-peer-tag]  
5. end

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  | • Enter your password if prompted.  
  | Example:  
  | Device> enable |
### Configuring Voice Ports

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice-port *port number*
4. vmwi [fsk | dc-voltage]
5. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** | enable | Enables privileged EXEC mode.  
- Enter your password if prompted. |
| **Step 2** | configure terminal | Enters global configuration mode. |
| **Example:** | Device# configure terminal |
| **Step 3** | sip-ua | Enters SIP user-agent configuration mode. |
| **Example:** | Device(config)# sip-ua |
| **Step 4** | mwi-server {ipv4: destination-address | ipv6: destination-address | dns: host-name} peer-tag [output-dial-peer-tag] | Configures voice-mail server settings on a voice gateway or user agent.  
- ipv4/ipv6: destination-address—IP address of the voice-mail server.  
- dns: host-name—Host device housing the domain name server that resolves the name of the voice-mail server. The argument should contain the complete hostname to be associated with the target address; for example, dns:test.example.com.  
- peer-tag—Attaches an existing dial peer to SIP MWI service. |
| **Example:** | Device(config-sip-ua)# mwi-server ipv6 2001:DB8::/48 peer-tag 3 |
| **Step 5** | end | Exits SIP user-agent configuration mode and returns to global configuration mode. |
| **Example:** | Device(config-sip-ua)# end |
## Configuring Cisco UBE Mid-call Re-INVITE Consumption

### Configuring Passthrough of Mid-call Signalling

Perform this task to configure passthrough of mid-call signaling (as Re-invites) only when bidirectional media is added.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. Configure passthrough of mid-call signaling changes only when bidirectional media is added.
   - In Global VoIP SIP configuration mode
     
     midcall-signaling passthru media-change
   
   - In dial-peer configuration mode
     
     voice-class sip mid-call signaling passthru media-change
4. `end`

---

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>voice-port port number</code></td>
<td>Enters voiceport configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device(config)# voice-port 3</code></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>`vmwi [fsk</td>
<td>dc-voltage]`</td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device(config-voiceport)# vmwi fsk</code></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>end</code></td>
<td>Exits voice-port configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device(config-voiceport)# end</code></td>
<td></td>
</tr>
</tbody>
</table>
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> Configure passthrough of mid-call signaling changes only when bidirectional media is added.</td>
<td></td>
</tr>
<tr>
<td>- In Global VoIP SIP configuration mode midcall-signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td>- In dial-peer configuration mode voice-class sip mid-call signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> In Global VoIP SIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# midcall-signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> In Dial-peer configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 2 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip mid-call signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

### Configuring Passthrough SIP Messages at Dial Peer Level

Perform this task to configure passthrough SIP messages at the dial-peer level. You need to perform this task at the dial-peer level to consume all media-related mid-call Re-INVITEs/UPDATEs.

**Note** If the Cisco UBE Mid-call Re-INVITE/UPDATE consumption feature is configured on global and dial-peer level, dial-peer level takes precedence.

### SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice dial-peer tag voip
4. voice-class sip mid-call signaling passthru media-change
5. exit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>dial-peer voice dial-peer tag  voip</td>
<td>Enters dial-peer voice configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 2 voip</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>voice-class sip mid-call signaling passthru media-change</td>
<td>Passes through SIP messages that involve media change.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# voice-class sip mid-call signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>exit</td>
<td>Exits dial-peer voice configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>

**Configuring H.323 IPv4-to-SIPv6 Connections in a Cisco UBE**

An organization with an IPv4 network can deploy a Cisco UBE on the boundary to connect with the service provider’s IPv6 network (see the figure below). 

---

Cisco Unified Border Element Configuration Guide
A Cisco UBE can interoperate between H.323/SIP IPv4 and SIP IPv6 networks in media flow-through mode. In media flow-through mode, both signaling and media flows through the Cisco UBE, and the Cisco UBE performs both signaling and media interoperation between H.323/SIP IPv4 and SIP IPv6 networks (see the figure below).

The Cisco UBE feature adds IPv6 capability to existing VoIP features. This feature adds dual-stack support on voice gateways and MTP, IPv6 support for SIP trunks, and SCCP-controlled analog voice gateways. In addition, the SBC functionality of connecting SIP IPv4 or H.323 IPv4 network to a SIP IPv6 network is implemented on an Cisco UBE to facilitate migration from VoIPv4 to VoIPv6.

Before you begin
Cisco UBE must be configured in IPv6-only or dual-stack mode to support IPv6 calls.

Note
A Cisco UBE interoperates between H.323/SIP IPv4 and SIP IPv6 networks only in media flow-through mode.
SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. allow-connections *from type to to type*

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>voice service voip</td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>allow-connections <em>from type to to type</em></td>
<td>Allows connections between specific types of endpoints in a VoIPv6 network.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-voi-serv)# allow-connections h323 to sip</td>
<td>Arguments are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>from-type</em> -- Type of connection. Valid values: <em>h323</em>, <em>sip</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <em>to-type</em> -- Type of connection. Valid values: <em>h323</em>, <em>sip</em>.</td>
</tr>
</tbody>
</table>

Example: Configuring H.323 IPv4-to-SIPv6 Connections in a Cisco UBE

```bash
Device(config)# voice service voip
Device(config-voi-serv)# allow-connections h323 to sip
```

Configuration Examples for VoIP over IPv6

Example: Configuring the SIP Trunk

This example shows how to configure the SIP trunk to use dual-stack mode, with IPv6 as the preferred mode. The SIP service must be shut down before any changes are made to protocol mode configuration.
Troubleshooting Tips for VoIP for IPv6

**Media Flow-Through**

To enable all Session Initiation Protocol (SIP)-related debugging, use the `debug ccsip all` command in privileged EXEC mode.

To trace the execution path through the call control application programming interface (CCAPI), use the `debug voip ccapi inout` command.

**Media Flow-Around**

To enable all Session Initiation Protocol (SIP)-related debugging, use the `debug ccsip all` command.

To trace the execution path through the call control application programming interface (CCAPI), use the `debug voip ccapi inout` command.

**SDP Pass-Through**

To enable all Session Initiation Protocol (SIP)-related debugging (when the call is active in Pass through mode), use the `debug ccsip all` command.

**RTP Port Range**

To enable all Session Initiation Protocol (SIP)-related debugging, use the `debug ccsip all` command.

To enable debugging for Real-Time Transport Protocol (RTP) named event packets, use the `debug voip rtp` command.

**VMWI SIP**

To collect debug information only for signaling events, use the `debug vpm signal` command.

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) message tracing, use the `debug ccsip messages` command.

Verifying and Troubleshooting Tips

**Verifying Cisco UBE ANAT Call Flows**

To verify that media settings are enabled in the media flowthrough and media flow-around feature, use the following commands:

**SUMMARY STEPS**

1. `show call active voice brief`
2. show call active voice compact
3. show voip rtp connections

DETAILED STEPS

Step 1  show call active voice brief

Example:

Device# show call active voice brief

<ID>: <CallID> <start>ms.<index> <(start)> +<connect> pid:<peer_id> <dir> <addr> <state>
IP <ip>:<udp> rtt:<time>ms pl:<play>/<gap>ms lost:<lost>/<early>/<late>
delay:<last>ms <(max)>ms <(min)>ms <codec> <textrelay> <transcoded>
media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>
long duration call detected:<y/n> long duration call duration :<sec> timestamp:<time>
LostPacketRate:<%> OutOfOrderRate:<%
MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
last <buf event time>s dur:<Min>/<Max>s
FR <protocol> [int dcli cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
Tele <int> (callID) [channel_id] tx:<tot>/<faxes> ms <codec> noise:<l> acom:<l> i/o:<l>/<l> dBm
MODEMRELAY info:<rcvd/vsd> <sent> <resent> xid:<rcvd> <sent> total:<rcvd> <sent> <drops>
speeds (bps): local <tx>/ <rx> remote <tx>/ <rx>
Proxy <ip> <(audio udp>,<video udp>,<tcp0>,<tcp1>,<tcp2>,<tcp3> endpt: <type>/<manf>
bw: <req> <act> code: <audio>:<video>
rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
x: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>

Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

0 : 987 361941011ms.1 (16:01:10.557 IST Tue May 14 2013) +530 pid:1 Answer 1005 connected
dur 00:00:56 tx:1082/173120 rx:1141/182560 dscp:0 media:0 audio tos:0x8B video tos:0x0
IP 2001:1111:2222:3333:4444:5555:6666:1012:38356 SRTP: off rtt:0ms pl:0/0/0 lost:0/0/0 delay:0/0/0ms
g711ulaw TextRelay: off Transcoded: No
media inactive detected:n media cntrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00

0 : 988 361941011ms.1 (16:01:10.567 IST Tue May 14 2013) +510 pid:2 Originate 2005 connected
dur 00:00:56 tx:1141/182560 rx:1082/173120 dscp:0 media:0 audio tos:0x8B video tos:0x0
IP 2001:1111:2222:3333:4444:5555:6666:1012:26827 SRTP: off rtt:0ms pl:0/0/0 lost:0/0/0 delay:0/0/0ms
g711ulaw TextRelay: off Transcoded: No
media inactive detected:n media cntrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

-------------------------------------------------------------------------

Step 2  show call active voice compact

Example:

Device# show call active voice compact

<table>
<thead>
<tr>
<th>&lt;callID&gt;</th>
<th>A/O FAX T&lt;sec&gt;</th>
<th>Codec</th>
<th>type</th>
<th>Peer Address</th>
<th>IP R&lt;ip&gt;:&lt;udp&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>987 ANS</td>
<td>T61 g711ulaw</td>
<td>VOIP</td>
<td></td>
<td>P1005 2001:...:1012:38356</td>
<td></td>
</tr>
<tr>
<td>988 ORG</td>
<td>T61 g711ulaw</td>
<td>VOIP</td>
<td></td>
<td>P2005 2001:...:1012:26827</td>
<td></td>
</tr>
</tbody>
</table>

Step 3  show voip rtp connections

Example:

Device# show voip rtp connections

VoIP RTP Port Usage Information:
Max Ports Available: 24273, Ports Reserved: 303, Ports in Use: 2
Port range not configured, Min: 16384, Max: 32767

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Ports Available</th>
<th>Ports Reserved</th>
<th>Ports In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Address-Range</td>
<td>8091 101 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001::</td>
<td>8091 101 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.0.0.0</td>
<td>8091 101 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Found 2 active RTP connections

Verifying and Troubleshooting Cisco UBE ANAT Flow-Through Call

To verify and troubleshoot Cisco UBE ANAT Flow-Through calls, use the following commands:

SUMMARY STEPS

1. debug ccsip message
2. show voip rtp connections

DETAILED STEPS

Step 1  debug ccsip message

Example:

Device# show logging

*Jun 7 09:17:41.135: //1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
INVITE sip:6000@[2001:DB8:C18:2:223:FEAC:4540]:5060 SIP/2.0
Verifying and Troubleshooting Cisco UBE ANAT Flow-Through Call

Cisco Unified Border Element Configuration Guide
226

**Verifying and Troubleshooting Cisco UBE ANAT Flow-Through Call**

Via: SIP/2.0/UDP [2001:DB8:C18:2:219:2FF:FE89:7928]:5060;branch=z9hG4bK1GA8CD
Date: Thu, 07 Jun 2012 10:47:17 GMT
Supported: 100rel,timer,resource-priority,replaces
Require: sdp-anat
Min-SE: 1800
Cisco-Guid: 4231321369-2948862433-216845193-0797538600
User-Agent: Cisco-SIPGateway/IOS-12.x
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1339066037
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 441

v=0
c=IN IP6 2001:DB8:C18:2:219:2FF:FE89:7928
a=setup:active
a=gti:1
a=fmtp:18 annexb=no
a=rtpmap:18 G729/8000
a=rtpmap:19 CN/8000
a=ptime:20
c=IN IP4 9.44.30.10
a=group:ANAT 1 2
a=rtpmap:18 G729/8000
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 16970 RTP/AVP 18 19
m=audio 17066 RTP/AVP 18 19

*Jun 7 09:17:41.159: //31/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 9.44.30.11:5060;branch=z9hG4bK1GA8CD
Date: Thu, 07 Jun 2012 09:17:41 GMT
Timestamp: 1339066037
CSeq: 101 INVITE
Allow-Events: telephone-event
Content: application/sdp
Content-Length: 441

*Jun 7 09:17:41.159: //31/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Sent:
INVITE sip:6000@9.44.30.11:5060 SIP/2.0
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK2688E
Remote-Party-ID: <sip:1001@9.44.30.14>;party=calling;screen=no;privacy=off
From: <sip:1001@9.44.30.14>
To: <sip:6000@9.44.30.11>
Content-Length: 0

*Jun 7 09:17:41.159: //32/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Sent:
INVITE sip:6000@9.44.30.11:5060 SIP/2.0
Via: SIP/2.0/UDP 9.44.30.11:5060;branch=z9hG4bK2688E
Remote-Party-ID: <sip:1001@9.44.30.11>;party=calling;screen=no;privacy=off
From: <sip:1001@9.44.30.11>
To: <sip:6000@9.44.30.11>
Date: Thu, 07 Jun 2012 09:17:41 GMT
Call-ID: 7780227E-ABFB811E1-8060F4DD-5665AA1B@9.44.30.14
Supported: timer, resource-priority, replaces
Require: sdp-anat
Min-SE: 1800
Cisco-Guid: 4231321369-2948862433-2168455193-0797538600
User-Agent: Cisco-SIPGateway/IOS-15.2.20120528.102328.
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Timestamp: 1339060661
Contact: <sip:1001@9.44.30.14:5060>
Expires: 180
Allow-Events: telephone-event
Max-Forwards: 69
Content-Type: application/sdp
Content-Disposition: session;handling=required Phone is offhook
Content-Length: 437
v=0
o=CiscoSystemsSIP-GW-UserAgent 3184 51 IN IP4 9.44.30.14
s=SIP Call
t=0 0
a=group:ANAT 1 2
m=audio 16438 RTP/AVP 18 19
a=mid:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 16440 RTP/AVP 18 19
a=mid:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20

*Jun 7 09:17:41.179: //32/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK2688E
From: <sip:1001@9.44.30.14>;tag=6D0FC0-1428
To: <sip:6000@9.44.30.11>
Date: Thu, 07 Jun 2012 10:40:14 GMT
Call-ID: 7780227E-ABFB811E1-8060F4DD-5665AA1B@9.44.30.14
Timestamp: 1339060661
CSeq: 101 INVITE
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

*Jun 7 09:17:41.203: //32/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK2688E
From: <sip:1001@9.44.30.14>;tag=6D0FC0-1428
To: <sip:6000@9.44.30.11>
Date: Thu, 07 Jun 2012 10:40:14 GMT
Call-ID: 7780227E-ABFB811E1-8060F4DD-5665AA1B@9.44.30.14
Timestamp: 1339060661
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Verifying and Troubleshooting Cisco UBE ANAT Flow-Through Call

Allow-Events: telephone-event
Remote-Party-ID: <sip:6000@9.44.30.11>;party=called;screen=no;privacy=off
Contact: <sip:6000@9.44.30.11:5060>
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

*Jun 7 09:17:41.207: //31/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP [2001:DB8:C18:2:219:2FFF:FE89:7928]:5060;branch=z9hG4bK1CA8CD
Date: Thu, 07 Jun 2012 09:17:41 GMT
Timestamp: 1339066037
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

*Jun 7 09:17:41.219: //32/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK2688E
From: <sip:1001@9.44.30.14>;tag=93D1F9D4-9E2
Date: Thu, 07 Jun 2012 10:40:14 GMT
Call-ID: 7780227E-AFB811E1-8060F4DD-5665AA1B@9.44.30.14
Timestamp: 1339060661
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

v=0
o=CiscoSystemsSIP-GW-UserAgent 8213 2783 IN IP4 9.44.30.11
s=SIP Call
c=IN IP6 2001:DB8:C18:2:217:59FF:FEDE:8898
m=audio 17200 RTP/AVP 18 19
a=group:ANAT 1
m=audio 0 RTP/AVP 18 19
a=mid:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 0 RTP/AVP 18 19
m=audio 0 RTP/AVP 18 19
a=mid:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20
a=rtpmap:19 CN/8000
a=ptime:20

*Jun 7 09:17:41.227: //32/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Sent:
ACK sip:6000@9.44.30.11:5060 SIP/2.0
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK27145B
From: <sip:1001@9.44.30.14>;tag=6D0FC0-1428
To: <sip:6000@9.44.30.11>;tag=93D1F9D4-9E2
Date: Thu, 07 Jun 2012 09:17:41 GMT
Call-ID: 7780227E-AFB811E1-8060F4DD-5665AA1B@9.44.30.14
Max-Forwards: 70
CSeq: 101 ACK
Allow-Events: telephone-event
Content-Length: 0

*Jun 7 09:17:41.235: //31/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP [2001:DB8:C18:2:219:2FFF:FE89:7928]:5060;branch=z9hG4bK1CA8CD
Date: Thu, 07 Jun 2012 09:17:41 GMT
Timestamp: 1339066037
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Supported: replaces
Require: sdp-anat
Server: Cisco-SIPGateway/IOS-15.2.20120528.102328.
Supported: timer
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 433

v=0
u=SIP Call
c=IN IP4 9.44.30.14
t=0 0
a=group:ANAT 1
m=audio 16436 RTP/AVP 18 19
c=IN IP4 9.44.30.14
a=mid:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 0 RTP/AVP 18 19
a=mid:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20

*Jun 7 09:17:41.251: //1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
Verifying Cisco UBE ANAT Flow-Around Calls

To verify Cisco UBE ANAT Flow-Around calls, use the **debug ccsip message** commands:

**SUMMARY STEPS**

1. **debug ccsip message**
2. **show voip rtp connections**

**DETAILED STEPS**

**Step 1**

**debug ccsip message**

**Example:**

```
Device# Show logging

*Jun  7 17:26:30.681: //1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
  Received:
  Via: SIP/2.0/UDP [2001:DB8:C18:2:2:23:33FF:FE89:7928]:5060;branch=z9hG4bK1CB1E77
  Date: Thu, 07 Jun 2012 10:47:17 GMT
  Max-Forwards: 70
  CSeq: 101 ACK
  Allow-Events: telephone-event
  Content-Length: 0
```

**Step 2**

**show voip rtp connections**

**Example:**

```
Device# show voip rtp connections

VoIP RTP Port Usage Information:
Max Ports Available: 8091, Ports Reserved: 101, Ports in Use: 3
Port range not configured, Min: 16384, Max: 32767

<table>
<thead>
<tr>
<th>Media Address Range</th>
<th>Ports Available</th>
<th>Ports Reserved</th>
<th>Ports In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Address Range</td>
<td>8091</td>
<td>101</td>
<td>3</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No. CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31</td>
<td>32</td>
<td>16436</td>
<td>16970</td>
<td>9.44.30.14</td>
</tr>
</tbody>
</table>

Found 2 active RTP connections
```
Date: Thu, 07 Jun 2012 17:35:05 GMT
Supported: 100rel,timer,resource-priority,replaces
Require: sdp-anat
Min-SE: 1800
Cisco-Guid: 1170397766-2953384417-2170945561-0797538600
User-Agent: Cisco-SIPGateway/IOS-12.x
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1339090505
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 465

v=0
s=SIP Call
t=0 0
a=group:ANAT 1 2
m=audio 18706 RTP/AVP 18 0 19
c=IN IP4 9.44.30.13
a=mid:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtmp:\000a66039b58d4000000
m=audio 16384 RTP/AVP 18 0 19
c=IN IP6 2001:DB8:C18:2:223:33FF:FEB1:B440
a=mid:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtmp:\000a66039b58d4000000
m=audio 16384 RTP/AVP 18 0 19
c=IN IP6 2001:DB8:C18:2:223:33FF:FEB1:B440
a=mid:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtmp:\000a66039b58d4000000
m=audio 16384 RTP/AVP 18 0 19

*Jun 7 17:26:30.705: //106/45C2DA668166/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 100 Trying
Date: Thu, 07 Jun 2012 17:26:30 GMT
Timestamp: 1339090505
CSeq: 101 INVITE
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.20120528.102328.
Content-Length: 0

*Jun 7 17:26:30.705: //107/45C2DA668166/SIP/Msg/ccsipDisplayMsg:
Sent:
INVITE sip:6000@9.44.30.11:5060 SIP/2.0
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK90BB
Remote-Party-ID: <sip:1001@9.44.30.14>;party=calling;screen=no;privacy=off
From: <sip:1001@9.44.30.14>;tag=22C984C-970
To: <sip:6000@9.44.30.11>
Date: Thu, 07 Jun 2012 17:26:30 GMT
Call-ID: C145AF07-AFFC11E1-813EF4DD-5665AA1B@9.44.30.14
Supported: timer, resource-priority, replaces
Require: sdp-anat
Min-SE: 1800
Cisco-Guid: 1170397766-2953384417-2170945561-0797538600
User-Agent: Cisco-SIPGateway/IOS-15.2.2.5.T
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Timestamp: 1339089990
Contact: <sip:1001@9.44.30.14:5060>
Expires: 180
Allow-Events: telephone-event
Max-Forwards: 69
Content-Type: application/sdp
Content-Disposition: session; handling=required
Content-Length: 418

v=0
c=IN IP4 9.44.30.13
a=group:ANAT 1 2
m=audio 18706 RTP/AVP 18 19
c=IN IP4 9.44.30.13
a=mid:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 16384 RTP/AVP 18 19
c=IN IP6 2001:DB8:C18:2:223:33FF:FEB1:B440
a=mid:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20

*Jun 7 17:26:30.729: /107/45C2DA468166/SIP/Msg/ccsipDisplayMsg: Received:
SIP/2.0 100 Trying
Via: SIPS/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK90BB
From: <sip:1001@9.44.30.14>;tag=22C984C-970
To: <sip:6000@9.44.30.11>
Date: Thu, 07 Jun 2012 18:49:04 GMT
Call-ID: C145AF07-AFFC11E1-813EF4DD-5665AA1B@9.44.30.14
Timestamp: 1339089990
CSeq: 101 INVITE
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

*Jun 7 17:26:30.753: /107/45C2DA468166/SIP/Msg/ccsipDisplayMsg: Received:
SIP/2.0 180 Ringing
Via: SIPS/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK90BB
From: <sip:1001@9.44.30.14>;tag=22C984C-970
To: <sip:6000@9.44.30.11>
Date: Thu, 07 Jun 2012 18:49:04 GMT
Call-ID: C145AF07-AFFC11E1-813EF4DD-5665AA1B@9.44.30.14
Timestamp: 1339089990

CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Remote-Party-ID: <sip:6000@9.44.30.11>;party=called;screen=no;privacy=off
Contact: <sip:6000@9.44.30.11:5060>
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

*Jun 7 17:26:30.753: //106/45C2DA468166/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 180 Ringing
Date: Thu, 07 Jun 2012 17:26:30 GMT
Timestamp: 1339090505
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.2.20120528.102328.
Content-Length: 0

*Jun 7 17:26:30.765: //107/45C2DA468166/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK90BB
From: <sip:1001@9.44.30.14>;tag=22C984C-970
To: <sip:6000@9.44.30.11>;tag=959183D0-2073
Date: Thu, 07 Jun 2012 18:49:04 GMT
Call-ID: C145AF07-AFFC11E1-813EF4DD-5665AA1B@9.44.30.14
Timestamp: 1339089990
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Remote-Party-ID: <sip:6000@9.44.30.11>;party=called;screen=no;privacy=off
Contact: <sip:6000@9.44.30.11:5060>
Supported: replaces
Require: sdp-anat
Server: Cisco-SIPGateway/IOS-15.2.2.20120528.102328.
Supported: timer
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 412

v=0
o=CiscoSystemsSIP-GW-UserAgent 2764 5975 IN IP4 9.44.30.11
s=SIP Call
c=IN IP4 9.44.30.11
t=0 0
group:ANAT 1
m=audio 17278 RTP/AVP 18 19
c=IN IP4 9.44.30.11
a=mid:1
a=rtpmap:18 G729/8000
a=fmt:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 0 RTP/AVP 18 19
c=IN IP6 2001:DB8:C18:2:217:59FF:FEDE:8898
Verifying Cisco UBE ANAT Flow-Around Calls

Jun 7 17:26:30.777: //107/45C2DA468166/SIP/Msg/ccsipDisplayMsg:
Sent:
ACK sip:60000@9.44.30.11:5060 SIP/2.0
Via: SIP/2.0/UDP 9.44.30.14:5060;branch=z9hG4bK91207D
From: <sip:1001@9.44.30.14>;tag=22C984C-970
To: <sip:60000@9.44.30.11>;tag=959183D0-2073
Date: Thu, 07 Jun 2012 17:26:30 GMT
Call-ID: C145AF07-AFFC11E1-813EF4DD-5665AA1B@9.44.30.14
Max-Forwards: 70
CSeq: 101 ACK
Allow-Events: telephone-event
Content-Length: 0

Jun 7 17:26:30.785: //106/45C2DA468166/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 200 OK
Date: Thu, 07 Jun 2012 17:26:30 GMT
Timestamp: 1339090505
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Contact: <sip:60000@[2001:DB8:C18:2:223:4FF:FEAC:4540]>;party=called;screen=no;privacy=off
Supported: replaces
Require: sdp-anat
Server: Cisco-SIPGateway/IOS-15.2.20120528.102328.
Supported: timer
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 421

v=0
s=SIP Call
c=IN IP4 9.44.30.11
t=0 0
a-group:ANAT 1
m=audio 17278 RTP/AVP 18 19
a=group:ANAT 1
m=audio 0 RTP/AVP 18 19
a=group:ANAT 1

Verifying VMWI SIP

SUMMARY STEPS

1. show sip-ua mwi
2. debug vpmsignal
3. debug ccsip messages

DETAILED STEPS

Step 1 show sip-ua mwi

Example:

Device# show sip-ua mwi
MWI type: 2
MWI server: 2001:10:12:1::2006 //IPv6 MWI Server Address//
MWI expires: 3600
MWI port: 5060
MWI dial peer tag: 0 //Shows the MWI-Server binding dial-peer tag. Tag “0” is default.//
MWI solicited //MWI type is solicited by default. Subscription of voice-port is required in this case only.//
MWI ipaddr cnt 1:
MWI ipaddr idx 0:
MWI server: 2001:10:12:1::2006, port 5060, transport 1 //IPv6 MWI Server Address//
MWI server dns lookup retry cnt: 0
Verifying SDP Passthrough Configuration

SUMMARY STEPS
1. debug ccsip all
2. show voip rtp connection

DETAILED STEPS

Step 1 debug ccsip all

Example:
Device# show logging

Received:
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:33FF:FEAC:4540]:5060;branch=z9hG4bK20277F
Date: Fri, 08 Jun 2012 11:01:48 GMT
Call-ID: 2D6EEC84-B09011E1-8235D9DB-F669887E@2001:DB8:C18:2:223:33FF:FEAC:4540
Supported: 100rel,timer,resource-priority,replaces
Require: sdp-anat
Min-SE: 1800
Cisco-Guid: 2131649325-2962952673-2175336473-0797538600
User-Agent: Cisco-SIPGateway/IOS-12.x
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1339153308
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 488

v=0
s=SIP Call
c=IN IP6 2001:DB8:C18:2:223:33FF:FEB1:B440
t=0
a=group:ANAT 1 2
m=audio 16406 RTP/AVP 18 0 19
c=IN IP6 2001:DB8:C18:2:223:33FF:FEB1:B440
a=mid:1
a=rtpmap:18 G729/8000
a=fmt:18 annexb=no
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
m=audio 18024 RTP/AVP 18 0 19
c=IN IP4 9.44.30.13
a=mid:2
a=rtpmap:18 G729/8000
a=fmt:18 annexb=no
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000

Sent:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:33FF:FEB1:B440]:5060;branch=z9hG4bK20277F
Date: Fri, 08 Jun 2012 10:53:14 GMT
Call-ID: 2D6EEC84-B09011E1-8235D9DB-F669887E@2001:DB8:C18:2:223:33FF:FEB1:B440
Timestamp: 1339153308
CSeq: 101 INVITE
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.20120528.102328.
Content-Length: 0

Sent:
INVITE sip:6000@[2001:DB8:C18:2:223:33FF:FEB1:B440]:5060;branch=z9hG4bK20277F SIP/2.0
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:33FF:FEB1:B440]:5060;branch=z9hG4bK20277F
Date: Fri, 08 Jun 2012 10:53:14 GMT
Call-ID: FB05CC74-B08E11E1-8235D9DB-F669887E@2001:DB8:C18:2:223:33FF:FEB1:B440
Timestamp: 1339153308
CSeq: 101 INVITE
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.20120528.102328.
Content-Length: 0

Min-SE: 1800
Cisco-Guid: 2131649325-2962952673-2175336473-0797538600
User-Agent: Cisco-SIPGateway/IOS-15.2.20120528.102328.
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Timestamp: 1339152794
Allow-Events: telephone-event
Max-Forwards: 69
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Timestamp: 1339152794
Allow-Events: telephone-event
Max-Forwards: 69
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 443

v=0
s=SIP Call
Verifying SDP Passthrough Configuration

*Jun 8 10:53:14.137: //243/7F0E632D81A9/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=z9hG4bK15D1013
Date: Fri, 08 Jun 2012 12:15:49 GMT
Call-ID: FB05CC74-B08E11E1-82C1F4DD-5665AA1B@2001:DB8:C18:2:223:4FF:FEAC:4540
Timestamp: 1339152794
CSeq: 101 INVITE
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=z9hG4bK15D1013
Date: Fri, 08 Jun 2012 12:15:49 GMT
Call-ID: FB05CC74-B08E11E1-82C1F4DD-5665AA1B@2001:DB8:C18:2:223:4FF:FEAC:4540
Timestamp: 1339152794
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=z9hG4bK20277F
Date: Fri, 08 Jun 2012 10:53:14 GMT
Call-ID: 2D6EEC84-B09011E1-8235D9DB-F6698878E02001:DB8:C18:2:223:33FF:FEAC:4540
Timestamp: 1339153308
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.20120528.112328.
Content-Length: 0

Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=z9hG4bK15D1013
Date: Fri, 08 Jun 2012 12:15:49 GMT
Call-ID: F005CC74-B08E11E1-82C1F4DD-5665AA1B@2001:DB8:C18:2:223:4FF:FEAC:4540
Timestamp: 1339152794
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Supported: replaces
Require: sdp-anat
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Supported: timer
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 434

v=0
a=SIP Call
c=IN IP6 2001:DB8:C18:2:217:59FF:FEDE:8898
t=0 0
a=group:ANAT 1
m=audio 17424 RTP/AVP 18 19
c=IN IP6 2001:DB8:C18:2:217:59FF:FEDE:8898
a=mid:1
a=rtpmap:18 G729/8000
a=fmt=18 annexb=no
a=rtpmap:19 CN/8000
m=audio 0 RTP/AVP 18 19
c=IN IP4 9.44.30.11
a=mid:2
a=rtpmap:18 G729/8000
a=fmt=18 annexb=no
a=rtpmap:19 CN/8000

Sent:
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=z9hG4bK20277F
Date: Fri, 08 Jun 2012 10:53:14 GMT
Call-ID: F005CC74-B08E11E1-82C1F4DD-5665AA1B@2001:DB8:C18:2:223:4FF:FEAC:4540
Max-Forwards: 70
CSeq: 101 ACK
Allow-Events: telephone-event
Content-Length: 0

Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=z9hG4bK20277F
Date: Fri, 08 Jun 2012 10:53:14 GMT
Call-ID: 2D6EEC84-B09011E1-8235D9DB-F669887E@2001:DB8:C18:2:223:4FF:FEAC:4540
Timestamp: 1339153308
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
Allow-Events: telephone-event
Verifying SDP Passthrough Configuration

Step 2  show voip rtp connection

Example:

Device# show voip rtp connection

VoIP RTP Port Usage Information:
Max Ports Available: 8091, Ports Reserved: 101, Ports in Use: 2
Port range not configured, Min: 16384, Max: 32767

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Ports Available</th>
<th>Ports Reserved</th>
<th>Ports In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Address-Range</td>
<td>8091</td>
<td>101</td>
<td>2</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:
No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP
Found 2 active RTP connections
Feature Information for VoIP for IPv6

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 32: Feature Information for VoIP for IPv6

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Cisco UBE support for IPv6    | 12.4(22)T | Cisco Unified Border Element (Cisco UBE) support for SIP IPv4-IPv6 dual stack and IPv4 and IPv6 capability provides the following functionality:  
  • Translation of SIP IPv4 to IPv6 addresses  
  • Administration and enforcement of policies for the IPv4/IPv6 mode of operation of each component.  
  • Supports the following scenarios: H.323 IPv4 to SIP IPv6; SIP IPv4 to SIP IPv6, SIP IPv6 to SIP IPv6  
  • DTMF: Interworking capability on Cisco UBE (H.245 Signal, RFC 2833, SIP Notify, Key Press Markup Language, H.323 to SIP, RFC 2833 to G.711 Inband)  
  • IPv6 topology hiding and demarcation  
  • SIP Options-ping  

The VoIP for IPv6 feature describes the Session Border Controller (SBC) functionality of connecting a SIP IPv4 or H.323 IPv4 network to a SIP IPv6 network that is implemented on a Cisco UBE to facilitate migration from VoIPv4 to VoIPv6.
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco UBE support for IPv6</td>
<td>15.3(2)T</td>
<td></td>
</tr>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following features are supported on Cisco UBE for 15.3(2)T:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Assisted RTCP (RTCP Keepalive)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Audio Transcoding using Local Transcoding Interface (LTI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Address Hiding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Call Transfer (re-INVITE, REFER)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Call Forward (302 based)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• IP Toll Fraud</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Hold/Resume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Media Flow-Through (FT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Media Flow-Around (FA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RE-INVITE Consumption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RTP Port Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SDP Pass-Through</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UDP Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Media Anti-Trombone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Header Passing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Refer-To Passing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Error Pass-through</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP UPDATE Interworking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP Session timer (RFC 4028)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP OPTIONS Ping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Configurable Error Response Code in OPTIONS Ping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Limiting the Rate of Incoming SIP Calls per Dial-Peer (aka Call Spike)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP Profiles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP Media Inactivity Detection</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DSCP-Based QoS Support</td>
<td>12.4(22)T</td>
<td>IPv6 supports this feature.</td>
</tr>
<tr>
<td>IPv6 Dual Stack</td>
<td>12.4(22)T</td>
<td>Adds IPv6 capability to existing VoIP features on the Cisco UBE. Additionally, the SBC functionality of connecting SIP IPv4 or H.323 IPv4 network to SIP IPv6 network is implemented on a Cisco UBE to facilitate migration from VoIPv4 to VoIPv6. The following commands were introduced or modified: None</td>
</tr>
<tr>
<td>RTP/RTCP over IPv6</td>
<td>12.4(22)T</td>
<td>RTP stack supports the ability to create IPv6 connections using IPv6 unicast and multicast addresses as well as IPv4 connections.</td>
</tr>
</tbody>
</table>
### Feature Information

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDM-SIP GW for IPv6</td>
<td>12.4(24)T &lt;br&gt;15.3(2)T</td>
<td>IPv6 supports this feature. &lt;br&gt;• Session Initiation Protocol Features Supported on IPv6 &lt;br&gt;• Cisco UBE features Supported on IPv6 &lt;br&gt;• SIP Gateway Generic Features &lt;br&gt;Apart from the SIP Gateway features already supported on IPv4 and IPv6 for 12.4(24)T release, the following features are also supported on IPv6: &lt;br&gt;• SIP VMWI for FXS phones &lt;br&gt;• History-Info &lt;br&gt;• Handling 181/183 Responses with/without SDP &lt;br&gt;• SIP Session Timer (4028) &lt;br&gt;• SIP Media Inactivity Detection &lt;br&gt;• PPI/PAI &amp; Privacy (RFC3323/RFC3325) Headers</td>
</tr>
</tbody>
</table>
 CHAPTER 17

Configurable SIP Parameters via DHCP

The Configurable SIP Parameters via DHCP feature allows a Dynamic Host Configuration Protocol (DHCP) server to provide Session Initiation Protocol (SIP) parameters via a DHCP client. These parameters are used for user registration and call routing.

The DHCP server returns the SIP Parameters via DHCP options 120 and 125. These options are used to specify the SIP user registration and call routing information. The SIP parameters returned are the SIP server address via Option 120, and vendor-specific information such as the pilot, contract or primary number, an additional range of secondary numbers, and the SIP domain name via Option 125.

In the event of changes to the SIP parameter values, this feature also allows a DHCP message called DHCPFORCERENEW to reset or apply a new set of values.

The SIP parameters provisioned by DHCP are stored, so that on reboot they can be reused.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configurable SIP Parameters via DHCP

- A DHCP interface has to be associated with SIP before configurable SIP parameters via DHCP can be enabled.
Cisco Unified Border Element

- Cisco IOS Release 12.4(22)YB or a later release must be installed and running on your Cisco Unified Border Element.

Cisco Unified Border Element (Enterprise)

- Cisco IOS XE Release 3.17S or a later release must be installed and running on your Cisco ASR 1000 Series Router.

Restrictions for Configurable SIP Parameters via DHCP

- DHCP Option 120 is the standard DHCP option (RFC3361) to get a SIP server address, and this can be used by any vendor DHCP server. Only one address is supported, which is in the IPv4 address format. Multiple IPv4 address entries are not supported. Also, there is no support for a DNS name in this or for any port number given behind the IPv4 address.

- DHCP Option 125 (RFC 3925) provides vendor-specific information and its interpretation is associated with the enterprise identity. The primary and secondary phone numbers and domain are obtained using Option 125, which is vendor-specific. As long as other customers use the same format as in the Next Generation Network (NGN) DHCP specification, they can use this feature.

- A primary or contract number is required in suboption 202 of DHCP Option 125. There can be only one instance of the primary number and not multiple instances.

- Multiple secondary or numbers in suboption 203 of DHCP Option 125 are supported. Up to five numbers are accepted and the rest ignored. Also, they have to follow the contract number in the DHCP packet data.

- Authentication is not supported for REGISTER and INVITE messages sent from a Cisco Unified Border Element that uses DHCP provisioning.

- The DHCP provisioning of SIP Parameters is supported only over one DHCP interface.

- The DHCP option is available only to be configured for the primary registrar. It will not be available for a secondary registrar.

Information About Configurable SIP Parameters via DHCP

To perform basic Configurable SIP Parameters via DHCP configuration tasks, you should understand the following concepts:

Cisco Unified Border Element Support for Configurable SIP Parameters via DHCP

The Cisco Unified Border Element provides the support for the DHCP provisioning of the SIP parameters. The NGN is modeled using SIP as a VoIP protocol. In order to connect to NGN, the User to Network Interface (UNI) specification is used. Cisco TelePresence Systems (CTS), consisting of an IP Phone, a codec, and Cisco Unified Communications Manager, are required to internetwork over the NGN for point-to-point and point-to-multipoint video calls. Because Cisco Unified Communications Manager does not provide a UNI
interface, there has to be an entity to provide the UNI interface. The Cisco Unified Border Element provides the UNI interface and has several advantages such as demarcation, delayed offer to early offer, and registration. The figure below shows the Cisco Unified Border Element providing the UNI interface for the NGN.

**Figure 31: Cisco NGN with Cisco Unified Border Element providing UNI interface**

### DHCP to Provision SIP Server, Domain Name, and Phone Number

NGN requires Cisco Unified Border Element to support DHCP (RFC 2131 and RFC 2132) to provision the following:

- IP address for Cisco Unified Border Element’s UNI interface facing NGN
- SIP server address using option 120
- Option 125 vendor specific information to get:
  - Pilot number (also called primary or contract number), there is only one pilot number in DHCPACK, and REGISTER is done only for the pilot number
  - Additional numbers, or secondary numbers, are in DHCPACK; there is no REGISTER for additional numbers
  - SIP domain name
- DHCPFORCERENEW to reset or apply a new set of SIP parameters (RFC 3203)

### DHCP-SIP Call Flow

The following scenario shows the DHCP messages involved in provisioning information such as the IP address for UNI interface, and SIP parameters including the SIP server address, phone number, and domain name, along with how SIP messages use the provisioned information.

The figure below shows the DHCP and SIP messages involved in obtaining the SIP parameters and using them for REGISTER and INVITE.
DHCP Message Details

The DHCP call flow involved in obtaining Cisco Unified Border Element provision information, including the IP address for UNI interface and SIP information such as phone number, domain, and SIP server, is shown in the figure below.
The DHCP messages involved in provisioning the SIP parameters are described in Steps 1 to 6.

1. **F1**: The Cisco Unified Border Element DHCP client sends a DHCPDISCOVER message to find the available NGN DHCP servers on the network and obtain a valid IPv4 address. The Cisco Unified Border Element DHCP client identity (computer name) and MAC address are included in this message.

2. **F2**: The Cisco Unified Border Element DHCP client receives a DHCPOFFER message from each available NGN DHCP server. The DHCPOFFER message includes the offered DHCP server’s IPv4 address, the DHCP client’s MAC address, and other configuration parameters.

3. **F3**: The Cisco Unified Border Element DHCP client selects an NGN DHCP server and its IPv4 address configuration from the DHCPOFFER messages it receives, and sends a DHCPREQUEST message requesting its usage. Note that this is where Cisco Unified Border Element requests SIP server information via DHCP Option 120 and vendor-identifying information via DHCP Option 125.

4. **F4**: The chosen NGN DHCP server assigns its IPv4 address configuration to the Cisco Unified Border Element DHCP client by sending a DHCPACK message to it. The Cisco Unified Border Element DHCP client receives the DHCPACK message. This is where the SIP server address, phone number and domain name information are received via DHCP options 120 and 125. The Cisco Unified Border Element will use the information for registering the phone number and routing INVITE messages to the given SIP server.
5. F5: When NGN has a change of information or additional information (such as changing SIP server address from 1.1.1.1 to 2.2.2.2) for assigning to Cisco Unified Border Element, the DHCP server initiates DHCPFORSERENEW to the Cisco Unified Border Element. If the authentication is successful, the Cisco Unified Border Element DHCP client accepts the DHCPFORSERENEW and moves to the next stage of sending DHCPREQUEST. Otherwise DHCPFORSERENEW is ignored and the current information is retained and used.

6. F6 and F7: In response to DHCPFORSERENEW, similar to steps F3 and F4, the Cisco Unified Border Element requests DHCP Options 120 and 125. Upon getting the response, SIP will apply these parameters if they are different by sending an UN-REGISTER message for the previous phone number and a REGISTER message for the new number. Similarly, a new domain and SIP server address will be used. If the returned information is the same as the current set, it is ignored and hence registration and call routing remains the same.

How to Configure SIP Parameters via DHCP

Configuring the DHCP Client

To receive the SIP configuration parameters the Cisco Unified Border Element has to act as a DHCP client. This is because in the NGN network, a DHCP server pushes the configuration to a DHCP client. Thus the Cisco Unified Border Element must be configured as a DHCP client.

Perform this task to configure the DHCP client.

Before you begin

You must configure the ip dhcp client commands before entering the ip address dhcp command on an interface to ensure that the DHCPDISCOVER messages that are generated contain the correct option values. The ip dhcp client commands are checked only when an IP address is acquired from DHCP. If any of the ip dhcp client commands are entered after an IP address has been acquired from DHCP, the DHCPDISCOVER messages’ correct options will not be present or take effect until the next time the router acquires an IP address from DHCP. This means that the new configuration will only take effect after either the ip address dhcp command or the release dhcp and renew dhcp EXEC commands have been configured.

SUMMARY STEPS

1. enable
2. configure terminal
3. interface type number
4. ip dhcp client request sip-server-address
5. ip dhcp client request vendor-identifying-specific
6. ip address dhcp
7. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>

**Step 2**

configure terminal

**Example:**

Router# configure terminal

Enables global configuration mode.

**Step 3**

interface type number

**Example:**

Router(config)# interface gigabitethernet 0/0

Configures an interface type and enters interface configuration mode.

**Step 4**

ip dhcp client request sip-server-address

**Example:**

Router(config-if)# ip dhcp client request sip-server-address

Configures the DHCP client to request a SIP server address from a DHCP server.

**Step 5**

ip dhcp client request vendor-identifying-specific

**Example:**

Router(config-if)# ip dhcp client request vendor-identifying-specific

Configures the DHCP client to request vendor-specific information from a DHCP server.

**Step 6**

ip address dhcp

**Example:**

Router(config-if)# ip address dhcp

Acquires an IP address on the interface from the DHCP.

**Step 7**

exit

**Example:**

Router(config-if)# exit

Exits the current mode.

### Configuring the DHCP Client Example

The following is an example of how to enable the DHCP client:

```plaintext
Router> enable
Router# configure terminal
Router(config)# interface gigabitethernet 0/1
Router(config-if)# ip dhcp client request sip-server-address
Router(config-if)# ip dhcp client request vendor-identifying-specific
Router(config-if)# ip address dhcp
Router(config-if)# exit
```
Enabling the SIP Configuration

Enabling the SIP configuration allows the Cisco Unified Border Element to use the SIP parameters received via DHCP for user registration and call routing. Perform this task to enable the SIP configuration.

Before you begin

The `dhcp interface` command has to be entered to declare the interface before the `registrar` and `credential` commands are entered.

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `interface type number`
4. `sip-ua`
5. `dhcp interface type number`
6. `registrar dhcp expires seconds random-contact refresh-ratio seconds`
7. `credentials dhcp password [0| 7] password realm domain-name`
8. `exit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> interface type number</td>
<td>Configures an interface type and enters interface configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config)# interface gigabitethernet 0/0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip-ua</td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-if)# sip-ua</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> dhcp interface type number</td>
<td>Assigns a specific interface for DHCP provisioning of SIP parameters.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Router(sip-ua)# dhcp interface gigabitethernet 0/0</td>
<td>• Multiple interfaces on the CUBE can be configured with DHCP--this command specifies the DHCP interface used with SIP.</td>
</tr>
</tbody>
</table>

**Step 6**

```
registrar dhcp expires seconds random-contact refresh-ratio seconds
```

**Example:**

```
Router(sip-ua)# registrar dhcp expires 100 random-contact refresh-ratio 90
```

**Step 7**

```
credentials dhcp password [0-7] password realm domain-name
```

**Example:**

```
Router(sip-ua)# credentials dhcp password cisco realm cisco.com
```

**Step 8**

```
exit
```

**Example:**

```
Router(sip-ua)# exit
```

---

**Enabling the SIP Configuration Example**

The following is an example of how to enable the SIP configuration:

```
Router> enable
Router# configure terminal
Router(config)# interface gigabitethernet 1/0
Router(config-if)# sip-ua
Router(sip-ua)# dhcp interface gigabitethernet 1/0
Router(sip-ua)# registrar dhcp expires 90 random-contact refresh-ratio 90
Router(sip-ua)# credentials dhcp password cisco realm cisco.com
Router(sip-ua)# exit
```

---

**Troubleshooting Tips**

To display information on DHCP and SIP interaction when SIP parameters are provisioned by DHCP, use the `debug ccsip dhcp` command in privileged EXEC mode.
Configuring a SIP Outbound Proxy Server

An outbound-proxy configuration sets the Layer 3 address (IP address) for any outbound REGISTER and INVITE SIP messages. The SIP server can be configured as an outbound proxy server in voice service SIP configuration mode or dial peer configuration mode. When enabled in voice service SIP configuration mode, all the REGISTER and INVITE messages are forwarded to the configured outbound proxy server. When enabled in dial-peer configuration mode, only the messages hitting the defined dial-peer will be forwarded to the configured outbound proxy server.

The configuration tasks in each mode are presented in the following sections:

Perform either of these tasks to configure the SIP server as a SIP outbound proxy server.

Configuring a SIP Outbound Proxy Server in Voice Service VoIP Configuration Mode

Perform this task to configure the SIP server as a SIP outbound proxy server in voice service SIP configuration mode.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. outbound-proxy dhcp
6. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service VoIP configuration mode and specifies VoIP as the voice-encapsulation type.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Router(config-voi-srv)# sip</td>
<td></td>
</tr>
</tbody>
</table>

**Step 5**  
outbound-proxy dhcp  
Example:  
Router(conf-serv-sip)# outbound-proxy dhcp

**Step 6**  
exit  
Example:  
Router(config-serv-sip)# exit

### Configuring a SIP Outbound Proxy Server in Voice Service VoIP Configuration Mode Example

The following is an example of how to configure a SIP outbound proxy in voice service SIP configuration mode:

```
Router> enable  
Router# configure terminal  
Router(config)# voice service voip  
Router(config-voi-srv)# sip  
Router(config-serv-sip)# outbound-proxy dhcp  
Router(config-serv-if)# exit
```

### Configuring a SIP Outbound Proxy Server and Session Target in Dial Peer Configuration Mode

Perform this task to configure the SIP server as a SIP outbound proxy server in dial peer configuration mode.

**Note**

SIP must be configured on the dial pier before DHCP is configured. Therefore the `session protocol sipv2` command must be executed before the `session target dhcp` command. DHCP is supported only with SIP configured on the dial peer.

### SUMMARY STEPS

1. enable  
2. configure terminal  
3. dial-peer voice number voip  
4. session protocol sipv2  
5. voice-class sip outbound-proxy dhcp
6. session target dhcp
7. exit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** | enable  
Example: Router> enable | Enables privileged EXEC mode.  
- Enter your password if prompted. |
| **Step 2** | configure terminal  
Example: Router# configure terminal | Enters global configuration mode. |
| **Step 3** | dial-peer voice number voip  
Example: Router(config)# dial-peer voice 10 voip | Defines a dial peer, specifies VoIP as the method of voice encapsulation, and enters dial peer configuration mode. |
| **Step 4** | session protocol sipv2  
Example: Router(config-dial-peer)# session protocol sipv2 | Enters the session protocol type as SIP. |
| **Step 5** | voice-class sip outbound-proxy dhcp  
Example: Router(config-dial-peer)# voice-class sip outbound-proxy dhcp | Configures the SIP server received from the DHCP server as a SIP outbound proxy server. |
| **Step 6** | session target dhcp  
Example: Router(config-dial-peer)# session target dhcp | Specifies that the DHCP protocol is used to determine the IP address of the session target. |
| **Step 7** | exit  
Example: Router(config-dial-peer)# exit | Exits the current mode. |

**Configuring a SIP Outbound Proxy Server in Dial Peer Configuration Mode Example**

The following is an example of how to configure a SIP outbound proxy in dial peer configuration mode:
Feature Information for Configurable SIP Parameters via DHCP

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 33: Feature Information for Configurable SIP Parameters via DHCP

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable SIP Parameters via DHCP</td>
<td>12.4(22)YB</td>
<td>The Configurable SIP Parameters via DHCP feature introduces the configuring of SIP parameters via DHCP.</td>
</tr>
<tr>
<td></td>
<td>15.0(1)M</td>
<td>The following commands were introduced or modified: credentials (sip-ua), debug ccsip dhcp, dhcp interface, ip dhcp-client forcere new, outbound-proxy, registrar, session target (VoIP dial peer), show sip dhcp, voice-class sip outbound-proxy.</td>
</tr>
</tbody>
</table>

Feature History Table for the ASR.

Table 34: Feature Information for Configurable SIP Parameters via DHCP

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable SIP Parameters via DHCP</td>
<td>IOS XE Release 3.17S</td>
<td>The Configurable SIP Parameters via DHCP feature introduces the configuring of SIP parameters via DHCP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following commands were introduced or modified: credentials (sip-ua), debug ccsip dhcp, dhcp interface, ip dhcp-client forcere new, outbound-proxy, registrar, session target (VoIP dial peer), show sip dhcp, voice-class sip outbound-proxy.</td>
</tr>
</tbody>
</table>
PART II

Dial Peer Enhancements

- Matching Inbound Dial Peers by URI, on page 263
- URI-Based Dialing Enhancements, on page 269
- Multiple Pattern Support on a Voice Dial Peer, on page 283
- Outbound Dial-Peer Group as an Inbound Dial-Peer Destination, on page 289
- Inbound Leg Headers for Outbound Dial-Peer Matching, on page 299
- Server Groups in Outbound Dial Peers, on page 309
- Domain-Based Routing Support on the Cisco UBE, on page 317
- ENUM Enhancement per Kaplan Draft RFC, on page 325
Matching Inbound Dial Peers by URI

The Matching Inbound Dial Peers by URI feature allows you to configure the selection of inbound dial peers by matching parts of the URI sent by a remote (neighboring) SIP entity. The match can be done on different parts of the URI like hostname, IP address, DNS name. This feature can be used to configure configuration policies, enforce specific call-treatment, security, and routing policies on each SIP trunk by originating SIP entity.

In a scenario where multiple SIP hops are involved in a call, there would be multiple via headers involved, and the topmost via header of an incoming SIP invite represents the last hop that forwarded the SIP request, and the bottom-most via header would represent the originator of the SIP request. This feature supports matching by the last hop that forwarded the request (neighboring SIP entity), which is the topmost via header.

For incoming dial-peer match based on URI, if there are multiple dial-peer matches, then the longest matching dial-peer is chosen (similar to multiple dial-peer match based on incoming called number). However for URI pattern match, there is no match length and hence this is the least preferred.

Feature Information for Matching Inbound Dial Peers by URI

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Configuring an Inbound Dial Peer to Match on URI

SUMMARY STEPS

1. enable
2. configure terminal
3. voice class uri voice-class-uri-tag
4. Specify a URI field for the voice class:
   - host hostname-pattern
   - host ipv4: ipv4-address
   - host ipv6: ipv6-address
   - host dns: dns-address
   - pattern uri-pattern
   - user-id username-pattern
5. exit
6. dial-peer voice tag voip
7. session protocol sipv2
8. incoming uri { from | request | to | via} voice-class-uri-tag
9. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>voice class uri voice-class-uri-tag</td>
<td>Creates a voice class for matching SIP dial peers and enters voice URI class configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice class uri 200</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>Specify a URI field for the voice class:</td>
<td>• You can specify up to ten instances of the <strong>host ipv4:</strong>; <strong>host ipv6:</strong>; and <strong>host dns:</strong> commands.</td>
</tr>
<tr>
<td>• host hostname-pattern</td>
<td>• You can specify only one instance of the <strong>host hostname-pattern</strong> commands.</td>
</tr>
<tr>
<td>• host ipv4: ipv4-address</td>
<td>• Length of <strong>uri-pattern</strong>, <strong>username-pattern</strong>, and <strong>hostname-pattern</strong> should be less than 32.</td>
</tr>
<tr>
<td>• host ipv6: ipv6-address</td>
<td>• <strong>username-pattern</strong> is matched against the username field of the URI.</td>
</tr>
<tr>
<td>• host dns: dns-address</td>
<td>• <strong>hostname-pattern</strong> is matched against the host field of the URI.</td>
</tr>
<tr>
<td>• pattern uri-pattern</td>
<td>• <strong>uri-pattern</strong> is matched against the entire URI.</td>
</tr>
<tr>
<td>• user-id username-pattern</td>
<td>• Only one instance of the <strong>pattern</strong> and <strong>host</strong> commands are possible.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voice-uri-class)# host server1</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voice-uri-class)# host ipv4:10.0.0.0</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voice-uri-class)# host dns:xxx.yyy.com</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voice-uri-class)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>dial-peer voice tag voip</td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 6000 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td>session protocol sipv2</td>
<td>Configures SIP as the session protocol type.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

Configure the voice class with an inbound dial peer, so that it matches against configured URI fields.

### Command or Action

**Step 8**

`incoming uri { from | request | to | via } voice-class-uri-tag`

**Example:**

Device(config-dial-peer)# incoming uri via 200

**Step 9**

`end`

**Example:**

Device(config-dial-peer)# end

### Examples for Configuring an Inbound Dial Peer to Match on a URI

#### Matching Against IPv4 Address and VIA

CUBE is configured to use incoming dial-peer 101 for incoming SIP calls from remote SIP endpoint having an IP address of 10.10.10.1

```plaintext
voice class uri 201 sip
host ipv4:10.10.10.1
dial-peer voice 101 voip
    session protocol sipv2
    incoming uri via 201
```

Incoming INVITE that can be matched against this dial peer.

```plaintext
INVITE sip:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP 10.10.10.1:5093;branch=z9hG4bK-17716-1-0
Via: SIP/2.0/TCP 10.10.14.20:5093;branch=z9hG4bK-28280-1-0
```

#### Matching Against DNS Name and VIA

CUBE is configured to use incoming dial-peer 102 for incoming SIP calls from sample.com or an IP address that represents one of the resolved IP address of sample.com.

```plaintext
voice class uri 202 sip
host dns:sample.com
dial-peer voice 101 voip
    session protocol sipv2
    incoming uri via 202
```

Incoming INVITE that can be matched against this dial peer.

```plaintext
INVITE sip:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP sample.com;branch=z9hG4bK-17716-1-0
```

10.10.10.25 is a resolved IP address of sample.com.
Matching Against Multiple Attributes and VIA

CUBE is configured to use incoming dial-peer 103 for incoming SIP calls from xxx.yyy.com, abc.def.com and IP addresses 10.10.10.10, 10.9.10.11 and 10.10.10.10.

```plaintext
voice class uri 203 sip
  host dns:xxx.yyy.com
  host dns:abc.def.com
  host ipv4:10.10.10.10
  host ipv4:10.9.10.11
  host ipv4:10.10.10.10

dial-peer voice 103 voip
  session protocol sipv2
  incoming uri via 203
```

Incoming INVITE that can be matched against this dial peer.

```plaintext
INVITE sip:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP 10.10.10.10:5093;branch=z9hG4bK-17716-1-0
Via: SIP/2.0/TCP 10.10.14.20:5093;branch=z9hG4bK-28280-1-0
10.10.10.25 is a resolved IP address of sample.com.
```
CHAPTER 19

URI-Based Dialing Enhancements

The URI-Based Dialing Enhancements feature describes the enhancements made to Uniform Resource Identifier (URI)-based dialing on Cisco Unified Border Element (CUBE) for Session Initiation Protocol (SIP) calls. The URI-Based Dialing Enhancements feature includes support for call routing on Cisco UBE when the user part of the incoming Request-URI is non-E164 (for example, INVITE sip:user@abc.com).

- Finding Feature Information, on page 269
- Feature Information for URI-Based Dialing Enhancements, on page 269
- Information About URI-Based Dialing Enhancements, on page 270
- How to Configure URI-Based Dialing Enhancements, on page 273
- Configuration Examples for URI-Based Dialing Enhancements, on page 280
- Additional References for URI-Based Dialing Enhancements, on page 282

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Feature Information for URI-Based Dialing Enhancements

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 35: Feature Information for URI-Based Dialing Enhancements

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| URI-Based Dialing Enhancements        |           | The URI-Based Dialing Enhancements feature includes support for call routing on Cisco UBE when the user-part of the incoming Request-URI is non-E164 (for example, INVITE sip:user@abc.com).
|                                       |           | The following commands were introduced or modified: contact-passing, requiri-passing, session target sip-uri and voice-class sip requiri-passing |

Information About URI-Based Dialing Enhancements

Cisco Unified Communications Manager (CUCM) supports dialing using directory Uniform Resource Identifiers (URIs) for call addressing. Directory URIs follow the username@host format where the host portion is an IPv4 address or a fully qualified domain name. A directory URI is a string of characters that can be used to identify a directory number. If that directory number is assigned to a phone, CUCM can route calls to that phone using the directory URI. URI dialing is available for Session Initiation Protocol (SIP) and Signaling Connection Control Part (SCCP) endpoints that support directory URIs.

The primary use of URI-based dialing is peer-to-peer calling between enterprises using complete URI addresses (that is, ‘username@host’). The host part of the URI identifies the destination to which the call should be routed. In earlier Cisco Unified Border Element (Cisco UBE) URI routing, the URI was replaced in the SIP header with the destination server IP address. Then routing of calls was based on the following restrictions:

- The user part of the incoming Request-URI must be an E164 number.
- The outgoing Request-URI is always set to the session target information of the outbound dial peer.

The URI-Based Dialing Enhancements feature extends support for Cisco UBE URI-based routing of calls. With these enhancements Cisco UBE supports:

- URI-based routing when the user part of the incoming Request-URI is non-E164 (for example, INVITE sip:user@abc.com).
- URI-based routing when the user part is not present. The user part is an optional parameter in the URI (for example, INVITE sip:abc.com).
- Copying the outgoing Request-URI and To header from the inbound Request-URI and To header respectively.
- Deriving (optionally) the session target for the outbound dial peer from the host portion of the inbound URI.
- URI-based routing for 302, Refer, and Bye Also scenarios.
- Call hunting where the subsequent dial peer is selected based on URI.
- Pass through of 302, with the host part of Contact: unmodified.
Call Flows for URI-Based Dialing Enhancements

Case 1: URI dialing with username being E164 or non-E164 number and Request-URI host copied from the inbound leg.

Case 2: Incoming Request-URI does not contain user part. The To: header information is also copied from the peer leg when the `requri-passing` command is enabled.

Case 3: The old behavior of setting the outbound Request-URI to session target is retained when the `requri-passing` command is not enabled.

Case 4: The session target derived from the host part of the URI. The outgoing INVITE is sent to resolved IP address of the host part of the URI.
Case 5: Pass through of contact URI to request URI.

Case 6: In 302 pass-through, contact header can be passed through from one leg to another by using the contact-passing command.

Case 7: Pass through of refer-to URI to request URI.
How to Configure URI-Based Dialing Enhancements

Configuring Pass Through of SIP URI Headers

Perform these tasks to configure the pass through of the host part of the Request-Uniform Resource Identifier (URI) and To Session Initiation Protocol (SIP) headers. By default, Cisco Unified Border Element (Cisco UBE) sets the host part of the URI to the value configured under the session target of the outbound dial peer. For more information, see Case 1 in the "Call Flows for URI-based Dialing Enhancements" section.

Configuring Pass Though of Request URI and To Header URI (Global Level)

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. requiri-passing
6. end
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  - Enter your password if prompted.  |
| **Step 2** configure terminal | Enters global configuration mode. |
| **Step 3** voice service voip | Specifies VoIP encapsulation and enters voice service configuration mode. |
| **Step 4** sip | Enters the Session Initiation Protocol (SIP) configuration mode. |
| **Step 5** requiri-passing | Enables pass through of the host part of the Request-URI and To SIP headers. By default, Cisco UBE sets the host part of the URI to the value configured under the session target of the outbound dial peer. |
| **Step 6** end | Ends the current configuration session and returns to privileged EXEC mode. |

### Configuring Pass Through of Request URI and To Header URI (Dial Peer Level)

### SUMMARY STEPS

1. enable
2. configure terminal
3. voice class uri tag sip
4. host hostname-pattern
5. exit
6. dial-peer voice tag voip
7. session protocol sipv2
8. destination uri tag
9. session target ipv4:ip-address
10. voice-class sip requiri-passing [system]
11. end
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | enable | Enables privileged EXEC mode.  
- Enter your password if prompted. |
| Step 2 | configure terminal | Enters global configuration mode. |
| Step 3 | voice class uri tag sip | Creates a voice class for matching dial peers to a Session Initiation Protocol (SIP) and enters voice URI class configuration mode. |
| Step 4 | host hostname-pattern | Matches a call based on the host field in a SIP Uniform Resource Identifier (URI). |
| Step 5 | exit | Exits voice URI class configuration mode. |
| Step 6 | dial-peer voice tag voip | Defines a VoIP dial peer and enters dial peer configuration mode. |
| Step 7 | session protocol sipv2 | Specifies a session protocol for calls between local and remote routers using the Internet Engineering Task Force (IETF) SIP. |
| Step 8 | destination uri tag | Specifies the voice class used to match a dial peer to the destination URI of an outgoing call. |
| Step 9 | session target ipv4:ip-address | Designates a network-specific address to receive calls from a VoIP. |
| Step 10 | voice-class sip requiri-passing [system] | Enables the pass through of SIP URI headers. |
**Command or Action** | **Purpose**
--- | ---
**Step 11** | **end**
*Example:*  
Device(config-dial-peer)# end

Ends the current configuration session and returns to privileged EXEC mode.

---

**Configuring Pass Through of 302 Contact Header**

**Configuring Pass Through of 302 Contact Header (Global Level)**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. sip
5. contact-passing
6. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
</tr>
</tbody>
</table>
| *Example:*  
  Device> enable                  | Enables privileged EXEC mode.                                           |
|                                 | • Enter your password if prompted.                                     |
| **Step 2**                      | configure terminal                                                      |
| *Example:*  
  Device# configure terminal     | Enters global configuration mode.                                       |
| **Step 3**                      | voice service voip                                                      |
| *Example:*  
  Device(config)# voice service voip | Specifies VoIP encapsulation and enters voice service configuration mode. |
| **Step 4**                      | sip                                                                      |
| *Example:*  
  Device(conf-voi-serv)# sip   | Enters voice service SIP configuration mode.                            |
| **Step 5**                      | contact-passing                                                         |
| *Example:*  
  Router(config-serv-sip)# contact-passing | Enables pass through of the contact header from one leg to the other leg in 302 pass through scenario. |
| **Step 6**                      | end                                                                     |
| *Example:*  
  Router(config-serv-sip)# end | Ends the current configuration session and returns to privileged EXEC mode. |
Configuring Pass Through of 302 Contact Header (Dial Peer Level)

SUMMARY STEPS

1. enable
2. configure terminal
3. voice class uri destination-tag sip
4. user-id id-tag
5. exit
6. voice service voip
7. allow-connections sip to sip
8. dial-peer voice tag voip
9. session protocol sipv2
10. destination uri destination-tag
11. voice-class sip contact-passing
12. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice class uri destination-tag sip</td>
<td>Creates a voice class for matching dial peers to a Session Initiation Protocol (SIP) and enters voice URI class configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice class uri mydesturi sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> user-id id-tag</td>
<td>Matches a call based on the User ID portion of the Uniform Resource Identifier (URI).</td>
</tr>
<tr>
<td>Example: Device(config-voice-uri-class)# user-id 5678</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits voice URI class configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config-voice-uri-class)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> voice service voip</td>
<td>Specifies Voice over IP (VoIP) as the voice encapsulation type and enters voice service configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> allow-connections sip to sip</td>
<td>Allows connections between SIP endpoints in a VoIP network.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Device(config-voi-serv)# allow-connections sip to sip</td>
<td>Defines a VoIP dial peer and enters dial peer configuration mode.</td>
</tr>
<tr>
<td><strong>Step 8</strong> dial-peer voice tag voip</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 200 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td>Specifies a session protocol for calls between local and remote routers using the Internet Engineering Task Force (IETF) SIP.</td>
</tr>
<tr>
<td><strong>Step 10</strong> destination uri destination-tag</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# destination uri mydesturi</td>
<td>Specifies the voice class used to match a dial peer to the destination URI of an outgoing call.</td>
</tr>
<tr>
<td><strong>Step 11</strong> voice-class sip contact-passing</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip contact-passing</td>
<td>Enables pass through of the contact header from one leg to the other leg in 302 pass through scenario.</td>
</tr>
<tr>
<td><strong>Step 12</strong> end</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td>Ends the current configuration session and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

**Deriving of Session Target from URI**

Perform this task to derive the session target from the host part of the Uniform Resource Identifier (URI). The outgoing INVITE is sent to the resolved IP address of the host part of the URI. For more information, see Case 4 in the "Call Flows for URI-Based Dialing Enhancements" section.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice class uri destination-tag sip
4. host hostname-pattern
5. exit
6. dial-peer voice tag voip
7. session protocol sipv2
8. destination uri destination-tag
9. session target sip-uri
10. exit
11. voice class uri source-tag sip
12. host hostname-pattern
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | enable | Enables privileged EXEC mode.  
  Example:  
  Device> enable |
| Step 2 | configure terminal | Enters global configuration mode.  
  Example:  
  Device# configure terminal |
| Step 3 | voice class uri *destination-tag* sip | Creates or modifies a voice class for matching dial peers to a Session Initiation Protocol (SIP) or telephone (TEL) Uniform Resource Identifier (URI) and enters voice URI class configuration mode.  
  Example:  
  Device(config)# voice class uri mydesturi sip |
| Step 4 | host *hostname-pattern* | Matches a call based on the host field in a SIP URI.  
  Example:  
  Device(config-voice-uri-class)# host destination.com |
| Step 5 | exit | Exits voice URI class configuration mode.  
  Example:  
  Device(config-voice-uri-class)# exit |
| Step 6 | dial-peer voice *tag* voip | Defines a VoIP dial peer and enters dial peer configuration mode.  
  Example:  
  Device(config)# dial-peer voice 25 voip |
| Step 7 | session protocol sipv2 | Specifies a session protocol for calls between local and remote routers using the Internet Engineering Task Force (IETF) SIP.  
  Example:  
  Device(config-dial-peer)# session protocol sipv2 |
| Step 8 | destination uri *destination-tag* | Specifies the voice class used to match a dial peer to the destination URI of an outgoing call.  
  Example:  
  Device(config-dial-peer)# destination uri mydesturi |
| Step 9 | session target sip-uri | Derives session target from incoming URI.  
  Example:  
  Device(config-dial-peer)# session target sip-uri |
| Step 10 | exit | Exits dial peer voice configuration mode.  
  Example:  
  Device(config-dial-peer)# exit |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 11</strong>&lt;br&gt;voice class uri source-tag sip&lt;br&gt;Example: Device(config)# voice class uri mysourceuri sip</td>
<td>Creates or modifies a voice class for matching dial peers to a SIP or TEL URI and enters voice URI class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 12</strong>&lt;br&gt;host hostname-pattern&lt;br&gt;Example: Device(config-voice-uri-class)# host abc.com</td>
<td>Matches a call based on the host field in a SIP URI.</td>
</tr>
<tr>
<td><strong>Step 13</strong>&lt;br&gt;end&lt;br&gt;Example: Device(config-voice-uri-class)# end</td>
<td>Ends the current configuration session and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

### Configuration Examples for URI-Based Dialing Enhancements

#### Example: Configuring Pass Though of Request URI and To Header URI

#### Example: Configuring Pass Though of Request URI and To Header URI (Global Level)

```bash
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# requri-passing
Device(conf-serv-sip)# end
```

#### Example: Configuring Pass Though of Request URI and To Header URI (Dial Peer Level)

```bash
! Configuring URI voice class destination
Device(config)# voice class uri mydesturi sip
Device(config-voice-uri-class)# host xyz.com
Device(config-voice-uri-class)# exit

! Configuring outbound dial peer
Device(config)# dial-peer voice 13 voip
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# destination uri mydesturi
Device(config-dial-peer)# session target ipv4:10.1.1.1
Device(config-dial-peer)# voice-class sip requri-passing system
Device(config-dial-peer)# end
```
Example: Configuring Pass Through of 302 Contact Header

Example: Configuring Pass Through of 302 Contact Header (Global Level)

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(config-voip)# sip
Device(config-voip)# contact-passing
Device(config-voip)# end
```

Example: Configuring Pass Through of 302 Contact Header (Dial Peer Level)

```
! Configuring URI voice class destination
Device> enable
Device# configure terminal
Device(config)# voice class uri mydesturi sip
Device(config-voice-uri-class)# user-id 5678
Device(config-voice-uri-class)# exit

! Configuring outbound dial peer
Device(config)# voice service voip
Device(config-voip)# allow-connections sip to sip
Device(config-voip)# dial-peer voice 200 voip
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# destination uri mydesturi
Device(config-dial-peer)# voice-class sip contact-passing
Device(config-dial-peer)# end
```

Example: Deriving Session Target from URI

```
Device> enable
Device# configure terminal
Device(config)# voice class uri mydesturi sip
Device(config-voice-uri-class)# host destination.com
Device(config-voice-uri-class)# exit

Device(config)# dial-peer voice 25 voip
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# destination uri mydesturi
Device(config-dial-peer)# session target sip-uri
Device(config-dial-peer)# exit

Device(config)# voice class uri mysouceuri sip
Device(config-voice-uri-class)# host abc.com
Device(config-voice-uri-class)# end
```
Additional References for URI-Based Dialing Enhancements

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice commands</td>
<td>Cisco IOS Voice Command Reference</td>
</tr>
<tr>
<td>Cisco IOS commands</td>
<td>Cisco IOS Master Command List, All Releases</td>
</tr>
<tr>
<td>SIP configuration tasks</td>
<td>SIP Configuration Guide, Cisco IOS Release 15M&amp;T</td>
</tr>
</tbody>
</table>

Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
<tr>
<td>To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds.</td>
<td></td>
</tr>
<tr>
<td>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td></td>
</tr>
</tbody>
</table>
Multiple Pattern Support on a Voice Dial Peer

The Multiple Pattern Support on a Voice Dial Peer feature enables you to configure multiple patterns on a VoIP dial peer using an E.164 pattern map. A dial peer can be configured to match multiple patterns to an incoming calling or called number or an outgoing destination number.

- Feature Information for Multiple Pattern Support on a Voice Dial Peer, on page 283
- Restrictions for Multiple Pattern Support on a Voice Dial Peer, on page 284
- Information About Multiple Pattern Support on a Voice Dial Peer, on page 284
- Configuring Multiple Pattern Support on a Voice Dial Peer, on page 284
- Verifying Multiple Pattern Support on a Voice Dial Peer, on page 286
- Configuration Examples for Multiple Pattern Support on a Voice Dial Peer, on page 288

Feature Information for Multiple Pattern Support on a Voice Dial Peer

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring Multiple Pattern Support on a Voice Dial Peer (Inbound Calls)</td>
<td>Cisco IOS 15.4 (1)T</td>
<td>This feature was extended for inbound VoIP dial peers for incoming calling and called numbers. The following commands were introduced or modified: <strong>incoming called e164-pattern-map,</strong> <strong>incoming calling e164-pattern-map</strong></td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.11S</td>
<td></td>
</tr>
</tbody>
</table>
## Restrictons for Multiple Pattern Support on a Voice Dial Peer

- This feature is supported only on a VoIP dial peer.
- Duplicate patterns cannot be added to a pattern map.

## Information About Multiple Pattern Support on a Voice Dial Peer

Matching an incoming or outgoing call using a pattern defined in a VoIP dial peer is an existing feature on the Cisco Unified Border Element (Enterprise) and Session Initiation Protocol (SIP) Gateway. You can now support multiple patterns on a VoIP dial peer using an E.164 pattern map. You can create a E.164 pattern map and then link it to one or more VoIP dial peers.

When a pattern is the only source to enable a dial peer, a valid E.164 pattern map enables the linked dial peers, whereas an invalid E.164 pattern map disables the linked dial peers. Additionally, whenever an E.164 pattern map is created or reloaded, one or more dial peers linked with an E.164 pattern map is enabled or disabled based on the validation of a pattern map.

You can match a pattern map to an incoming calling or called number or an outgoing destination number.

When a dial peer has multiple patterns, the pattern with the longest prefix is considered as the matching criteria.

## Configuring Multiple Pattern Support on a Voice Dial Peer

### SUMMARY STEPS

1. `enable`
2. **configure terminal**
3. **voice class e164-pattern-map** *pattern-map-id*
4. Do one of the following:
   - **e164** *pattern-map-tag*
   - **url** *url*
5. *(Optional)* **description** *string*
6. **exit**
7. **dial-peer voice** *dial-peer-id voip*
8. *(destination|incoming called|incoming calling)* **e164-pattern-map** *pattern-map-group-id*
9. **end**
10. *(Optional)* **voice class e164-pattern-map load** *pattern-map-group-id*
11. **show dial-peer voice** *(summary|dial-peer-id)*

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <em>enable</em></td>
<td>Enters privileged EXEC mode.</td>
</tr>
<tr>
<td><em>Example:</em> <strong>Device&gt; enable</strong></td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> <em>configure terminal</em></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><em>Example:</em> <strong>Device# configure terminal</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <strong>voice class e164-pattern-map</strong> <em>pattern-map-id</em></td>
<td>Creates a pattern map for configuring one or multiple E.164 patterns on a dial peer and enters voice class configuration mode.</td>
</tr>
<tr>
<td><em>Example:</em> <strong>Device(config)# voice class e164-pattern-map 1111</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Configure one or more E.164 telephone number prefix match patterns for the pattern map.</td>
</tr>
<tr>
<td><em>Do one of the following:</em></td>
<td></td>
</tr>
<tr>
<td>- <strong>e164</strong> <em>pattern-map-tag</em></td>
<td>- Repeat this step for each pattern if you are using the <strong>e164</strong> command.</td>
</tr>
<tr>
<td>- <strong>url</strong> <em>url</em></td>
<td>- You can specify a file URL containing the patterns for this dial peer using the <strong>url</strong> <em>url</em> command. You must then load the E.164 telephone prefixes using Step 10. The file can be internal (on the device) or external.</td>
</tr>
<tr>
<td><em>Example:</em></td>
<td></td>
</tr>
<tr>
<td><strong>Using URL text file:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Device(voice-class)# url</strong></td>
<td></td>
</tr>
<tr>
<td><strong><a href="http://http-host/config-files/pattern-map.cfg">http://http-host/config-files/pattern-map.cfg</a></strong></td>
<td></td>
</tr>
<tr>
<td><strong>Directly specifying match patterns:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Device(voice-class)# e164 5557123</strong></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>(Optional) <strong>description string</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(voice-class)# <code>description It has 1 entry</code></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>exit</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(voice-class)# <code>exit</code></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>dial-peer voice dial-peer-id voip</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config)# <code>dial-peer voice 2222 voip</code></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>`{destination</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>end</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-dial-peer)# <code>end</code></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>(Optional) <strong>voice class e164-pattern-map load pattern-map-group-id</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device# <code>voice class e164-pattern-map load 1111</code></td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>**show dial-peer voice [summary</td>
</tr>
</tbody>
</table>

### Verifying Multiple Pattern Support on a Voice Dial Peer

**SUMMARY STEPS**

1. `show voice class e164-pattern-map [summary | pattern-map-id]`
2. `show dial-peer voice [summary | dial-peer-id]`
3. `show dialplan incall {sip | h323} {calling | called} e164-pattern`

**DETAILED STEPS**

**Step 1**

`show voice class e164-pattern-map [summary | pattern-map-id]`

Displays the status and contents of a specified pattern map or a status summary of all pattern maps.

**Example:**

```
Device# show voice class e164-pattern-map 200

e164-pattern-map 200
-----------------------------------------
 It has 1 entries
 It is not populated from a file.
 Map is valid.

E164 pattern
-------------
 200
```

**Step 2**

`show dial-peer voice [summary | dial-peer-id]`

Displays the status of pattern maps associated with all or a specified dial peer.

**Example:**

```
Device# show dial-peer voice | include e164-pattern-map

incoming calling e164-pattern-map tag = `200' status = valid,
destination e164-pattern-map tag = 3000 status = valid,

Device# show dial-peer voice 2222 | include e164-pattern-map

incoming calling e164-pattern-map tag = `200' status = valid,
```

**Step 3**

`show dialplan incall {sip | h323} {calling | called} e164-pattern`

Displays inbound dial peer details and associated pattern maps based on an incoming calling or called number.

**Example:**

```
Device# show dialplan incall voip calling 23456

VoiceOverIpPeer1234567
 peer type = voice, system default peer = FALSE, information type = voice,
 description = '',
 tag = 1234567, destination-pattern = '',
 destination e164-pattern-map tag = 200 status = valid,
 destination dpg tag = 200 status = valid,
 voice reg type = 0, corresponding tag = 0,
 allow watch = FALSE
 answer-address = '', preference=0,
 incoming calling e164-pattern-map tag = `200' status = valid,
 CLID Restriction = None
```
Configuration Examples for Multiple Pattern Support on a Voice Dial Peer

Example: Configuring Multiple Patterns for Outbound Dial Peers Using a File URL

Device# voice class e164-pattern-map 1111
Device(voice-class)# url http://http-host/config-files/pattern-map.cfg
Device(voice-class)# description For Outbound Dial Peer
Device(voice-class)# exit
Device(config)# dial-peer voice 2222 voip
Device(voice-dial-peer)# destination e164-pattern-map 1111
Device(voice-dial-peer)# exit
Device(config)# voice class e164-pattern-map load 1111
Device(config)# end

Example: Configuring Multiple Patterns for Outbound Dial Peers bySpecifying Each E164 Pattern

Device# voice class e164-pattern-map 1112
Device(voice-class)# e164 5557456
Device(voice-class)# e164 5557455
Device(voice-class)# e164 5557454
Device(voice-class)# e164 5557453
Device(voice-class)# e164 5557452
Device(voice-class)# description For Outbound Dial Peer
Device(voice-class)# exit
Device(config)# dial-peer voice 2222 voip
Device(voice-dial-peer)# destination e164-pattern-map 1112
Device(voice-dial-peer)# end
!

Example: Configuring Multiple Patterns for Inbound Dial Peer

Device# voice class e164-pattern-map 1113
Device(voice-class)# url http://http-host/config-files/pattern-map.cfg
Device(voice-class)# description For Inbound Dial Peer
Device(voice-class)# exit
Device(config)# dial-peer voice 2222 voip
Device(voice-dial-peer)# incoming calling e164-pattern-map 1113
Device(voice-dial-peer)# exit
Device(config)# voice class e164-pattern-map load 1113
Device(config)# end
CHAPTER 21

Outbound Dial-Peer Group as an Inbound Dial-Peer Destination

This feature can group multiple outbound dial peers into a dial-peer group and configure this dial-peer group as the destination of an inbound dial peer.

- Feature Information for Outbound Dial-Peer Group as an Inbound Dial-Peer Destination, on page 289
- Restrictions, on page 290
- Information About Outbound Dial-Peer Group as an Inbound Dial-Peer Destination, on page 290
- Configuring Outbound Dial-Peer Group as an Inbound Dial-Peer Destination, on page 291
- Verifying Outbound Dial-Peer Groups as an Inbound Dial-Peer Destination, on page 293
- Troubleshooting Tips, on page 294
- Configuration Examples for Outbound Dial Peer Group as an Inbound Dial-Peer Destination, on page 295

Feature Information for Outbound Dial-Peer Group as an Inbound Dial-Peer Destination

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 37: Feature Information for Outbound Dial-Peer Group as an Inbound Dial-Peer Destination

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for POTS dial-peer</td>
<td>Cisco IOS 15.5(1)T</td>
<td>An outgoing POTS dial peer can be part of a dial-peer group. An inbound POTS dial peer can have a dial-peer group as the destination.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.14S</td>
<td></td>
</tr>
</tbody>
</table>
Restrictions

- If a dial-peer group is in the shutdown state, regular dial-peer search occurs.
- If all dial peers in an active dial-peer group are unavailable, call is disconnected.
- The number of matched digits is zero.
- The `destination-pattern` command is required on the outbound dial peer even matching is not done based on this command.
- The outgoing call setup is deferred until inter-digit timer expires or a terminator is entered.

For POTS dial peers:

- Two-stage dialing is not supported.
- Overlapping dialing is not supported.
- TCL and VXML routing changes are not supported.
- Digit-stripping is not supported.

Information About Outbound Dial-Peer Group as an Inbound Dial-Peer Destination

You can group up to 20 outbound (H.323, SIP or POTS) dial peers into a dial-peer group and configure this dial-peer group as the destination of an inbound dial peer. Once an incoming call is matched by an inbound dial peer with an active destination dial-peer group, dial peers from this group are used to route the incoming call. No other outbound dial-peer provisioning to select outbound dial peers is used.

A preference can be defined for each dial peer in a dial-peer group. This preference is used to decide the order of selection of dial peers from the group for the setup of an outgoing call.

You can also specify various dial-peer hunt mechanism using the existing `dial-peer hunt` command.
Configuring Outbound Dial-Peer Group as an Inbound Dial-Peer Destination

Perform this task to configure a dial-peer group with multiple outbound peers and an inbound dial peer referencing this dial-peer group as a destination.

**Before you begin**

- Configure SIP, H.323 or POTS outbound dial peers to be associated with a dial-peer group.
- For an outbound POTS dial peer, ensure that `destination-pattern .T` and `no digit-strip` are configured to avoid unexpected dialed digit strip.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `dial-peer voice outbound-dial-peer-id [voip | pots]`
4. `destination-pattern pattern`
5. `no digit-strip` for POTS dial peers.
6. `exit`
7. (Optional) `dial-peer hunt hunt-order-number`
8. `voice class dpg dial-peer-group-id`
9. `dial-peer outbound-dial-peer-id [preference preference-order]`
10. (Optional) `description string`
11. `exit`
12. `dial-peer voice inbound-dial-peer-id [voip | pots]`
13. `destination dpg dial-peer-group-id`
14. `end`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enters privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; <code>enable</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# <code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>`dial-peer voice outbound-dial-peer-id [voip</td>
<td>pots]`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>For VoIP dial peer:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# dial-peer voice 123 voip</code></td>
<td>Configures a destination pattern. This step is required even though the value is not used for dial-peer matching.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>For POTS dial peer:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# dial-peer voice 345 pots</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>destination-pattern pattern</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>For VoIP Dial Peers</td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# destination-pattern 1004</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>For POTS Dial Peers</td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# destination-pattern .T</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td><code>no digit-strip</code> for POTS dial peers.</td>
<td>Disable unexpected dialed digit strip.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# no digit-strip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td><code>exit</code></td>
<td>Exits to global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# exit</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td><code>(Optional) dial-peer hunt hunt-order-number</code></td>
<td>Specifies a hunt selection mechanism for dial peers.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# dial-peer hunt 0</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td></td>
</tr>
<tr>
<td><code>voice class dpg dial-peer-group-id</code></td>
<td>Creates a dial-peer group for grouping multiple outbound dial peers and enters voice class configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# voice class dpg 181</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td></td>
</tr>
<tr>
<td><code>dial-peer outbound-dial-peer-id [preference preference-order]</code></td>
<td>Associates a configured outbound dial peer with this dial-peer group and configures a preference value.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-class)# dial-peer 123 preference 1</code></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>• If preference is not specified, the order of selection is random or as specified by the <code>dial-peer hunt</code> command.</td>
<td></td>
</tr>
</tbody>
</table>

**Step 10**
(Optional) `description string`

**Example:**

Device(config-class) # description Boston Destination

**Step 11**
`exit`

**Example:**

Device(config-class) # exit

**Step 12**
`dial-peer voice inbound-dial-peer-id [voip | pots]`

**Example:**

For VoIP dial peer:

Device(config) # dial-peer voice 789 voip

For POTS dial peer:

Device(config) # dial-peer voice 678 pots

**Step 13**
`destination dpg dial-peer-group-id`

**Example:**

Device(config-dial-peer) # destination dpg 181

**Step 14**
`end`

**Example:**

Device(config-dial-peer) # end

---

**Verifying Outbound Dial-Peer Groups as an Inbound Dial-Peer Destination**

**SUMMARY STEPS**

1. `show voice class dpg dial-peer-group-id`
2. `show dial-peer voice inbound-dial-peer-id`
DETAILED STEPS

Step 1  
**show voice class dpg  dial-peer-group-id**
Displays the configuration of an outbound dial-peer group.

**Example:**
```
Device# show voice class dpg 200
```

Voice class dpg: 200  AdminStatus: Up
Description: Boston Destination
Total dial-peer entries: 4

<table>
<thead>
<tr>
<th>Peer Tag</th>
<th>Pref</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>2</td>
</tr>
<tr>
<td>1004</td>
<td>0</td>
</tr>
<tr>
<td>1003</td>
<td>1</td>
</tr>
</tbody>
</table>

Step 2  
**show dial-peer voice  inbound-dial-peer-id**
Displays the referencing of destination dial-peer group from an inbound dial peer.

**Example:**
```
Device# show dial-peer voice 100 | include destination dpg
```

destination dpg tag = 200 status = valid,

Troubleshooting Tips

**SUMMARY STEPS**

1. Enter the following:
   - debug voip dialpeer inout
   - debug voip ccapi inout

**DETAILED STEPS**

Enter the following:
   - debug voip dialpeer inout
   - debug voip ccapi inout

Displays the configuration of an outbound dial-peer group.

**Example:**
Configuration Examples for Outbound Dial Peer Group as an Inbound Dial-Peer Destination

![Configuration Examples for Outbound Dial Peer Group as an Inbound Dial-Peer Destination](image-url)
Verifying Outbound Dial-Peer Group Configuration

Device# show voice class dpg 200

Voice class dpg: 200  AdminStatus: Up
Description: Boston Destination
Total dial-peer entries: 4

<table>
<thead>
<tr>
<th>Peer Tag</th>
<th>Pref</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>2</td>
</tr>
<tr>
<td>1003</td>
<td>0</td>
</tr>
<tr>
<td>1004</td>
<td>1</td>
</tr>
</tbody>
</table>
Verifying Inbound Dial-Peer Referencing Outbound Dial-Peer Group

Device# show dial-peer voice 100 | include destination dpg

  destination dpg tag = 200 status = valid,

Device# show dial-peer voice 600 | include destination dpg

  destination dpg tag = 200 status = valid,
Configuration Examples for Outbound Dial Peer Group as an Inbound Dial-Peer Destination
Inbound Leg Headers for Outbound Dial-Peer Matching

The Inbound Leg Headers for Outbound Dial-Peer Matching feature allows you to match and provision an outbound dial peer for an outbound call leg using the headers from an inbound call leg. The following headers of an incoming call leg can be used for outbound dial-peer matching:

- VIA (SIP Header)
- FROM (SIP Header)
- TO (SIP Header)
- DIVERSION (SIP Header)
- REFERRED BY (SIP Header)
- Called Number
- Calling Number
- Carrier ID

- Feature Information for Inbound Leg Headers for Outbound Dial-Peer Matching, on page 299
- Prerequisites for Inbound Leg Headers for Outbound Dial-Peer Matching, on page 300
- Restrictions for Inbound Leg Headers for Outbound Dial-Peer Matching, on page 300
- Information About Inbound Leg Headers for Outbound Dial-Peer Matching, on page 300
- Configuring Inbound Leg Headers for Outbound Dial-Peer Matching, on page 301
- Verifying Inbound Leg Headers for Outbound Dial-Peer Matching, on page 304
- Configuration Example: Inbound Leg Headers for Outbound Dial-Peer Matching, on page 306

Feature Information for Inbound Leg Headers for Outbound Dial-Peer Matching

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
### Prerequisites for Inbound Leg Headers for Outbound Dial-Peer Matching

- CUBE or Voice Gateway must be configured.

### Restrictions for Inbound Leg Headers for Outbound Dial-Peer Matching

- The existing **header-passing** command supports modification of SIP headers of INVITE message by the Tool Command Language (TCL) application. If the above SIP headers are modified by the TCL application, they cannot be used for outbound dial-peer provisioning.

- If multiple SIP via headers and diversion headers are found in an incoming INVITE or REFER message, only the top-most via header and top-most diversion header of an incoming INVITE or REFER message are used for outbound dial-peer provisioning.

- When an incoming call is matched to an inbound dial peer with an associated provision profile without rules, outbound dial-peer provisioning is disabled and the incoming call is disconnected by CUBE or voice gateway with cause code "unassigned number (1)".

### Information About Inbound Leg Headers for Outbound Dial-Peer Matching

This feature allows you to match headers of an inbound call leg and provision an outbound dial peer for an outbound call leg. The following SIP headers of an incoming call leg can be used for outbound dial-peer matching

- VIA (SIP Header)
The above headers are retrieved from an incoming INVITE or REFER message and used for outbound dial-peer provisioning.

SIP headers of an INVITE message are saved to an associated call leg. For example, an INVITE message is received for a new call leg A. Then, SIP headers are saved to call leg A itself for outbound dial-peer lookup.

On the other hand, SIP headers of a REFER message are saved to the peer call leg of the associated call leg. For example, call leg A and call leg B are connected in CUBE. The party at Call Leg B makes a blind transfer to the party at Call Leg C. A REFER message is received in CUBE for call leg B (transferor). But, SIP headers of the REFER message are saved under call leg A (transferee) for an outbound dial-peer lookup for Party C.

### Configuring Inbound Leg Headers for Outbound Dial-Peer Matching

#### Before you begin

Necessary pattern maps have been configured.

#### SUMMARY STEPS

1. enable
2. configure terminal
3. voice class dial-peer provision-policy *tag*
4. (Optional) description *string*
5. preference *preference-order* first-attribute second-attribute
6. exit
7. dial-peer voice inbound-dial-peer-tag *voip*
8. destination provision-policy *tag*
9. exit
10. dial-peer voice outbound-dial-peer-tag *voip*
11. Configure a match command for an outbound dial peer according to the provision policy rule attribute configured.
12. end

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enters privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>

---

Dial Peer Enhancements

Configuring Inbound Leg Headers for Outbound Dial-Peer Matching

---

Cisco Unified Border Element Configuration Guide
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device&gt; enable</strong></td>
<td><strong>Enters global configuration mode.</strong></td>
</tr>
</tbody>
</table>
| **Step 2** **configure terminal**  
**Example:**  
Device# configure terminal | **Creates a provision policy profile in which a set of attributes for dial-peer matching can be defined.**  
- You can use the **shutdown** command to deactivate the provision policy and allow normal outbound dial-peer provisioning. |
| **Step 3** **voice class dial-peer provision-policy tag**  
**Example:**  
Device(config)# voice class dial-peer provision-policy 200 | **Provides a description for the provision policy profile.** |
| **Step 4**  
(Optional) **description string**  
**Example:**  
Device(voice-class)# description match both calling and called | **Configures a provision policy rule.**  
- You can configure up to two rules. This means up to four attributes can be configured for matching outbound dial peers.  
- If rules are not configured, outbound dial-peer provisioning is disabled, and an incoming call matched to an inbound dial peer associated with this profile is disconnected by CUBE or voice gateway with cause code “unassigned number (1)”.
| **Step 5**  
**preference preference-order first-attribute second-attribute**  
**First Attribute** | **Second Attribute** |
| diversion       | from, referred-by, to, uri, via |
| from            | diversion, referred-by, to, uri, via |
| referred-by     | diversion, from, to, uri, via |
| to              | diversion, referred-by, from, uri, via |
| uri             | diversion, referred-by, to, from, via, carrier-id |
| via             | diversion, referred-by, to, uri, via |
| called          | calling, carrier-id |
| calling         | called |
| carrier-id      | called, uri |

**Example:**  
Device(voice-class)# preference 2 calling called |

| **Step 6** **exit**  
**Example:**  
Device(voice-class)# exit | **Exits voice class configuration mode and enters global configuration mode.** |
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><code>dial-peer voice inbound-dial-peer-tag voip</code></td>
<td>Enters dial peer configuration mode for an inbound dial peer.</td>
</tr>
<tr>
<td>8</td>
<td><code>destination provision-policy tag</code></td>
<td>Associates a provision policy profile with an inbound dial peer.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong>&lt;br&gt;<code>Device(config)# dial-peer voice 100 voip</code>&lt;br&gt;<code>Device(config-dial-peer)# destination provision-policy 200</code>&lt;br&gt;<code>Device(config)# exit</code></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><code>exit</code></td>
<td>Exits dial peer configuration mode.</td>
</tr>
<tr>
<td>10</td>
<td><code>dial-peer voice outbound-dial-peer-tag voip</code></td>
<td>Enters dial peer configuration mode for an outbound dial peer.</td>
</tr>
<tr>
<td>11</td>
<td>Configure a match command for an outbound dial peer according to the provision policy rule attribute configured.</td>
<td>Configure a match command based on any of the four attributes defined in the provision policy rule.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Provision Policy Rule Attribute</th>
<th>Dial-peer Match command</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>called</strong></td>
<td><code>destination-pattern pattern</code>&lt;br&gt;<code>destination e164-pattern-map pattern-map-class-id</code></td>
</tr>
<tr>
<td><strong>calling</strong></td>
<td><code>destination calling e164-pattern-map pattern-map-class-id</code></td>
</tr>
<tr>
<td><strong>carrier-id</strong></td>
<td><code>carrier-id target</code></td>
</tr>
<tr>
<td><strong>uri</strong></td>
<td><code>destination uri uri-class-tag</code></td>
</tr>
<tr>
<td><strong>via</strong></td>
<td><code>destination uri-via uri-class-tag</code></td>
</tr>
<tr>
<td><strong>to</strong></td>
<td><code>destination uri-to uri-class-tag</code></td>
</tr>
<tr>
<td><strong>from</strong></td>
<td><code>destination uri-from uri-class-tag</code></td>
</tr>
<tr>
<td><strong>diversion</strong></td>
<td><code>destination uri-diversion uri-class-tag</code></td>
</tr>
<tr>
<td><strong>referred-by</strong></td>
<td><code>destination uri-referred-by uri-class-tag</code></td>
</tr>
</tbody>
</table>

**Example:**<br>`Device(config)# dial-peer voice 300 voip`<br>`Device(config-dial-peer)# destination uri-from 200`<br>`Device(config)# exit` |

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td><code>end</code></td>
<td>Exits dial peer configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong>&lt;br&gt;<code>Device(config)# dial-peer voice 300 voip</code>&lt;br&gt;<code>Device(config-dial-peer)# destination uri-from 200</code>&lt;br&gt;<code>Device(config)# exit</code></td>
<td></td>
</tr>
</tbody>
</table>
Verifying Inbound Leg Headers for Outbound Dial-Peer Matching

**SUMMARY STEPS**

1. `show dialplan incall {sip | h323} {calling | called} e164-pattern | include voice`
2. `show dialplan dialpeer inbound-dial-peer-id number e164-pattern [timeout] | include Voice`
3. `show voice class dial-peer provision-policy`

**DETAILED STEPS**

Step 1

`show dialplan incall {sip | h323} {calling | called} e164-pattern | include voice`

Displays inbound dial peers based on an incoming calling or called number. Once you have the dial peer number, you can use it to search for the complete dial-peer details in the running-config.

**Example:**

Device# `show dialplan incall sip calling 3333 | include Voice`

VoiceOverIpPeer1

Device# `show dialplan incall sip calling 4444 | include Voice`

VoiceOverIpPeer1

Device# `show running-config | section dial-peer voice 1 voip`

dial-peer voice 1 voip
    destination dpg 10000
    incoming calling e164-pattern-map 100
dtmf-relay rtp-nte
    codec g711ulaw

Device# `show dialplan incall sip called 6000 timeout | include Voice`

VoiceOverIpPeer100

Device# `show running-config | section dial-peer voice 100 voip`

dial-peer voice 100 voip
    incoming called e164-pattern-map 1
    incoming calling e164-pattern-map 1
dtmf-relay rtp-nte
    codec g711ulaw

Device# `show dialplan incall voip calling 23456`

VoiceOverIpPeer1234567

    peer type = voice, system default peer = FALSE, information type = voice,
description = '',
tag = 1234567, destination-pattern = '',
destination e164-pattern-map tag = 200 status = valid,
destination dpq tag = 200 status = valid,
voice reg type = 0, corresponding tag = 0,
allow watch = FALSE
answer-address = '', preference=0,
incoming calling e164-pattern-map tag = '200' status = valid,
CLID Restriction = None

Step 2  
show dialplan dialpeer inbound-dial-peer-id number e164-pattern [timeout] | include Voice
Displays a list of outbound dial peers based on a specified inbound dial peer. This command line will be helpful find a list of outbound dial peer of a destination dial-peer group.

Example:
Device# show dialplan dialpeer 1 number 23457 timeout | include Voice
VoiceOverIpPeer100013
VoiceOverIpPeer100012

Example:
voice class dial-peer provision-policy 2000
  preference 2 diversion to
!...
!
dial-peer voice 32555 voip
  session protocol sipv2
  session target ipv4:1.5.14.9
  destination uri-diversion 1
  destination uri-to test2
!
dial-peer voice 32991 voip
  destination provision-policy 2000
  incoming called-number 1234
!
Device# show dialplan dialpeer 32991 number 2234 timeout
Macro Exp.: 2234
Enter Diversion header:sip:1234@cisco.com
Enter To header:sip:2234@10.0.0.0
VoiceOverIpPeer32134
  peer type = voice, system default peer = FALSE, information type = voice,
  description = '',

Step 3  
show voice class dial-peer provision-policy
Displays a list of configured provision policies and associated rules.

Example:
Device# show voice class dial-peer provision-policy
Voice class dial-peer provision-policy: 100 AdminStatus: Up
Description: match only called

Pref  Policy Rule
----  --------------
1     called

Voice class dial-peer provision-policy: 101 AdminStatus: Up
Description: match both calling and called

Pref  Policy Rule
Configuration Example: Inbound Leg Headers for Outbound Dial-Peer Matching

Example: Configuring Inbound Called or Calling Numbers Used for Outbound Dial-Peer Matching

Device> enable
Device# configure terminal

Device(config)# voice class dial-peer provision-policy 200
Device(config)# description match both calling and called
Device(config)# preference 2 calling called
Device(config)# exit

Device(config)# voice class e164-pattern-map 300
Device(config)# description patterns
Device(config)# e164 5557123
Device(config)# e164 5558123
Device(config)# e164 5559123
Device(config)# exit

!Associating the Provision Policy with an Inbound Dial Peer
Device(config)# dial-peer voice 100 voip
Device(config-dial-peer)# destination provision-policy 200
Device(config-dial-peer)# end

!Associates a Pattern Map with an Outbound Dial Peer.
! The called number in the SIP headers of the inbound leg is matched to select the below
outbound dial peer.
Device(config)# dial-peer voice 200 voip
Device(config-dial-peer)# destination e164-pattern-map 300
Device(config-dial-peer)# end

Example: Configuring Inbound SIP Headers for Outbound Dial-Peer Matching

Device> enable
Device# configure terminal

Device(config)# voice class dial-peer provision-policy 200
Device(voice-class)# description match both calling and called
Device(voice-class)# preference 2 via from
Device(voice-class)# exit

!Associating the Provision Policy with an Inbound Dial Peer
Device(config)# dial-peer voice 100 voip
Device(config-dial-peer)# destination provision-policy 200
Device(config-dial-peer)# end

Device(config)# voice class uri 200 sip
Device(config-voice-uri-clas)# pattern 25054..

!Associates a Provision Policy with an Outbound Dial Peer.
The FROM SIP headers of the inbound leg is matched to select the below outbound dial peer.
Device(config)# dial-peer voice 200 voip
Device(config-dial-peer)# destination uri-from 200
Device(config-dial-peer)# end
Server Groups in Outbound Dial Peers

This feature configures a server group (group of server addresses) that can be referenced from an outbound dial peer.

- Feature Information for Configuring Server Groups in Outbound Dial Peers, on page 309
- Information About Server Groups in Outbound Dial Peers, on page 310
- How to Configure Server Groups in Outbound Dial Peers, on page 310
- Configuration Examples for Server Groups in Outbound Dial Peers, on page 313

Feature Information for Configuring Server Groups in Outbound Dial Peers

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 39: Feature Information for Configuring Server Groups in Outbound Dial Peers

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring Server Groups in Outbound Dial Peers</td>
<td>Cisco IOS XE Release 3.11S 15.4(1)T</td>
<td>This feature configures server groups (groups of IPv4 and IPv6 addresses) which can be referenced from an outbound SIP dial peer. The following commands were introduced or modified: voice class server-group, description, ipv4 port preference, ipv6 port preference, hunt-scheme, show voice class server-group, shutdown (Server Group).</td>
</tr>
</tbody>
</table>
Information About Server Groups in Outbound Dial Peers

You can now group IPv4 and IPv6 addresses of servers and configure it as an outbound SIP dial-peer destination. A server group is first created and associated with a SIP outbound dial peer. When a call matches an outbound dial peer, a connection is made to one of the servers of the group, which can then route the call to the destination.

You can associate a preference with each of the servers in the group, which decides the order of selection for outgoing call setup.

You can also configure the round-robin selection of servers in the group.

You can configure up to five servers per server group.

If neither round robin nor preference order is configured, the selection of servers is random.

If the specified server-group is in shutdown mode, the referenced dial peers are not selected to route outgoing calls.

Note

If the configured IP address in a server group is unreachable, you receive an error message. If you receive an error message with code 404, 500, or 503, hunting for the next destination IP address is supported in destination server groups. However, for a destination server group that is configured in a dial-peer, hunting for the next destination IP address is not supported if you receive an error message with code 480, 486, or 600. As a workaround, it is recommended that you configure dial-peer with preference (Hunt Groups) instead of server groups.

How to Configure Server Groups in Outbound Dial Peers

Configuring Server Groups in Outbound Dial Peers

SUMMARY STEPS

1. enable
2. configure terminal
3. voice class server-group server-group-id
4. {ipv4 | ipv6} address [port port] [preference preference-order]
5. (Optional) hunt-scheme round-robin
6. (Optional) description string
7. exit
8. dial-peer voice dial-peer-id voip
9. session protocol sipv2
10. destination-pattern [+] string [T]
11. session server-group server-group-id
12. end
13. show voice class server-group server-group-id
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters privileged EXEC mode.</td>
</tr>
<tr>
<td><em>enable</em></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><em>configure terminal</em></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Configures a voice class server group and enters voice class configuration mode.</td>
</tr>
<tr>
<td><em>voice class server-group server-group-id</em></td>
<td>• You can use the <strong>shutdown</strong> command to make the server group inactive.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Configures a server IP address as a part of this server group along with an optional port number and preference order.</td>
</tr>
<tr>
<td>*{ipv4</td>
<td>ipv6} address [port port] [preference preference-order]*</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Defines a hunt method for the order of selection of target server IP addresses (from the IP addresses configured for this server group) for the setting up of outgoing calls.</td>
</tr>
<tr>
<td><em>(Optional) hunt-scheme round-robin</em></td>
<td>• If a hunt scheme is not defined, an available IP address of highest preference value is selected. If neither a round-robin hunt scheme nor a preference values is configured, the selection of servers is random.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Provides a description for the server group.</td>
</tr>
<tr>
<td><em>(Optional) description string</em></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Exits voice class mode and enters global configuration mode.</td>
</tr>
<tr>
<td><em>exit</em></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>8</td>
<td><code>dial-peer voice dial-peer-id voip</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td><code>Device(config)# dial-peer voice 123 voip</code></td>
</tr>
<tr>
<td>9</td>
<td><code>session protocol sipv2</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td><code>Device(config-dial-peer)# session protocol sipv2</code></td>
</tr>
<tr>
<td>10</td>
<td><code>destination-pattern [+][string [T]]</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td><code>Device(config-dial-peer)# destination-pattern +5550179</code></td>
</tr>
</tbody>
</table>
| 11   | `session server-group server-group-id` | Configures the specified server group as the destination of the dial peer.  
- This command is available for SIP dial peers only.  
- If the specified server group is in shutdown mode, the dial peer is not selected to route outgoing calls. |
|      | **Example:** | |
|      | `Device(config-dial-peer)# session server-group 171` | |
| 12   | `end` | Exits dial peer configuration mode and enters privileged EXEC mode. |
|      | **Example:** | |
|      | `Device(config-dial-peer)# end` | |
| 13   | `show voice class server-group server-group-id` | Displays information about the voice class server group. |
|      | **Example:** | |
|      | `Device# show voice class server-group 171` | |

### Verifying Server Groups in Outbound Dial Peers

**SUMMARY STEPS**

1. `show voice class server-group [server-group-id]`
2. `show voice class server-group dialpeer dial-peer-id`

**DETAILED STEPS**

**Step 1**  
`show voice class server-group [server-group-id]`  
Displays the configurations for all configured server groups or a specified server group.  
**Example:**
Device# `show voice class server-group 171`

Voice class server-group: 171
AdminStatus: Up OperStatus: Up
Hunt-Scheme: preference Last returned server: 10.1.1.1
Description:
Total server entries: 3

<table>
<thead>
<tr>
<th>Pref</th>
<th>Type</th>
<th>IP Address</th>
<th>IP Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ipv4</td>
<td>10.1.1.1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ipv4</td>
<td>10.1.1.2</td>
<td>34515</td>
</tr>
<tr>
<td>3</td>
<td>ipv4</td>
<td>2001:DB8:12:1::26</td>
<td></td>
</tr>
</tbody>
</table>

Step 2 `show voice class server-group dialpeer dial-peer-id`  
Displays the configurations for dial peers associated with server groups.

Example:

Device# `show voice class server-group dialpeer 181`

Voice class server-group: 171 AdminStatus: Up
Hunt-Scheme: round-robin
Total Remote Targets: 3

<table>
<thead>
<tr>
<th>Pref</th>
<th>Type</th>
<th>IP Address</th>
<th>IP Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ipv4</td>
<td>10.1.1.2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ipv4</td>
<td>10.5.0.1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ipv4</td>
<td>10.1.1.1</td>
<td></td>
</tr>
</tbody>
</table>

Configuration Examples for Server Groups in Outbound Dial Peers

Server Groups in Outbound Dial Peers (Preference-Based Selection)

! Configuring the Server Group
Device(config)# `voice class server-group 171`
Device(config-class)# `ipv4 10.1.1.1 preference 1`
Device(config-class)# `ipv4 10.1.1.2 preference 2`
Device(config-class)# `ipv4 10.1.1.3 preference 3`
Device(config-class)# `description It has 3 entries`
Device(config-class)# `exit`

! Configuring an outbound SIP dial peer.
Device(config)# `dial-peer voice 181 voip`
Device(config-dial-peer)# `destination-pattern 3001`
Device(config-dial-peer)# `session protocol sipv2`
Device(config-dial-peer)# `session server-group 171`
Configuration Examples for Server Groups in Outbound Dial Peers

Server Groups in Outbound Dial Peers (Round-Robin-Based Selection)

! Configuring the Server Group
Device(config)# voice class server-group 171
Device(config-class)# ipv4 10.1.1.1
Device(config-class)# ipv4 10.1.1.2
Device(config-class)# ipv4 10.1.1.3
Device(config-class)# hunt-scheme round-robin
Device(config-class)# description It has 3 entries
Device(config-class)# exit

! Configuring an outbound SIP dial peer.
Device(config)# dial-peer voice 181 voip
! Associate a destination pattern
Device(config-dial-peer)# destination-pattern 3001
Device(config-dial-peer)# session protocol sipv2
! Associate a server group with the dial peer
Device(config-dial-peer)# session server-group 171
Device(config-dial-peer)# end

! Displays the configurations made for the outbound dial peer 181 associated with a server group
Device# show voice class server-group dial-peer 181

Voice class server-group: 171 AdminStatus: Up
Hunt-Scheme: round-robin
Total Remote Targets: 3

<table>
<thead>
<tr>
<th>Pref</th>
<th>Type</th>
<th>IP Address</th>
<th>IP Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ipv4</td>
<td>10.1.1.1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ipv4</td>
<td>10.1.1.2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ipv4</td>
<td>10.1.1.3</td>
<td></td>
</tr>
</tbody>
</table>

---

Cisco Unified Border Element Configuration Guide

314
<table>
<thead>
<tr>
<th>Pref</th>
<th>Type</th>
<th>IP Address</th>
<th>IP Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ipv4</td>
<td>10.1.1.1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>ipv4</td>
<td>10.1.1.2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>ipv4</td>
<td>10.1.1.3</td>
<td></td>
</tr>
</tbody>
</table>

! Displays the configurations made for the server group.

Device# `show voice class server-group 171`

Voice class server-group: 171
AdminStatus: Up
OperStatus: Up
Hunt-Scheme: round-robin
Last returned server: 10.1.1.1
Description: It has 3 entries
Total server entries: 3

<table>
<thead>
<tr>
<th>Pref</th>
<th>Type</th>
<th>IP Address</th>
<th>IP Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ipv4</td>
<td>10.1.1.1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>ipv4</td>
<td>10.1.1.2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>ipv4</td>
<td>10.1.1.3</td>
<td></td>
</tr>
</tbody>
</table>

-------------------------------------
Chapter 24

Domain-Based Routing Support on the Cisco UBE

First Published: June 15, 2011
Last Updated: July 22, 2011

The Domain-based routing feature provides support for matching an outbound dial peer based on the domain name or IP address provided in the request URI of the incoming SIP message or an inbound dial peer.

Domain-based routing enables for calls to be routed on the outbound dialpeer based on the domain name or IP address provided in the request Uniform Resource Identifier (URI) of incoming Session IP message.

- Feature Information for Domain-Based Routing Support on the Cisco UBE, on page 317
- Restrictions for Domain-Based Routing Support on the Cisco UBE, on page 318
- Information About Domain-Based Routing Support on the Cisco UBE, on page 318
- How to Configure Domain-Based Routing Support on the Cisco UBE, on page 319
- Configuration Examples for Domain-Based Routing Support on the Cisco UBE, on page 324

Feature Information for Domain-Based Routing Support on the Cisco UBE

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 40: Feature Information for Domain-Based Routing Support on the Cisco UBE

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Based Routing Support on the Cisco UBE</td>
<td>15.2(1)T</td>
<td>The domain-based routing enables for calls to be routed on the outbound dial peer based on the domain name or IP address provided in the request URI (Uniform Resource Identifier) of incoming SIP message. The following commands were introduced or modified: call-route, voice-class sip call-route.</td>
</tr>
<tr>
<td>Domain Based Routing Support on the Cisco UBE</td>
<td>Cisco IOS XE Release 3.8S</td>
<td>The domain-based routing enables for calls to be routed on the outbound dial peer based on the domain name or IP address provided in the request URI (Uniform Resource Identifier) of incoming SIP message. The following commands were introduced or modified: call-route, voice-class sip call-route.</td>
</tr>
</tbody>
</table>

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental. © 2011 Cisco Systems, Inc. All rights reserved

Restrictions for Domain-Based Routing Support on the Cisco UBE

Domain-based routing support is available only for SIP-SIP call flows.

Information About Domain-Based Routing Support on the Cisco UBE

When a dial peer has an application configured as a session application, then only the user parameter of the request URI is used and is sent from the inbound SIP SPI to the application. The session application performs a match on an outbound dial peer based on the user parameter of the request URI sent from the inbound dial peer. In the figure below, 567 is the user portion of the request-URI that is passed from the inbound dial peer to the application and the matching outbound dial-peer found is 1000.

Cisco Unified Border Element Configuration Guide
With the introduction of the domain-based routing feature, all parameters including the domain name of the request URI will be sent to the application and the outbound dial peer can be matched with any parameter. In Figure 1, when the domain name example.com is used to match an outbound dial peer the resulting dial peer is 2000. The **call route url** command is used for configuring domain-based routing.

**Note** Whenever using the **call route url** command, apply translation rule at outbound dial-peer not in to call-route url.

---

**How to Configure Domain-Based Routing Support on the Cisco UBE**

**Configuring Domain-Based Routing at Global Level**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. sip
5. call-route url
6. exit
## Configuring Domain-Based Routing at Dial Peer Level

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `dial-peer voice dial-peer tag voip`
4. `voice-class sip call-route url`
5. `exit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td>Example: Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> call-route url</td>
<td>Routes calls based on the URL.</td>
</tr>
<tr>
<td>Example: Device(conf-serv-sip)# call-route url</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>Example: Device(conf-serv-sip)# exit</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>configure terminal</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device# configure terminal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dial-peer voice dial-peer tag voip</td>
<td>Enter dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# dial-peer voice 2 voip</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 4</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>voice-class sip call-route url</td>
<td>Routes calls based on the URL</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)#</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 5</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>

**Verifying and Troubleshooting Domain-Based Routing Support on the Cisco UBE**

**SUMMARY STEPS**

1. `enable`
2. `debug ccsip all`
3. `debug voip dialpeer inout`

**DETAILED STEPS**

**Step 1**

```
enable
```

Enables privileged EXEC mode.

**Example:**

```
Device> enable
```

**Step 2**

```
debug ccsip all
```

Enables all SIP-related debugging.

**Example:**

```
Device# debug ccsip all
```

Received:
Via: SIP/2.0/UDP [2208:1:1:1:1:1:1:1115]:5060;branch=z9hG4bK83AE3
Verifying and Troubleshooting Domain-Based Routing Support on the Cisco UBE

Dial Peer Enhancements

Cisco Unified Border Element Configuration Guide

322

Verifying and Troubleshooting Domain-Based Routing Support on the Cisco UBE

Date: Tue, 01 Mar 2011 08:49:48 GMT
Call-ID: B30FCDEB-431711E0-8EDCB51-E9F6BBF1@2208:1:1:1:1:1:1:1115
Supported: 100rel,timer,resource-priority,replaces
Require: sdp-anat
Min-SE: 1800
Cisco-Guid: 2948477781-1125585376-2396638033-3925258737
User-Agent: Cisco-SIPGateway/IOS-15.1(3.14.2)PIA16
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1298969388
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 495
v=0
c=IN IP6 2208:1:1:1:1:1:1:1115
s=SIP Call
c=IN IP6 2208:1:1:1:1:1:1:1115
t=0 0
a=group:ANAT 1 2
m=audio 17836 RTP/AVP 0 101 19
c=IN IP6 2208:1:1:1:1:1:1:1115
a-mid:1
a-rtpmap:0 PCMU/8000
a-rtpmap:101 telephone-event/8000
a-fmtmap:101 0-16
a-rtpmap:19 CN/8000
a-ptime:20
m-audio 18938 RTP/AVP 0 101 19
c=IN IP6 2208:1:1:1:1:1:1:1115
a-mid:2
a-rtpmap:0 PCMU/8000
a-rtpmap:101 telephone-event/8000
a-fmtmap:101 0-16
a-rtpmap:19 CN/8000
a-ptime:20
"Received:
Via: SIP/2.0/UDP [2208:1:1:1:1:1:1:1116]:5060;branch=z9hG4bK38ACE
Date: Thu, 10 Feb 2011 12:15:08 GMT
Call-ID: 5DEDB77E-ADC11208-808BE770-8FCACF34@2208:1:1:1:1:1:1:1117
Supported: 100rel,timer,resource-priority,replaces,sdp-anat
Min-SE: 1800
Cisco-Guid: 1432849350-0876876256-2424621905-3925258737
User-Agent: Cisco-SIPGateway/IOS-15.1(3.14.2)PIA16
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1297340108
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Length: 424
v=0
s=SIP Call
c=IN IP6 2208:1:1:1:1:1:1:1116
t=0 0
m=image 17278 udptl t38
c=IN IP6 2208:1:1:1:1:1:1:1116
a=T38FaxVersion:0
a=T38MaxBitRate:14400
a=T38FaxFillBitRemoval:0
a=T38FaxTranscodingMMR:0
a=T38FaxTranscodingJBIG:0
a=T38FaxRateManagement:transferredTCF
a=T38FaxMaxBuffer:200
a=T38FaxMaxDatagram:320
a=T38FaxUdpEC:t38UDPRedundancy"

Step 3  
**debug voip dialpeer inout**

The **debug ccsp all** and **debug voip dialpeer inout** commands can be entered in any order and any of the commands can be used for debugging depending on the requirement.

**Example:**

Displays information about the voice dial peers
Device# debug voip dialpeer inout
voip dialpeer inout debugging is on

The following event shows the calling and called numbers:

**Example:**

*May 1 19:32:11.731: */-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:  
Calling Number=4085550111, Called Number=3600, Voice-Interface=0x0,  
Timeout=TRUE, Peer Encap Type=ENCAP_VOIP, Peer Search Type=PEER_TYPE_VOICE,  
Peer Info Type=DIALPEER_INFO_SPEECH*

The following event shows the incoming dial peer:

**Example:**

*May 1 19:32:11.731: */-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:  
Result=Success(0) after DP_MATCH_INCOMING_DNIS; Incoming Dial-peer=100  
*May 1 19:32:11.731: */-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:  
Calling Number=4085550111, Called Number=3600, Voice-Interface=0x0,  
Timeout=TRUE, Peer Encap Type=ENCAP_VOIP, Peer Search Type=PEER_TYPE_VOICE,  
Peer Info Type=DIALPEER_INFO_SPEECH*

*May 1 19:32:11.731: */-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:  
Result=Success(0) after DP_MATCH_INCOMING_DNIS; Incoming Dial-peer=100  
*May 1 19:32:11.735: */-1/6372E2598012/DPM/dpMatchPeersCore:  
Calling Number=, Called Number=3600, Peer Info Type=DIALPEER_INFO_SPEECH  
*May 1 19:32:11.735: */-1/6372E2598012/DPM/dpMatchPeersCore:  
Match Rule=DP_MATCH_DEST; Called Number=3600  
*May 1 19:32:11.735: */-1/6372E2598012/DPM/dpMatchPeersCore:  
Result=Success(0) after DP.Match_DEST  
*May 1 19:32:11.735: */-1/6372E2598012/DPM/dpMatchPeersMoreArg:  
Result=SUCCESS(0)

The following event shows the matched dial peers in the order of priority:

**Example:**

List of Matched Outgoing Dial-peer(s):
Configuration Examples for Domain-Based Routing Support on the Cisco UBE

Example Configuring Domain-Based Routing Support on the Cisco UBE

The following example shows how to enable domain-based routing support on the Cisco UBE:

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# call-route url
Device(conf-serv-sip)# exit
Device(config)# dial-peer voice 2 voip
Device(config-dial-peer)# voice-class sip call-route url
Device(config-dial-peer)# exit
```
Chapter 25

ENUM Enhancement per Kaplan Draft RFC

The Cisco Unified Border Element (CUBE) facilitates the mapping of E.164 called numbers to Session Initiation Protocol (SIP) Uniform Resource Identifiers (URIs). The SIP ENUM technology allows the traditional telephony part of the network (using E.164 numbering to address destinations) to interwork with the SIP telephony part of the network, generally using SIP URLs. From the Public Switched Telephone Network (PSTN) network, if an end user dials an E.164 called party, the number can be translated by an ENUM gateway into the corresponding SIP URI. This SIP URI is then used to look up the Domain Name System (DNS) Naming Authority Pointer (NAPTR) Resource Records (RR). The NAPTR RR (as defined in RFC 2915) describes how the call should be forwarded or terminated and records information, such as email addresses, a fax number, a personal website, a VoIP number, mobile telephone numbers, voice mail systems, IP-telephony addresses, and web pages. Alternately, when the calling party is a VoIP endpoint and dials an E.164 number, then the originator's SIP user agent (UA) converts it into a SIP URI to be used to look up at the ENUM gateway DNS and fetch the NAPTR RR.

The ENUM enhancement per Kaplan draft RFC provides source-based routing, that is, SIP-to-SIP calls can be routed based on the source SIP requests. To provide source-based routing and to interact with the Policy Server, an EDNS0 OPT pseudo resource record with source URI, incoming SIP call ID, outbound SIP call ID, and Call Session Identification are added to the ENUM DNS query, according to draft-kaplan-enum-sip-routing-04. The incoming SIP call ID, outbound SIP call ID, and Call Session Identification are automatically included with an EDNS0 OPT pseudo resource record in the ENUM DNS query only if “source-uri no-cache” is enabled and XCC service is registered. This feature also provides the flexibility to disable route caching.

- Feature Information for ENUM Enhancement per Kaplan Draft RFC, on page 325
- Restrictions for ENUM Enhancement per Kaplan Draft RFC, on page 326
- Information About ENUM Enhancement per Kaplan Draft RFC, on page 327
- How to Configure ENUM Enhancement per Kaplan Draft RFC, on page 327
- Troubleshooting Tips, on page 330
- Configuration Examples for ENUM Enhancement per Kaplan Draft RFC, on page 330

Feature Information for ENUM Enhancement per Kaplan Draft RFC

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 41: Feature Information for ENUM Enhancement per Kaplan Draft RFC

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENUM Enhancement per Kaplan Draft RFC</td>
<td>Cisco IOS XE 3.14S, Cisco IOS 15.5(1)T</td>
<td>The ENUM enhancement per Kaplan draft RFC provides source-based routing, that is, SIP-to-SIP calls can be routed based on the source SIP requests. To provide this source-based routing, an EDNS0 OPT pseudo resource record with source URI is added to the ENUM DNS query, according to draft-kaplan-enum-sip-routing-04. This feature also provides the flexibility to disable route caching.</td>
</tr>
<tr>
<td>Support to include inbound call ID, outbound call ID and Call Session Identification to ENUM DNS query</td>
<td>Cisco IOS 15.5(2)T, Cisco IOS XE 3.15S</td>
<td>This feature allows you to add incoming SIP call ID, outbound SIP call ID, and Call Session Identification to an EDNS0 OPT pseudo resource record in the ENUM DNS query.</td>
</tr>
</tbody>
</table>

Restrictions for ENUM Enhancement per Kaplan Draft RFC

- Supported only for SIP-to-SIP calls.
- The full command of voice enum-match-table, including the options, needs to be specified whenever being referenced by its subcommand. If not, the defaults, no source-uri and no no-cached (or caching) will take effect.
- As the maximum number of characters of the host shown in the show host command is 25, the source URI may not be displayed completely.
- The source URI is displayed in a separate line below, starting with “source-uri=”. Refer to the show command outputs in this chapter.
- If no-cache is configured in the voice enum-match-table, no cache table look-up would be made and hence an ENUM query would be made regardless of what is in the cache table.
- Both the target and source, where the source can be null/undefined or defined, need to be matched when looking up the cache table.
- The OPT RR will be added to the query for a SIP-to-SIP call only if the source-uri is configured for the outbound enum-match-table.
- The route will not be cached if the server does not support the OPT RR (it is recommended to remove the source-uri for this scenario if caching is preferred).
- The source URL can be prefixed with a host/target in the host name field in a double quote in the show host host command to display routes for the host specific with this source.
- A wild card, “*”, can be used to denote “all” hosts in the show host command. It can be by itself or any host matched with its prefix. The prefix can be a host name, partial or complete, or a domain name with partial or complete source URL.
Refer to the document titled *Unified Border Element ENUM Support Configuration Example* for a detailed message format.

**Information About ENUM Enhancement per Kaplan Draft RFC**

SIP-to-SIP calls can be routed based on the source SIP requests, using the ENUM enhancement feature. To provide source-based routing and to interact with Policy Server, an EDNS0 OPT pseudo resource record with source URI, incoming SIP call ID, outbound SIP call ID, and Call session Identification are added to the ENUM DNS query. The DNS server filters its response based on the source URI and call ID information and returns the appropriate NAPTR entries. To enable this feature, you must use the `source-uri` option in the `voice enum-match-table <table-number>` command. In addition, you can use the `no-cache` option to disable caching.

Refer to RFC 3761 and `draft-kaplan-enum-sip-routing-04` for more information about routing SIP requests with ENUM.

**How to Configure ENUM Enhancement per Kaplan Draft RFC**

**Enabling Source-Based Routing**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. `voice enum-match-table match-table-index [source-uri] [no-cache]`
4. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>voice enum-match-table match-table-index [source-uri] [no-cache]</code></td>
<td>Enables source URI filtering for the enum match table entry. You can use the <code>no-cache</code> option to disable the caching to the <code>voice enum</code> command.</td>
</tr>
<tr>
<td>Example: Device(config)# voice enum-match-table 5 source-uri no-cache</td>
<td></td>
</tr>
</tbody>
</table>

Cisco Unified Border Element Configuration Guide
Testing the ENUM Request

To test the ENUM request, you can use the source-url option so that the source-based routing enum can be tested.

SUMMARY STEPS

1. enable
2. test enum match-table-index input -pattern source-url source-url more parameter
3. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> test enum match-table-index input -pattern source-url source-url more parameter</td>
<td>Tests the source-based routing ENUM.</td>
</tr>
<tr>
<td>Example:</td>
<td>• The source routing or no caching features depend on the voice enum-match-table command. If the source-uri command is not configured, the source-url source-url in the test command is ignored.</td>
</tr>
<tr>
<td>Device# test enum 1117777 source sip:1116666@10.1.50.16 more &quot;ibcall-id=1-23735@10.1.50.16;obcall-id=7190DF-F1AA3CF1@10.1.110.222; sbc-id=1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# end</td>
<td></td>
</tr>
</tbody>
</table>

Verifying the ENUM Request

The following show commands can be used to verify the operation of the test command. If the no-cache option is enabled, the show host command does not display the enum entry. Some sample outputs of the show command are shown below. The show commands can be entered in any order.

SUMMARY STEPS

1. show host *
2. show host 1.0.9.3.e164-test*
3. show host 1*
4. show host "1.0.9.3.e164-test sip*"

DETAIL STEPS

Step 1: show host *

Example:

```
Device# show host *

Host Port Flags Age Type Address(es)
ns.e164-test None (temp, OK) 0 IP 127.0.0.1
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.1.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```

Step 2: show host 1.0.9.3.e164-test*

Example:

```
Device# show host 1.0.9.3.e164-test*

Host Port Flags Age Type Address(es)
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```

Step 3: show host 1*

Example:

```
Device# show host 1*

Host Port Flags Age Type Address(es)
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.1.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```

Step 4: show host "1.0.9.3.e164-test sip*"

Example:

```
Device# show host "1.0.9.3.e164-test sip*"

Host Port Flags Age Type Address(es)
ns.e164-test None (temp, OK) 0 IP 127.0.0.1
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.0.9.3.e164-test sip:540 NA (temp, OK) 0 NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```
Troubleshooting Tips

Use the following commands for debugging information:

- `debug voip enum detail`
- `debug ip domain`
- `debug ccsip message`
- `debug voip ccapi in/out`
- `clear voip fpi session correlator-id` — This command is used to clear the hung FPI sessions. After the hung session is identified using the existing `show` commands and its correlator is obtained, the `clear voip fpi session correlator-id` command can be used to clear the session.

Use the following `show` command that is helpful for debugging:

- `show host [all | * | host-name | partial-host-name*]`

Below is an extract of a sample ENUM DNS query containing the EDNS0 OPT psedor resource record fields as per Kaplan Draft that is helpful in debugging. In the below query the values corresponding to ibcall-id, obcall-id, and sbc-id represent the incoming SIP call ID, outbound SIP call ID and Call Session Identification respectively.

```
7.7.7.7.1.1.1.e164.arpa sip:1116666@10.1.50.16enum_dns_query: name = 7.7.7.7.1.1.1.e164.arpa sip:1116666@10.1.50.16 type = 35, ns_server = 0x0 no_cache 1 more_data
;ibcall-id=1-23735@10.1.50.16;
obcall-id=7190DF-39DD11E4-8008EDAD-F1AA3CF1@10.1.110.222;sbc-id=1
```

Configuration Examples for ENUM Enhancement per Kaplan Draft RFC

```
voice enum-match-table 1 source-uri // The source URI is sent to the DNS server to filter the route. //
   description enable source-uri
   rule 2 1 /^\(.\)*$/ /\1/ e164.arpa

voice enum-match-table 2 source-uri no-cache
rule 1 1 /^\(.\)*$/ /\1/ e164-test

voice enum-match-table 3 no-cache // The cache table is not looked up and the route is not cached. //
rule 1 1 /^\(.\)*$/ /\1/ e164-test

The following is a sample configuration for the ENUM enhancement feature:

dial-peer voice 1 voip
description ENUM Inbound dialpeer
session protocol sipv2
   incoming called-number 1116666

dial-peer voice 2 voip
```
description ENUM Outbound dialpeer
destination-pattern 1117777
session protocol sipv2
session target enum:1 //Session target configured to look up ENUM table 1.//
PART III

Multi-Tenancy

• Support for Multi-VRF, on page 335
• Configuring Multi-Tenants on SIP Trunks, on page 375
Support for Multi-VRF

The Virtual Routing and Forwarding (VRF) feature allows Cisco Unified Border Element (CUBE) to have multiple instances of routing and forwarding table to co-exist on the same device at the same time.

With Multi-VRF feature, each interface or subinterface can be associated with a unique VRF.

---

**Note**

The information in this chapter is specific to Multi-VRF feature beginning in Cisco IOS Release 15.6(2)T. However, there is some information on Voice-VRF feature for the reference purpose only. For detailed information on the Voice-VRF feature, see [http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t15/vrfawvgw.html](http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t15/vrfawvgw.html).

---

- Feature Information for VRF, on page 335
- Information About Voice-VRF, on page 336
- Information About Multi-VRF, on page 336
- VRF Preference Order, on page 337
- Restrictions, on page 337
- Recommendations, on page 338
- Configuring VRF, on page 338
- Configure VRF Specific RTP Port Ranges, on page 344
- Directory Number (DN) Overlap across Multiple-VRFs, on page 347
- IP Overlap with VRF, on page 349
- Using Server Groups with VRF, on page 351
- Inbound Dial-Peer Matching Based on Multi-VRF, on page 352
- VRF Aware DNS for SIP Calls, on page 354
- High Availability with VRF, on page 355
- Configuration Examples, on page 355

---

**Feature Information for VRF**

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.
Information About Voice-VRF

Support for Voice-VRF (also known as VRF-Aware) was introduced in Cisco IOS Release 12.4(11)XJ to provide support for configuring a VRF specific to voice traffic. Voice-VRF can be configured using `voice vrf vrf-name` command. For more information on voice-VRF, see [http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t15/vrfawvgw.html](http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t15/vrfawvgw.html).

Information About Multi-VRF

The Multi-VRF feature allows you to configure and maintain more than one instance of routing and forwarding tables within the same CUBE device and segregate voice traffic based on the VRF.

Multi-VRF uses input interfaces to distinguish calls for different VRFs and forms VRF tables by associating with one or more Layer 3 interfaces. Interface can be physical interface (such as FastEthernet ports, Gigabit Ethernet ports) or sub-interface. CUBE supports bridging calls on both intra-VRF and inter-VRF.

---

**Note**

One physical interface or sub-interface can be associated with one VRF only. One VRF can be associated with multiple interfaces.
As per the Multi-VRF feature, the dial-peer configuration must include the use of the interface bind functionality. This is mandatory. It allows dial-peers to be mapped to a VRF via the interface bind.

The calls received on a dial-peer are processed based on the interface to which it is associated with. The interface is in turn associated with the VRF. So, the calls are processed based on the VRF table associated with that particular interface.

**VRF Preference Order**

Voice-VRF and Multi-VRF configurations can co-exist. The following is the binding preference order for call processing:

**Table 43: VRF Preference Order and Recommendations**

<table>
<thead>
<tr>
<th>Preference Order</th>
<th>Bind</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dial-peer Bind</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>Tenant Bind</td>
<td>Recommended for SIP trunk, especially when Cisco UBE is collocated with Cisco Unified Survivability Remote Site Telephony. If Tenant bind is not configured, Voice-VRF is preferred for SIP trunk.</td>
</tr>
<tr>
<td>3</td>
<td>Global Bind</td>
<td>Not recommended.</td>
</tr>
<tr>
<td>4</td>
<td>Voice-VRF</td>
<td>Recommended for hosted and cloud services configurations when Cisco UBE is collocated with Cisco Unified Survivability Remote Site Telephony.</td>
</tr>
</tbody>
</table>

**Note**

Voice-VRF is not supported if Cisco Unified Survivability Remote Site Telephony is running on Cisco 4000 Series Integrated Services Router.

**Restrictions**

- Supports only SIP-SIP calls.
- Cisco Unified Communications Manager Express (Unified CME) and CUBE co-located with VRF is not supported.
- Cisco Unified Survivability Remote Site Telephony (Unified SRST) and CUBE co-location is not supported prior to Cisco IOS XE Fuji 16.7.1 Release.
- Multi-VRF call across CUBE is supported in Flow-through mode only.
- IPv6 on VRF is not supported.
- SDP pass-through is not supported prior to IOS Release 15.6(3)M and IOS XE Denali 16.3.1.
• Calls are not supported when incoming dial-peer matched is default dial-peer (dial-peer 0).
• Media Anti-trombone is not supported with VRF.
• Cisco UC Services API with VRF is not supported.
• Multi-VRF is not supported on TDM-SIP gateway.
• VRF aware matching is applicable only for inbound dial-peer matching and not for outbound dial-peer matching.
• Invoking TCL scripts via a dial-peer is not supported with the Multi-VRF feature.
• Multi-VRF using global routing table or default routing table (VRF 0) with virtual interfaces is not supported on ISR-G2 (2900 and 3900 series) routers.
• SCCP based media resources are not supported with VRF feature.

**Recommendations**

• For new deployments, we recommend a reboot of the router once all VRFs' are configured under interfaces.
• No VRF Route leaks are required on CUBE to bridge VoIP calls across different VRFs.
• High Availability(HA) with VRF is supported where VRF IDs are check-pointed in the event of fail-over. Ensure that same VRF configuration exists in both the HA boxes.
• Whenever destination server group is used with VRF, ensure that the server group should have the session targets, belonging to the same network as that of sip bind on the dial-peer, where the server-group is configured. This is because, dial-peer bind is mandatory with VRF and only one sip bind can be configured on any given dial-peer.
• If there are no VRF configuration changes at interface level, then reload of the router is not required.

**Configuring VRF**

---

**Note**

We recommend you NOT to modify VRF settings on the interfaces in a live network as it requires CUBE reload to resume VRF functionality.

This section provides the generic configuration steps for creating a VRF. For detailed configuration steps specific to your network scenario (Multi-VRF and Multi-VRF with HA), refer to Configuration Examples section.
You can also use the latest configuration option, which allows creation of multiprotocol VRFs that support both IPv4 and IPv6. Entering the command `vrf definition vrf-name` creates the multiprotocol VRF. Under VRF definition submode, you can use the command `address-family {ipv4 | ipv6}` to specify appropriate address family. To associate the VRF with an interface, use the command `vrf forwarding vrf-name` under the interface configuration submode.

For more information about the `vrf definition` and `vrf forwarding` commands, refer to the Cisco IOS Easy Virtual Network Command Reference Guide.

---

### Create a VRF

**SUMMARY STEPS**

1. **enable**
2. **configure terminal**
3. `ip vrf vrf-name`
4. `rd route-distinguisher`
5. **exit**

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **enable** | **Example:** Device> enable | Enables privileged EXEC mode  
  • Enter your password if prompted. |
| **Step 2** | **configure terminal** | Enters global configuration mode. |
| **Example:** | Device# configure terminal | |
| **Step 3** | `ip vrf vrf-name` | Creates a VRF with the specified name. In the example, VRF name is VRF1.  
  **Note**  
  Space is not allowed in VRF name. |
| **Example:** | Device(config)# ip vrf VRF1 | |
| **Step 4** | `rd route-distinguisher` | Creates a VRF table by specifying a route distinguisher. Enter either an AS number and an arbitrary number (xxx:y) or an IP address and arbitrary number (A.B.C.D:y) |
| **Example:** | Device(config)# rd 1:1 | |
| **Step 5** | **exit** | Exits present mode. |
| **Example:** | Device(config)# exit | |
Assign Interface to VRF

If an IP address is already assigned to an interface, then associating a VRF with interface will disable the interface and remove the existing IP address. An error message (sample error message shown below) is displayed on the console. Assign the IP address to proceed further.

% Interface GigabitEthernet0/1 IPv4 disabled and address(es) removed due to enabling VRF VRF1

SUMMARY STEPS

1. enable
2. configure terminal
3. interface interface-name
4. ip vrf forwarding vrf-name
5. ip address ip address subnet mask
6. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> interface interface-name</td>
<td>Enters the interface configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# interface GigabitEthernet 0/1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ip vrf forwarding vrf-name</td>
<td>Associates VRF with the interface.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-if)# ip vrf forwarding VRF1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> ip address ip address subnet mask</td>
<td>IP address is assigned to the interface.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-if)# ip address 10.0.0.1 255.255.255.0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits present mode.</td>
</tr>
</tbody>
</table>
Create Dial-peers

SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `dial-peer voice number voip`
4. `session protocol protocol`
5. Create dial-peer:
   - To create inbound dial-peer:
     `incoming called number number`
   - To create outbound dial-peer:
     `destination pattern number`
6. `codec codec-name`
7. `exit`

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>dial-peer voice number voip</code></td>
<td>Creates the dial-peer with the specified number.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(config)# dial-peer voice 1111 voip</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>session protocol protocol</code></td>
<td>Specifies the protocol associated with the dial-peer.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(config-dial-peer)# session protocol sipv2</code></td>
<td></td>
</tr>
</tbody>
</table>
| Step 5 | Create dial-peer:  
  - To create inbound dial-peer:  
    `incoming called number number` | Creates inbound and outbound dial-peer. |
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• To create outbound dial-peer:</td>
<td></td>
</tr>
<tr>
<td>destination pattern number</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Inbound dial-peer:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)#</td>
<td></td>
</tr>
<tr>
<td>incoming called-number 1111</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Outbound dial-peer:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)#</td>
<td></td>
</tr>
<tr>
<td>destination pattern 3333</td>
<td></td>
</tr>
</tbody>
</table>

**Step 6**

**codec codec-name**

**Example:**

Device(config-dial-peer)# codec g711ulaw

**Step 7**

**exit**

**Example:**

Device(config-dial-peer)# exit

### Bind Dial-peers

You can configure SIP binding at global level as well as at dial-peer level.

- Control and Media on a dial-peer have to bind with same VRF. Else, while configuring, the CLI parser will display an error.
- Whenever global sip bind interface associated with a VRF is added, modified, or removed, you should restart the sip services under 'voice service voip > sip' mode so that the change in global sip bind comes into effect with associated VRF ID.

```
CUBE(config)# voice service voip
CUBE(config-voi-serv)# sip
CUBE(config-serv-sip)# call service stop
CUBE(config-serv-sip)# no call service stop
CUBE(config-serv-sip)# end
```

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **Bind control and media to the interface**
   - At dial-peer level:
     
     **dial-peer voice number voip**
voice-class sip bind control source-interface interface-name
voice-class sip bind media source-interface interface-name

• At global configuration level
voice service voip
sip
bind control source-interface interface-name
bind media source-interface interface-name

4. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| enable | Example: Device> enable | Enables privileged EXEC mode
• Enter your password if prompted. |

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure terminal</td>
<td>Example: Device# configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bind control and media to the interface</td>
<td></td>
<td>Interface bind associates VRF to the specified dial-peer.</td>
</tr>
</tbody>
</table>
• At dial-peer level:
dial-peer voice number voip
voice-class sip bind control source-interface interface-name
voice-class sip bind media source-interface interface-name
• At global configuration level
voice service voip
sip
bind control source-interface interface-name
bind media source-interface interface-name

Example:
At dial-peer level:
Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config-dial-peer)# voice-class sip bind
Configure VRF Specific RTP Port Ranges

You can configure each VRF to have its own set of RTP port range for VoIP RTP connections under voice service voip. A maximum of ten VRF port ranges are supported. Different VRFs can have overlapping RTP port range. VRF based RTP port range limits (min, max port numbers) are same as global RTP port range. All three port ranges (global, media-address, VRF based) can coexist on CUBE and the preference order of RTP port allocation is as follows:

- VRF based port range
- Media-address based port range
- Global RTP port range

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `voice service voip`
4. `media-address voice-vrf vrf-name port-range min max`
5. `exit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# exit</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device(config)# sip</td>
<td></td>
</tr>
<tr>
<td>Device(config)# bind control source-interface GigabitEthernet0/1</td>
<td></td>
</tr>
<tr>
<td>Device(config)# bind media source-interface GigabitEthernet0/1</td>
<td></td>
</tr>
</tbody>
</table>
### Configure VRF Specific RTP Port Ranges

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 2** | configure terminal  
**Example:**  
Device# configure terminal | Enters global configuration mode. |
| **Step 3** | voice service voip  
**Example:**  
Device(config)# voice service voip | Enters voice service voip mode. |
| **Step 4** | media-address voice-vrf vrf-name port-range min max  
**Example:**  
Example 1  
Device(conf-voi-serv)# media-address voice-vrf VRF1 port 16000 32000  
*Here, the port-range is configured on the same line as the media address.*  
Example 2  
Device(conf-voi-serv)# media-address voice-vrf VRF1 port-range 6000 7000  
Device(conf-voi-serv)# port-range 8000 10000  
Device(conf-voi-serv)# port-range 11000 20000  
*In this case, multiple port range lines are configured under the media address.*  
**Example:**  
Device(conf-voi-serv)# media-address voice-vrf VRF1 port-range 8000 48000 | Associates the RTP Port range with the VRF.  
If the RTP port range is not configured per each VRF, the default RTP port range is used across the VRFs used. You can configure up to ten port ranges per media address.  
The default port range is **8000-48198** for ASR and ISR G3 platforms, and **16384-32766** for ISR G2 platforms.  
**Note** From Cisco IOS 15.6(3)M and Cisco IOS XE Denali Release 16.3.1 onwards, you can configure 54 VRFs for up to 10 different RTP port ranges (that is, 10 different port ranges per each VRF). |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>media-address voice-vrf VRF2 port-range 8000 48000</td>
<td></td>
</tr>
<tr>
<td>media-address voice-vrf VRF53 port-range 8000 48000</td>
<td></td>
</tr>
<tr>
<td>media-address voice-vrf VRF54 port-range 8000 48000</td>
<td></td>
</tr>
</tbody>
</table>

**Step 5**

**Example:**

Device(conf-voi-serv)# exit

Exits present mode.

---

**Example: VRF with overlapping and non-overlapping RTP Port Range**

**Example 1 - Non-overlapping Port Range**

The following is an example that shows two VRFs with non-overlapping RTP port range:

```
Device(config)# voice service voip
Device(config-voi-serv)# no ip address trusted authenticate
Device(config-voi-serv)# media bulk-stats
Device(config-voi-serv)# media-address voice-vrf vrf1 port-range 25000 28000
Device(config-voi-serv)# media-address voice-vrf vrf2 port-range 29000 32000
Device(config-voi-serv)# allow-connections sip to sip
Device(config-voi-serv)# redundancy-group 1
Device(config-voi-serv)# sip
```

The output for command `show voip rtp connections` shows as follows:

```
Device(config)# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 23001, Ports Reserved: 101, Ports in Use: 2

<table>
<thead>
<tr>
<th>Ports</th>
<th>Min</th>
<th>Max</th>
<th>Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media Pool</td>
<td>8000</td>
<td>48198</td>
<td>19999</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>VRF ID Based Media Pool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vrf1</td>
<td>25000</td>
<td>28000</td>
<td>1501</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>vrf2</td>
<td>29000</td>
<td>32000</td>
<td>1501</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

VoIP RTP active connections:

```
No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP MPSS VRF
1  1001 1002 25000 16400 10.0.0.1 10.0.0.2 NO vrf1
2  1002 1001 29000 16392 11.0.0.1 11.0.0.2 NO vrf2
```

In the above output, you can observe that for both the VRF's having non-overlapping rtp port ranges, the local RTP port allocated for vrf1 and vrf2 are different.

**Example 2 - Overlapping Port Range**

The following is an example that shows two VRFs with overlapping RTP port range:
Device(conf)# voice service voip
Device(conf-voi-serv)# no ip address trusted authenticate
Device(conf-voi-serv)# media bulk-stats
Device(conf-voi-serv)# media-address voice-vrf vrf1 port-range 25000 28000
Device(conf-voi-serv)# media-address voice-vrf vrf2 port-range 25000 28000
Device(conf-voi-serv)# allow-connections sip to sip
Device(conf-voi-serv)# redundancy-group 1
Device(conf-voi-serv)# sip

The output for command `show voip rtp connections` shows as follows:

```
Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 23001, Ports Reserved: 101, Ports in Use: 2

<table>
<thead>
<tr>
<th>Ports</th>
<th>Media-Address Range</th>
<th>Port</th>
<th>Port</th>
<th>Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>8000</td>
<td>48198</td>
<td>19999</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vrf1</td>
<td>25000</td>
<td>28000</td>
<td>1501</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vrf2</td>
<td>25000</td>
<td>28000</td>
<td>1501</td>
<td>0</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No.</th>
<th>CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
<th>MPSS</th>
<th>VRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001</td>
<td>1002</td>
<td>25000</td>
<td>16400</td>
<td>10.0.0.1</td>
<td>10.0.0.2</td>
<td>NO</td>
<td>vrf1</td>
</tr>
<tr>
<td>2</td>
<td>1002</td>
<td>1001</td>
<td>25000</td>
<td>16392</td>
<td>11.0.0.1</td>
<td>11.0.0.2</td>
<td>NO</td>
<td>vrf2</td>
</tr>
</tbody>
</table>
```

In the above output, you can observe that for both the VRF’s having overlapping rtp port ranges, the local RTP port allocated for vrf1 and vrf2 is same.

**Directory Number (DN) Overlap across Multiple-VRFs**

CUBE has the capability to bridge calls across VRFs without the need for route leaks to be configured.

If multiple dial-peers on two different VRFs have the same destination-pattern and preference, CUBE will randomly choose a dial-peer and route the call using the session target of the selected dial-peer. Due to this, the call intended for one VRF may be routed to another VRF.

Dial-peer group feature allows you to route calls within the same VRF and not across VRFs. Configuring dial-peer group, routes the call to a specific VRF even if multiple dial-peers on two different VRFs have the same destination-pattern and preference.

To use dial-peer group feature, configure dial-peers such that there is a unique inbound dial-peer match for calls related to each VRF. Configuring dial-peer group, limits the outbound dial-peer search within the VRF.
Example: Associating Dial-peer Groups to Overcome DN Overlap

If a call is received on VRF1 and there are two dial-peers with same destination-pattern (one dial-peer bind to VRF1 and second dial-peer bind to VRF2), then by default, CUBE picks the VRF in random to route the call.

If you intended to route this call only to VRF1 dial-peer, then dial-peer group can be applied on inbound dial-peer which will restrict the CUBE to route the call only across the dial-peers within the dial-peer group and not pick a dial-peer bind to a different VRF.

Figure 34: Associating Dial-peer Group to overcome DN overlap

The following scenario is considered in the below example:

- VRF1 associated with Gigabitethernet Interface 0/0 and 0/1
- VRF 2 associated with Gigabitethernet Interface 0/2
- Dial-peer Group: dpg1
- VRF1 is associated with dial-peer group - dpg 1
- Outbound dial-peer 300 is selected as preference 1
- Inbound dial-peer 3000 associated with VRF 1 and dial-peer group 1 (dpg1)
- Outbound Dial-peer: 300 – destination pattern “3001” associated with VRF1
- Outbound dial-peer: 301 – destination pattern “3001” associated with VRF2

Configure a dial-peer group and set the outbound dial-peer preference.

```
Device# enable
Device# configure terminal
Device(config)# voice class dpg 1
Device(config)# dial-peer 300 preference 1
```

Create inbound dial-peer and associated with dial-peer group 1 (dpg1)

```
Device(config)# dial-peer voice 3000 voip
Device(config-dial-peer)# video codec h264
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session transport udp

Device(config-dial-peer)# destination dpg 1
Device(config-dial-peer)# incoming called-number 3001
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
```
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1
Device(config-dial-peer)# dtmf-relay sip-kpml
Device(config-dial-peer)# srtp fallback
Device(config-dial-peer)# codec g711ulaw

Creating outbound dial-peer with destination pattern ‘3001’ associated with VRF1.

Device(config)# dial-peer voice 300 voip
Device(config-dial-peer)# destination-pattern 3001
Device(config-dial-peer)# video codec h264
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.0.0.1
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1
Device(config-dial-peer)# dtmf-relay sip-kpml
Device(config-dial-peer)# codec g711ulaw

Creating outbound dial-peer with destination pattern ‘3001’ associated with VRF2.

Device(config)# dial-peer voice 301 voip
Device(config-dial-peer)# destination-pattern 3001
Device(config-dial-peer)# video codec h264
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:11.0.0.1
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/2
Device(config-dial-peer)# dtmf-relay sip-kpml
Device(config-dial-peer)# codec g711ulaw

With above dial-peer group configuration, whenever dial-peer “3000” is matched as inbound dial-peer, CUBE will always route call using dial-peer “300” (VRF1). Without dial-peer group, CUBE would have picked dial-peers “300”(VRF1) and “301”(VRF2) in random to route the call.

Device# show vrf brief

<table>
<thead>
<tr>
<th>Name</th>
<th>Default RD</th>
<th>Protocols</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRF1</td>
<td>1:1</td>
<td>ipv4</td>
<td>Gi0/0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gi0/1</td>
</tr>
<tr>
<td>VRF2</td>
<td>2:2</td>
<td>ipv4</td>
<td>Gi0/2</td>
</tr>
</tbody>
</table>

Device# show dial-peer voice summary
dial-peer hunt 0

<table>
<thead>
<tr>
<th>TAG</th>
<th>TYPE</th>
<th>MIN</th>
<th>OPER</th>
<th>PREFIX</th>
<th>DEST-PATTERN</th>
<th>PRE</th>
<th>PASS</th>
<th>THRU</th>
<th>SESS-TARGET</th>
<th>OUT</th>
<th>STAT</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>3001</td>
<td>0</td>
<td>syst</td>
<td>ipv4:10.0.0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>3001</td>
<td>0</td>
<td>syst</td>
<td>ipv4: 10.0.0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>301</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>3001</td>
<td>0</td>
<td>syst</td>
<td>ipv4: 11.0.0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IP Overlap with VRF**

Generally, on a router, two interfaces cannot be configured with the same IP address. With the VRF feature, you can configure two or more interfaces with the same IP address because, each interface having the same IP address belongs to a unique VRF and hence belongs to a different routing domain. However, for successful call processing, you must ensure that appropriate call routing protocols are configured on the VRFs.
The following is a sample configuration:

Configure Gigabit Ethernet 0/0 that belongs to VRF1 with IP address 10.0.0.0.

Device# enable
Device# configure terminal
Device(config)# ip vrf VRF1
Device(config)# rd 1:1
Device(config)# exit

Device# enable
Device# configure terminal
Device(config)# interface GigabitEthernet0/0
Device(config-if)# ip vrf forwarding VRF1
Device(config-if)# ip address 10.0.0.0 255.255.255.0
Device(config-if)# speed auto
Device(config-if)# exit

Configure Gigabit Ethernet 0/1 that belongs to VRF2 with IP address 10.0.0.0.

Device# enable
Device# configure terminal
Device(config)# ip vrf VRF2
Device(config)# rd 1:1
Device(config)# exit

Device# enable
Device# configure terminal
Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip vrf forwarding VRF2
Device(config-if)# ip address 10.0.0.0 255.255.255.0
Device(config-if)# speed auto
Device(config-if)# exit

For call routing on VRF1 and VRF2, ensure that appropriate routing entries are configured for both VRF1 and VRF2.

---

**Note**

Even though Gigabit Ethernet 0/0 and Gigabit Ethernet 0/1 have an overlapping IP address, the call processing is not overlapped as they belong to different VRFs.

**show ip interface brief** command shows that GigabitEthernet 0/0 and GigabitEthernet 0/1 have an overlapping IP address:

<table>
<thead>
<tr>
<th>Interface</th>
<th>IP-Address</th>
<th>OK? Method Status</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded-Service-Engine/0</td>
<td>unassigned</td>
<td>YES NVRAM</td>
<td>admin down down</td>
</tr>
<tr>
<td>GigabitEthernet0/0</td>
<td>10.0.0.0</td>
<td>YES NVRAM</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/0/1</td>
<td>10.0.0.0</td>
<td>YES NVRAM</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/0/1.1</td>
<td>unassigned</td>
<td>YES NVRAM</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/2</td>
<td>unassigned</td>
<td>YES NVRAM</td>
<td>up</td>
</tr>
</tbody>
</table>
**Using Server Groups with VRF**

Whenever destination server group is used with VRF, ensure that the server group should have the session targets, belonging to the same network as that of sip bind on the dial-peer, where the server-group is configured. This is because the dial-peer bind is mandatory with VRF and only one sip bind can be configured on any given dial-peer.

The following scenario is considered in the below example:

Interaces and associated IP address

- GigabitEthernet0/0/2 12.0.0.1
- GigabitEthernet0/0/1 11.0.0.1

```
Device# show ip interface brief
Interface IP-Address OK? Method Status Protocol
GigabitEthernet0/0/0 10.0.0.1 YES NVRAM up up
GigabitEthernet0/0/1 11.0.0.1 YES NVRAM up up
GigabitEthernet0/0/2 12.0.0.1 YES NVRAM up up
```

- dial-peer 200 is bind to GigabitEthernet0/0/1
- server-group 1 (belonging to VRF1) is applied to dial-peer 200

```
Device(config)# dial-peer voice 200 voip
Device(config-dialpeer)# destination-pattern 4.....
Device(config-dialpeer)# session protocol sipv2
Device(config-dialpeer)# session transport udp
Device(config-dialpeer)# session server-group 1
Device(config-dialpeer)# voice-class sip bind control source-interface GigabitEthernet0/0/1
Device(config-dialpeer)# voice-class sip bind media source-interface GigabitEthernet0/0/1
```
Inbound Dial-Peer Matching Based on Multi-VRF

From Cisco IOS Release 15.6(3)M and Cisco IOS XE Denali 16.3.1 onwards, dial-peer matching is done based on the VRF ID associated with a particular interface.

Example: Inbound Dial-Peer Matching based on Multi-VRF

Prior to Cisco IOS 15.6(3)M and Cisco IOS XE Denali 16.3.1 releases, when an incoming out-of-dialog message such as INVITE, REGISTER, OPTIONS, NOTIFY, and so on are received on a particular VRF bound interface, inbound dial-peer matching was done using the complete set of inbound dial-peers regardless of the VRF association. The response would be sent based on this matched dial-peer. Since the inbound dial-peer selected could have a different VRF bound to it, the response was sent to the wrong VRF.

To overcome this issue, the inbound dial-peers are filtered based on the incoming VRF and then followed by the regular inbound dial-peer matching. Now, the response is sent to the same VRF on which the request was received.

Consider the following configuration example output to understand the inbound dial-peer matching criteria used in multi-VRF:

```
interface GigabitEthernet0/0
   ip address 8.39.18.37 255.255.0.0
   duplex auto
   ip vrf forwarding VRF ID1
   speed auto

interface GigabitEthernet0/1
   ip address 9.39.18.55 255.255.0.0
   duplex auto
   ip vrf forwarding VRF ID2
   speed auto

interface GigabitEthernet0/2
   ip address 10.39.18.68 255.255.0.0
   duplex auto
   ip vrf forwarding VRF ID3
   speed auto

dial-peer voice 1000 voip
description “Inbound dial-peer bound to VRF ID2”
session protocol sipv2
session target sip-server
session transport udp
incoming called-number 5678
voice-class sip bind control source-interface GigabitEthernet0/1
voice-class sip bind media source-interface GigabitEthernet0/1
```
codec g711ulaw

dial-peer voice 2000 voip
description “Inbound dial-peer bound to VRF ID1”
session protocol sipv2
session target sip-server
session transport udp
incoming called-number 5678
voice-class sip bind control source-interface GigabitEthernet0/0
voice-class sip bind media source-interface GigabitEthernet0/0
codec g711ulaw

dial-peer voice 3000 voip
description “Inbound dial-peer bound to VRF ID3”
session protocol sipv2
session target sip-server
session transport udp
incoming called-number 8000
voice-class sip bind control source-interface GigabitEthernet0/2
voice-class sip bind media source-interface GigabitEthernet0/2
codec g711ulaw

dial-peer voice 4000 voip
description “Inbound dial-peer bound to VRF ID1”
session protocol sipv2
session target sip-server
session transport udp
incoming called-number 2000
voice-class sip bind control source-interface GigabitEthernet0/0
voice-class sip bind media source-interface GigabitEthernet0/0
codec g711ulaw

Prior to Cisco IOS 15.6(3)M and Cisco IOS XE Denali 16.3.1 releases, when an incoming call is received for the dialed number 5678 on GigabitEthernet0/0 (VRF ID1), inbound dial-peer matching was done based on the called-number 5678. In this case, dial-peer 1000 which is bound to GigabitEthernet0/1 (VRF ID2) was considered to be the first matched dial-peer for this call. And, the response was sent incorrectly to VRF ID2 instead of VRF ID1.

With the introduction of VRF aware inbound dial-peer matching, the initial filtering is done based on the VRF ID and then based on the called-number. For the above example, a call with called-number of 5678 that is received on GigabitEthernet 0/0 with VRF ID 1 configured, the dial-peers will first be filtered to those that are bound to GigabitEthernet 0/0 before selection of the inbound dial-peer is performed. Now, the response is sent successfully on VRF ID1.

Whenever the VRF ID is added, modified, or removed under the interface, it is mandatory to execute the following command before making any calls: clear interface <interface>. If the clear interface <Interface> command is not executed, the dial-peer is bound to the old VRF ID and not to the new VRF ID.
Inbound dial-peer matching based on VRF ID is selected in the following order of preference:

1. Dial-peer based configuration
2. Tenant based configuration
3. Global based configuration

Example: Tenant based Inbound Dial-Peer Matching

```plaintext
voice class tenant 1
  bind control source-interface GigabitEthernet0/0
  bind media source-interface GigabitEthernet0/0
  dial-peer voice 2000 voip
    description “Inbound dial-peer bound to VRF-ID 1”
    session protocol sipv2
    session target sip-server
    session transport udp
    incoming called-number 5678
    voice-class sip tenant 1
    codec g711ulaw
```

Example: Global based Inbound Dial-Peer Matching

```plaintext
voice service voip
  sip
    bind control source-interface GigabitEthernet0/0
    bind media source-interface GigabitEthernet0/0
```

VRF Aware DNS for SIP Calls

The VRF Aware DNS for SIP Calls feature enables you to specify the Virtual Routing and Forwarding (VRF) table so that the domain name system (DNS) can forward queries to name servers using the VRF table.

Because the same IP address can be associated with different DNS servers in different VRF domains, a separate list of name caches for each VRF is maintained. The DNS looks up the specific VRF name cache before sending a query to the VRF name server. All IP addresses obtained from a VRF-specific name cache are routed using the VRF table.

While processing a SIP call, if a hostname has to be resolved, only the VRF associated with the SIP call is used during DNS resolutions.

Note: Ensure that the name-server is configured using `ip name-server vrf` command. For configuration details, see Name Server Configuration.
High Availability with VRF

CUBE supports VRF in both HSRP and RG Infra high availability mode. VRF is supported on CUBE box-to-box and inbox high availability types.

For box-to-box high availability in Aggregation Services Routers 1000 Series and Integrated Services Routers 4000 Series, RG interface must not be associated with VRF where as the inbound and outbound interfaces (meant for handling VoIP traffic) can be associated with VRF’s depending upon the deployment.

For box-to-box high availability in Integrated Services Routers Generation 2, HSRP interface must not be associated with VRF where as the inbound and outbound interfaces (meant for handling VoIP traffic) can be associated with VRFs depending upon the deployment.

All the configurations including the VRF based RTP port range has to be identical on active and standby routers. VRF IDs will be checkpointed before and after the switchover.

Configuration Examples

Note

The steps in the following configuration example is for a new network and hence it is assumed that there is no existing configuration.

Example: Configuring Multi-VRF in Standalone Mode

The configuration in this scenario is as shown below where the Gigabitethernet 0/1 is assigned to VRF1 and GigabitEthernet 0/2 is assigned to VRF2.

Figure 35: Multi-VRF in Standalone Mode

Configuring VRF

Device# enable
Device# configure terminal
Device(config)# ip vrf VRF1
Device(config)# rd 1:1
Device(config)# ip vrf VRF2
Device(config)# rd 2:2
Device(config)# exit

Associating interfaces with VRF
Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip vrf forwarding VRF1
Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2

If an IP address is already assigned to an interface, then associating a VRF with interface will disable the interface and remove the existing IP address. An error message (sample error message shown below) is displayed on the console. Assign the IP address to proceed further.

% Interface GigabitEthernet0/1 IPv4 disabled and address(es) removed due to enabling VRF VRF1

Configure Interface GigabitEthernet0/1

Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip address 10.0.0.2 255.255.255.0
Device(config-if)# speed auto
Device(config-if)# exit

Configure Interface GigabitEthernet0/2

Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip address 11.0.0.2 255.255.255.0
Device(config-if)# speed auto
Device(config-if)# exit

Creating Dial-peer

Creating Inbound Dial-peer:

Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# incoming called-number 1111
Device(config-dial-peer)# codec g711ulaw

Creating Outbound Dial-peer:

Device(config)# dial-peer voice 2222 voip
Device(config-dial-peer)# destination pattern 1111
Device(config-dial-peer)# session protocol sipv2

Execute the following command to verify the dial-peer association with interface:

Device# show dial-peer voice summary

<table>
<thead>
<tr>
<th>TAG</th>
<th>TYPE</th>
<th>MIN</th>
<th>OPER</th>
<th>PREFIX</th>
<th>DEST-PATTERN</th>
<th>FER</th>
<th>THRU</th>
<th>SESS-TARGET</th>
<th>STAT</th>
<th>PORT</th>
<th>KEEPALIVE</th>
<th>VRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td>-</td>
<td></td>
<td>0</td>
<td>syst</td>
<td>ipv4:10.0.0.2</td>
<td></td>
<td></td>
<td></td>
<td>VRF</td>
</tr>
<tr>
<td>VRF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2222</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td>-</td>
<td></td>
<td>0</td>
<td>syst</td>
<td>ipv4:11.0.0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VRF2

Configure Binding

- Control and Media on a dial-peer have to bind with same VRF. Else, while configuring, the CLI parser will display an error.
- Whenever global sip bind interface associated with a VRF is added, modified, or removed, you should restart the sip services under voice service voip sip mode so that the change in global sip bind comes into effect with associated VRF ID.

```
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# call service stop
Device(conf-serv-sip)# no call service stop
Device(conf-serv-sip)# end
```

```
Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1
```

```
Device(config)# dial-peer voice 2222 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/2
```

Execute the following command to verify the interface association with VRF:

```
Device# show ip vrf brief
```

```
Name          Default  RD  Interfaces
Mgmt-intf     <not set> Gi0
VRF1          1:1       Gi0/1
VRF2          2:2       Gi0/2
```

Execute the following command to verify a successful and active calls:

For a single call, you should be able to see two RTP connections as shown in the below example.

```
Device# show voip rtp connections
```

```
VoIP RTP Port Usage Information:
Max Ports Available: 23001, Ports Reserved: 101, Ports in Use: 2
Media-Address Range.................................................
------------------------------------------------------------------------------
Global Media Pool                         8000  48198  19999       101  0
------------------------------------------------------------------------------
VoIP RTP active connections :
No.   Callld dstCallld LocalRTP  RmtRTP  LocalIP  RemoteIP  MPSS  VRF
 1     1     2      25000       16390  10.0.0.1  10.0.0.2  NO  VRF1
 2     2     1      25002       16398  11.0.0.1  11.0.0.2  NO  VRF2
```
Device# show call active voice brief

Perf-AR1006# show call active voice brief
[ID]: <CallID> <start>ms.<index> (<start>) +<connect> pid:<peer_id> <dir> <addr> <state>
  dur hh:mm:ss tx:<packets>/<bytes> rx:<packets>/<bytes> dscp:<packets violation>
 media:<packets violation> audio tos:<audio tos value> video tos:<video tos value>
 IP <ip>:<udp> rtt:<time>ms pl:<play>/gap<ms lost:<lost>/early/late>
delay:<last>/min/<max>ms <codec> <textrelay> <transcoded>
media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>
long duration call detected:<y/n> long duration call duration :<sec> timestamp:<time>
LostPacketRate:% OutOfOrderRate:%
VRP:<%
  MODEMPASS <method> buf:<fills>/drains> loss <overall>% <multipkt>/<corrected>
    last <buf event time> dur:<Min>/<Max>ms
  FR <protocol> [int dlc|cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>
long duration call detected:<y/n> long duration call duration :<sec> timestamp:<time>
LostPacketRate:% OutOfOrderRate:%
VRP:<%
  MODEMPASS <method> buf:<fills>/drains> loss <overall>% <multipkt>/<corrected>
    last <buf event time> dur:<Min>/<Max>ms
  FR <protocol> [int dlc|cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>

Telephony call-legs: 0
SIP call-legs: 2
H233 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2
Answer 777412373 active
dur 00:00:22 tx:<audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
dscp:0 media:0 audio tos:<0x0> video tos:<0x0>
IP 10.0.0.2:30804 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay: off
Transcoded: No ICE: Off
media inactive detected:<n media cntrl rcvd:<n a timestamp:<n/a>
long duration call detected:<n/a> long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00
VRF: VRF1

11FF : 8565723 511605470ms.1 (*16:21:53.696 IST Tue Aug 4 2015) +0 pid:400000 Originate
777512373 active
dur 00:00:22 tx:<audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
dscp:0 media:0 audio tos:<0x0> video tos:<0x0>
IP 11.0.0.2:30804 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay: off
Transcoded: No ICE: Off
media inactive detected:<n media cntrl rcvd:<n/a> timestamp:n/a>
long duration call detected:<n/a> long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00
VRF: VRF2

Telephony call-legs: 0
SIP call-legs: 2
H233 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2
Device# show sip-ua connections udp brief

Total active connections : 2
No. of send failures    : 0
No. of remote closures  : 0
No. of conn. failures   : 0
No. of inactive conn. ageouts : 2

-------------- SIP Transport Layer Listen Sockets ---------------
 Conn-Id Local-Address
 =========== ============== ================
      2       [10.0.0.1]:5060:VRF1
      3       [11.0.0.1]:5060:VRF2

Device# show call active voice compact

<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp> VRF
Total call-legs: 2
 8565722  ANS  T12  g711ulaw VOIP P777412373 10.0.0.2:30804 VRF1
 8565723  ORG  T12  g711ulaw VOIP P777512373 11.0.0.2:30804 VRF2

Device# show call active video compact

MVRF-CUBE1# show call active video compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp> VRF
Total call-legs: 2
 10193983  ANS  T30  H264 VOIP-VIDEO P2005 10.0.0.2:18078 VRF1
 10193985  ORG  T30  H264 VOIP-VIDEO P3001 11.0.0.2:27042 VRF2

Example: Configuring RG Infra High Availability with VRF

---

**Note**

Below configuration example is applicable for Cisco ASR 1000 Series Aggregated Services Routers (ASR) and Cisco 4000 Series Integrated Services Routers (ISR G3).

---

**Note**

Do not configure VRF on the interface that is used for RG Infra. Traffic of VRF and RG Infra should be on different interfaces.
Configuration on Active Router

The configurations of Active Router and Stand By Router should be identical.

Configuring VRF

Device> enable
Device# configure terminal
Device(config)# ip vrf VRF1
Device(config)# rd 1:1
Device(config)# ip vrf VRF2
Device(config)# rd 2:2
Device(config)# voice service voip
Device(config)# no ip address trusted authenticate
Device(config)# media bulk-stats
Device(config)# allow-connections sip to sip
Device(config)# redundancy-group 1
Device(config)# sip
Device(config)# redundancy
Device(config)# mode none
Device(config)# application redundancy
Device(config)# group 1
Device(config)# name raf-b2b
Device(config)# priority 1
Device(config)# timers delay 30 reload 60
Device(config)# control GigabitEthernet0/0/0 protocol 1
Device(config)# data GigabitEthernet0/0/0

Associating interfaces with VRF

Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding vrf2
If an IP address is already assigned to an interface, then associating a VRF with interface will disable the interface and remove the existing IP address. An error message (sample error message shown below) is displayed on the console. Assign the IP address to proceed further.

% Interface GigabitEthernet0/1 IPv4 disabled and address(es) removed due to enabling VRF VRF1

GigabitEthernet0/0/0 is used for configuring RG Infra and therefore do not configure any VRF with this interface.

Device(config)# interface GigabitEthernet0/0/0
Device(config-if)# ip address 14.2.43.81 255.255.0.0
Device(config-if)# negotiation auto
Device(config-if)# cdp enable

Inbound interface - GigabitEthernet0/1 is used for voice traffic configured with VRF1.

Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip vrf forwarding VRF1
Device(config-if)# ip address 10.0.0.3 255.0.0.0
Device(config-if)# negotiation auto
Device(config-if)# cdp enable
Device(config-if)# redundancy rii 1
Device(config-if)# redundancy group 1 ip 10.0.0.1 exclusive

Outbound interface - GigabitEthernet0/2 is used for voice traffic configured with VRF2.

Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2
Device(config-if)# ip address 11.0.0.3 255.0.0.0
Device(config-if)# negotiation auto
Device(config-if)# cdp enable
Device(config-if)# redundancy rii 2
Device(config-if)# redundancy group 1 ip 11.0.0.1 exclusive

Creating Dial-peer

Creating Inbound Dial-peer:

Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# destination pattern 1111
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.0.0.2
Device(config-dial-peer)# incoming called-number 1111

Creating Outbound Dial-peer:

Device(config)# dial-peer voice 3333 voip
Device(config)# destination-pattern 2222
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:11.0.0.2
Configuring Binding

Control and Media on a dial-peer have to bind with same VRF. Else, while configuring, the CLI parser will display an error.

```
Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1

Device(config)# dial-peer voice 3333 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/2
```

Configuration on Standby Router

The configurations of Active and Stand By should be identical.

Configuring VRF

```
Device> enable
Device# configure terminal
Device(config)# ip vrf VRF1
Device(config)# rd 1:1
Device(config)# ip vrf VRF2
Device(config)# rd 2:2

Device(config)# voice service voip
Device(config)# no ip address trusted authenticate
Device(config)# media bulk-stats
Device(config)# allow-connections sip to sip
Device(config)# redundancy-group 1
Device(config)# sip

Device(config)# redundancy
Device(config)# mode none
Device(config)# application redundancy
Device(config)# group 1
Device(config)# name raf-b2b
Device(config)# priority 1
Device(config)# timers delay 30 reload 60
Device(config)# control GigabitEthernet0/0/0 protocol 1
Device(config)# data GigabitEthernet0/0/0
```

Associating interfaces with VRF

```
Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2
```
If an IP address is already assigned to an interface, then associating a VRF with interface will disable the interface and remove the existing IP address. An error message (sample error message shown below) is displayed on the console. Assign the IP address to proceed further.

% Interface GigabitEthernet0/1 IPv4 disabled and address(es)removed due to enabling VRF VRF1

GigabitEthernet0/0/0 is used for configuring RG Infra and therefore do not configure any VRF with this interface.

```bash
Device(config)# interface GigabitEthernet0/0/0
Device(config-if)# ip address 14.2.43.81 255.255.0.0
Device(config-if)# negotiation auto
Device(config-if)# cdp enable
```

Inbound interface - GigabitEthernet0/1 is used for voice traffic configured with VRF1.

```bash
Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip vrf forwarding VRF1
Device(config-if)# ip address 10.0.0.4 255.0.0.0
Device(config-if)# negotiation auto
Device(config-if)# cdp enable
Device(config-if)# redundancy rii 1
Device(config-if)# redundancy group 1 ip 10.0.0.1 exclusive
```

Outbound interface - GigabitEthernet0/2 is used for voice traffic configured with VRF2.

```bash
Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2
Device(config-if)# ip address 11.0.0.4 255.0.0.0
Device(config-if)# negotiation auto
Device(config-if)# cdp enable
Device(config-if)# redundancy rii 2
Device(config-if)# redundancy group 1 ip 11.0.0.1 exclusive
```

Creating Dial-peer

Creating Inbound Dial-peer:

```bash
Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# destination pattern 1111
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.0.0.2
Device(config-dial-peer)# incoming called-number 1111
```

Creating Outbound Dial-peer:

```bash
Device(config)# dial-peer voice 3333 voip
Device(config)# destination-pattern 2222
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:11.0.0.2
```
Configuring Binding

Control and Media on a dial-peer have to bind with same VRF. Else, while configuring, the CLI parser will display an error.

Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/1

Device(config)# dial-peer voice 3333 voip
Device(config)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config)# voice-class sip bind media source-interface Gigabit Ethernet0/2

Verification of Calls Before and After Switchover

RTP Connections on Active router:

Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 2

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Port</th>
<th>Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media Pool</td>
<td>8000</td>
<td>19999</td>
<td>101</td>
<td>2</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No.</th>
<th>CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
<th>MPSS</th>
<th>VRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>6</td>
<td>8008</td>
<td>16388</td>
<td>10.0.0.1</td>
<td>11.0.0.2</td>
<td>NO</td>
<td>VRF1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>8100</td>
<td>16388</td>
<td>11.0.0.1</td>
<td>11.0.0.2</td>
<td>NO</td>
<td>VRF2</td>
</tr>
</tbody>
</table>

Found 2 active RTP connections

RTP Connections on Standby Router after switchover

Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 2

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Port</th>
<th>Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media Pool</td>
<td>8000</td>
<td>19999</td>
<td>101</td>
<td>2</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No.</th>
<th>CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
<th>MPSS</th>
<th>VRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>8</td>
<td>8012</td>
<td>16390</td>
<td>10.0.0.1</td>
<td>11.0.0.2</td>
<td>NO</td>
<td>VRF1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
<td>8014</td>
<td>16390</td>
<td>11.0.0.1</td>
<td>11.0.0.2</td>
<td>NO</td>
<td>VRF2</td>
</tr>
</tbody>
</table>

Found 2 active RTP connections

Active calls on Active Router
Device# show call active voice brief

dur 00:03:37 tx:6757/405420 rx:6757/405420
dscp:0 media:0 audio tos:0x0 video tos:0x0
IP 11.0.0.2:16390 SRTPF: off rtt:65531ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:
off Transcoded: No ICE: Off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00

11F9 : 7 245073850ms.1 (*12:16:18.114 UTC Mon May 25 2015) +26840 pid:0 Answer connected
dur 00:03:37 tx:6757/405420 rx:6757/405420
dscp:0 media:0 audio tos:0x0 video tos:0x0
IP 10.0.0.2:16390 SRTPF: off rtt:65531ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:

Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# show sip-ua connections udp brief

Total active connections : 2
No. of send failures : 0
No. of remote closures : 0
No. of conn. failures : 0
No. of inactive conn. ageouts : 2

------------- SIP Transport Layer Listen Sockets -------------

<table>
<thead>
<tr>
<th>Conn-Id</th>
<th>Local-Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[10.0.0.1]:5060:VRF1</td>
</tr>
<tr>
<td>3</td>
<td>[11.0.0.1]:5060:VRF2</td>
</tr>
</tbody>
</table>

Active calls on Standby router after switchover:

Device# show call active voice brief

dur 00:03:37 tx:6757/405420 rx:6757/405420
dscp:0 media:0 audio tos:0x0 video tos:0x0
IP 11.0.0.2:16390 SRTPF: off rtt:65531ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:
off Transcoded: No ICE: Off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00

11F9 : 7 245073850ms.1 (*12:16:18.114 UTC Mon May 25 2015) +26840 pid:0 Answer connected
dur 00:03:37 tx:6757/405420 rx:6757/405420
dscp:0 media:0 audio tos:0x0 video tos:0x0
IP 10.0.0.2:16390 SRTPF: off rtt:65531ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:
Example: Configuring HSRP High Availability with VRF

Below configuration example is applicable for Cisco Integrated Services Routers Generation 2 (ISR G2) Platforms. [Cisco 2900 Series Integrated Services Routers and Cisco 3900 Series Integrated Services Routers]

Note: Do not configure VRF on the interface that is used for HSRP. Traffic of VRF and HSRP should be on different interfaces.

Figure 37: Multi-VRF in High Availability Mode (HSRP)

Configuration on Active Router

Note: The configurations of Active Router and Stand By Router should be identical.

Configuring VRF

Device> enable
Device# configure terminal
Device(config)# ip vrf VRF1
Device(config)# rd 1:1
Device(config)# ip vrf VRF2
Associating interfaces with VRF

Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip vrf forwarding VRF1

Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2

If an IP address is already assigned to an interface, then associating a VRF with interface will disable the interface and remove the existing IP address. An error message (sample error message shown below) is displayed on the console. Assign the IP address to proceed further.

% Interface GigabitEthernet0/1 IPv4 disabled and address(es) removed due to enabling VRF VRF1

Note

The interface used for HSRP should not be configured with any VRF. In this example, GigabitEthernet0/0/0 is used for configuring HSRP and therefore no VRF is associated with this interface.

Device(config)# interface GigabitEthernet0/0/0
Device(config-if)# ip address 14.2.43.81 255.255.0.0
Device(config-if)# standby version 2
Device(config-if)# standby 93 ip 14.2.43.82
Device(config-if)# standby 93 priority 50
Device(config-if)# standby 93 preempt
Device(config-if)# standby 93 name cubeha
Device(config-if)# standby 93 track 1 decrement 5
Device(config-if)# standby 93 track 2 decrement 5
Device(config-if)# duplex auto
Device(config-if)# speed auto

Inbound interface - GigabitEthernet0/1 is used for voice traffic configured with VRF1.

Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip vrf forwarding VRF1
Device(config-if)# ip address 10.0.0.3 255.0.0.0
Device(config-if)# standby version 2
Device(config-if)# standby 63 ip 10.0.0.4
Device(config-if)# standby 63 priority 50
Device(config-if)# standby 63 preempt
Device(config-if)# standby 63 track 1 decrement 5
Device(config-if)# duplex auto
Device(config-if)# speed auto
Device(config-if)# media-type rj45

Outbound interface - GigabitEthernet0/2 is used for voice traffic configured with VRF2.

Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2
Device(config-if)# ip address 11.0.0.3 255.0.0.0
Device(config-if)# standby version 2
Device(config-if)# standby 36 ip 11.0.0.4
Device(config-if)# standby 36 priority 50
Device(config-if)# standby 36 preempt
Device(config-if)# standby 36 track 1 decrement 5
Device(config-if)# duplex auto
Device(config-if)# speed auto
Device(config-if)# media-type rj45

Device(config)# ipc zone default
Device(config-ipczone)# association 1
Device(config-ipczone-assoc)# no shutdown
Device(config-ipczone-assoc)# protocol sctp
Device(config-ipc-protocol-sctp)# local port 5000
Device(config-ipc-local-sctp)# local-ip 14.2.43.81
Device(config-ipc-local-sctp)# exit
Device(config-ipc-protocol-sctp)# remote port 5000
Device(config-ipc-remote-sctp)# remote-ip 14.2.43.82

Creating Dial-peer
Creating Inbound Dial-peer:

Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# destination pattern 1111
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.0.0.2
Device(config-dial-peer)# incoming called-number 1111

Creating Outbound Dial-peer:

Device(config)# dial-peer voice 3333 voip
Device(config)# destination-pattern 2222
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:11.0.0.2

Configuring Binding

Control and Media on a dial-peer have to bind with same VRF. Else, while configuring, the CLI parser will display an error.

Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1
Device(config)# dial-peer voice 3333 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/2

Configuration on Standby Router
The configurations of Active and Stand By should be identical.

**Configuring VRF**

Device> **enable**
Device# **configure terminal**
Device(config)# **ip vrf VRF1**
Device(config)# **rd 1:1**
Device(config)# **ip vrf VRF2**
Device(config)# **rd 2:2**

**Associating interfaces with VRF**

Device(config)# **interface GigabitEthernet0/1**
Device(config-if)# **ip vrf forwarding VRF1**

Device(config)# **interface GigabitEthernet0/2**
Device(config-if)# **ip vrf forwarding VRF2**

**Note**

If an IP address is already assigned to an interface, then associating a VRF with interface will disable the interface and remove the existing IP address. An error message (sample error message shown below) is displayed on the console. Assign the IP address to proceed further.

```
% Interface GigabitEthernet0/1 IPv4 disabled and address(es) removed due to enabling VRF VRF1
```

The interface used for HSRP should not be configured with any VRF. In this example, GigabitEthernet0/0/0 is used for configuring HSRP and therefore no VRF is associated with this interface.

Device(config)# **interface GigabitEthernet0/0/0**
Device(config-if)# **ip address 14.2.43.82 255.255.0.0**
Device(config-if)# **standby version 2**
Device(config-if)# **standby 93 ip 14.2.43.81**
Device(config-if)# **standby 93 priority 50**
Device(config-if)# **standby 93 preempt**
Device(config-if)# **standby 93 name cubeha**
Device(config-if)# **standby 93 track 1 decrement 5**
Device(config-if)# **standby 93 track 2 decrement 5**
Device(config-if)# **duplex auto**
Device(config-if)# **speed auto**

Inbound interface - GigabitEthernet0/1 is used for voice traffic configured with VRF1.

Device(config)# **interface GigabitEthernet0/1**
Device(config-if)# **ip vrf forwarding VRF1**
Device(config-if)# **ip address 10.0.0.4 255.0.0.0**
Device(config-if)# **standby version 2**
Device(config-if)# **standby 63 ip 10.0.0.3**
Device(config-if)# **standby 63 priority 50**
Device(config-if)# **standby 63 preempt**
Example: Configuring HSRP High Availability with VRF

Device(config-if)# standby 63 track 1 decrement 5
Device(config-if)# duplex auto
Device(config-if)# speed auto
Device(config-if)# media-type rj45

Outbound interface - GigabitEthernet0/2 is used for voice traffic configured with VRF2.

Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2
Device(config-if)# ip address 11.0.0.4 255.0.0.0
Device(config-if)# standby version 2
Device(config-if)# standby 36 ip 11.0.0.3
Device(config-if)# standby 36 priority 50
Device(config-if)# standby 36 preempt
Device(config-if)# standby 36 track 1 decrement 5
Device(config-if)# duplex auto
Device(config-if)# speed auto
Device(config-if)# media-type rj45

Device(config)# ip zone default
Device(config-ipzone)# association 1
Device(config-ipzone-assoc)# no shutdown
Device(config-ipzone-assoc)# protocol sctp
Device(config-ip-protocol-sctp)# local port 5000
Device(config-ip-local-sctp)# local-ip 14.2.43.82
Device(config-ip-remote-sctp)# exit
Device(config-ip-protocol-sctp)# remote port 5000
Device(config-ip-remote-sctp)# remote-ip 14.2.43.81

Creating Dial-peer

Creating Inbound Dial-peer:

Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# destination pattern 1111
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.0.0.2
Device(config-dial-peer)# incoming called-number 1111

Creating Outbound Dial-peer:

Device(config)# dial-peer voice 3333 voip
Device(config)# destination-pattern 2222
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:11.0.0.2

Configuring Binding

Note

Control and Media on a dial-peer have to bind with same VRF. Else, while configuring, the CLI parser will display an error.

Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/1

Device(config)# dial-peer voice 3333 voip
Device(config)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/2

Verification of redundancy States

On Active Router

Device(config)# show redundancy status

my state = 13 -ACTIVE
peer state = 8 -STANDBY HOT
Mode = Duplex
Unit ID = 0

Maintenance Mode = Disabled
Manual Swact = enabled
Communications = Up

client count = 17
client_notification_TMR = 120000 milliseconds
RF debug mask = 0x0

On Standby Router

Device(config)# show redundancy status

my state = 8 -STANDBY HOT
peer state = 13 ACTIVE
Mode = Duplex
Unit ID = 0

Maintenance Mode = Disabled
Manual Swact = enabled
Communications = Up

client count = 17
client_notification_TMR = 120000 milliseconds
RF debug mask = 0x0

Verification of Calls Before and After Switchover

RTP Connections on Active router:

Device# show voip rtp connections

VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 2

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Min</th>
<th>Max</th>
<th>Ports</th>
<th>Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media Pool</td>
<td>8000</td>
<td>48198</td>
<td>19999</td>
<td>101</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No. CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
<th>MPSS</th>
<th>VRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>6</td>
<td>8008</td>
<td>16388</td>
<td>10.0.0.1</td>
<td>10.0.0.2</td>
<td>NO</td>
</tr>
</tbody>
</table>
Found 2 active RTP connections

RTP Connections on Standby Router after switchover

Device# show voip rtp connections

VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 2

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Min Port</th>
<th>Max Port</th>
<th>Available Ports</th>
<th>Reserved Ports</th>
<th>In-use Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media Pool</td>
<td>8000</td>
<td>48198</td>
<td>19999</td>
<td>101</td>
<td>2</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No. CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>8</td>
<td>8012</td>
<td>16390</td>
<td>10.0.0.1</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>8</td>
<td></td>
<td>10.0.0.2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
<td>8014</td>
<td>16390</td>
<td>11.0.0.1</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td></td>
<td></td>
<td>11.0.0.2</td>
<td></td>
</tr>
</tbody>
</table>

Found 2 active RTP connections

Active calls on Active Router

Device# show call active voice brief

11F3 : 5 243854170ms.1 (*11:48:43.972 UTC Mon May 25 2015) +6770 pid:0 Answer active
dur 00:00:14 tx:843/50551 rx:1028/61680 dscp:0 media:0 audio tos:0x08 video tos:0x0
IP 10.0.0.2:16388 SRTP: off rtt:ms pl:0/0ms lost:0/0 delay:0/0/0ms g729r8 TextRelay: off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00

active
dur 00:00:14 tx:1028/61680 rx:843/50551 dscp:0 media:0 audio tos:0x08 video tos:0x0
IP 11.0.0.2:16388 SRTP: off rtt:65522ms pl:0/0ms lost:0/0 delay:0/0/0ms g729r8 TextRelay: off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00

Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# show sip-ua connections udp brief

Total active connections: 2
No. of send failures: 0
No. of remote closures : 0
No. of conn. failures : 0
No. of inactive conn. ageouts : 2

--------------- SIP Transport Layer Listen Sockets ---------------

<table>
<thead>
<tr>
<th>Conn-Id</th>
<th>Local-Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[10.0.0.1]:5060:VRF1</td>
</tr>
<tr>
<td>3</td>
<td>[11.0.0.1]:5060:VRF2</td>
</tr>
</tbody>
</table>

Active calls on Standby router after switchover:

Device# show call active voice brief

dur 00:03:37 tx:6757/405420 rx:6757/405420 dscp:0 media:0 audio tos:0x0 video tos:0x0
IP 11.0.0.2:16390 SRTF: off rtt:65531ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay: off
Transcoded: No ICE: Off
media inactive detected:n media ctrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00

11F9 : 7 245073850ms.1 (*12:16:18.114 UTC Mon May 25 2015) +26840 pid:0 Answer connected
dur 00:03:37 tx:6757/405420 rx:6757/405420 dscp:0 media:0 audio tos:0x0 video tos:0x0
IP 10.0.0.2:16390 SRTF: off rtt:65523ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay: off
Transcoded: No ICE: Off
media inactive detected:n media ctrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00

Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2
Example: Configuring HSRP High Availability with VRF
CHAPTER 27

Configuring Multi-Tenants on SIP Trunks

This feature allows specific global configurations for multiple tenants on SIP trunks that allow differentiated services for tenants. Configuring Multi-Tenants on SIP Trunks allows each tenant to have their own individual configurations. The configurations include timers, credentials, bind requests, and other parameters which are available under sip-ua and voice service voip/sip configurations. Multi-tenant functionality helps to create multiple configurations with ease and provides support for scalable and flexible mix of typical enterprise services.

- Feature Information for Configuring Multi-Tenants on SIP Trunks, on page 375
- Information About Configuring Multi-tenants on SIP Trunks, on page 375
- How to Configure Multi-Tenants on SIP Trunks, on page 379
- Example: SIP Trunk Registration in Multi-Tenant Configuration, on page 381

Feature Information for Configuring Multi-Tenants on SIP Trunks

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Support for Configuring Multi Tenants on SIP Trunks | Cisco IOS 15.6(2)T  
Cisco IOS XE Denali 16.3.1 | This feature allows the provision to configure specific global configurations for multiple tenants on SIP trunks.

The following commands were introduced: voice class tenant <tag> and voice-class sip tenant <tag>.

Information About Configuring Multi-tenants on SIP Trunks

With the introduction of multi-tenancy support on CUBE, the sip-specific attributes can be configured at per tenant basis in addition to the existing global or dial-peer levels.
The **voice class tenant** <tag> command allows sip-specific attributes to be configured at per tenant basis. The command **voice class tenant** <tag> can be then applied to individual dial-peers, thereby associating them to a particular tenant. See the following table "Table 44: Multi-Tenant Configuration List" for information on the complete list of configurations present under the **voice class tenant** <tag>.

If tenants are configured under dial-peer, then configurations are applied in the following order of preference.

- Dial-peer configuration
- Tenant configuration
- Global configuration

That is, if the value of the attribute under dial-peer configuration is system, then the value is taken from the tenant configuration. And, if the value under the tenant configuration is also system, then the global configuration is used.

If there are no tenants configured under dial-peer, then the configurations are applied using the default behavior in the following order:

- Dial-peer configuration
- Global configuration

The following table lists the various configurations present under **voice class tenant** <tag>. For more information on specific configurations, see the Voice and Video command reference guide lists.

---

**Note**

Attributes that are not available under **voice class tenant** <tag> use the default behavior—With preference of dial-peer followed by the global configuration.

### Table 44: Multi-Tenant Configuration List

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaa</td>
<td>SIP-UA AAA related configuration</td>
</tr>
<tr>
<td>anat</td>
<td>Allow alternative network address types IPv4 and IPv6</td>
</tr>
<tr>
<td>asserted-id</td>
<td>Configure SIP UA privacy identity settings</td>
</tr>
<tr>
<td>associate</td>
<td>Associate a RCB for outgoing calls</td>
</tr>
<tr>
<td>asymmetric</td>
<td>Configure global SIP asymmetric payload support</td>
</tr>
<tr>
<td>authenticate</td>
<td>Call authentication policy</td>
</tr>
<tr>
<td>authentication</td>
<td>Digest Authentication Configuration</td>
</tr>
<tr>
<td>bandwidth</td>
<td>Allow SIP SDP bandwidth-related options</td>
</tr>
<tr>
<td>bind</td>
<td>SIP bind command</td>
</tr>
<tr>
<td>block</td>
<td>Block 18X response to INVITE</td>
</tr>
<tr>
<td><strong>Command</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>call-route</td>
<td>Configure call routing options</td>
</tr>
<tr>
<td>conn-reuse</td>
<td>Reuse the sip registration tcp connection for the end-point behind a Firewall</td>
</tr>
<tr>
<td>connection-reuse</td>
<td>Use listener port for sending requests over UDP</td>
</tr>
<tr>
<td>contact-passing</td>
<td>302 contact to be passed through for CFWD</td>
</tr>
<tr>
<td>content</td>
<td>Content carried as part of SIP message</td>
</tr>
<tr>
<td>copy-list</td>
<td>Configure list of entities to be sent to peer leg</td>
</tr>
<tr>
<td>credentials</td>
<td>User credentials for registration</td>
</tr>
<tr>
<td>disable-early-media</td>
<td>Disable early-media cut through</td>
</tr>
<tr>
<td>dns -a-override</td>
<td>Skip DNS A/AAAA query when SRV query timesout</td>
</tr>
<tr>
<td>dscp -profile</td>
<td>DSCP Profile global config</td>
</tr>
<tr>
<td>early-media</td>
<td>Configure method to handle early-media Update Request</td>
</tr>
<tr>
<td>early-offer</td>
<td>Configure sending Early-Offer</td>
</tr>
<tr>
<td>encaps</td>
<td>Configure SDP encapsulation</td>
</tr>
<tr>
<td>error-code-override</td>
<td>Configure sip error code</td>
</tr>
<tr>
<td>error- passthru</td>
<td>SIP error response pass-thru functionality</td>
</tr>
<tr>
<td>exit</td>
<td>Exits from the voice class configuration mode</td>
</tr>
<tr>
<td>g729</td>
<td>G729 codec interoperability settings</td>
</tr>
<tr>
<td>handle-replaces</td>
<td>Handle INVITE with REPLACES header at SIP spi</td>
</tr>
<tr>
<td>header-passing</td>
<td>SIP Headers need to be passed to applications</td>
</tr>
<tr>
<td>help</td>
<td>Description of the interactive help system</td>
</tr>
<tr>
<td>history-info</td>
<td>History Info header support</td>
</tr>
<tr>
<td>host-registrar</td>
<td>Use sip-ua registrar value in Diversion and Contact header for 3xx messages</td>
</tr>
<tr>
<td>interop-handling</td>
<td>Enable interop-handling</td>
</tr>
<tr>
<td>localhost</td>
<td>Specify the DNS name for the localhost</td>
</tr>
<tr>
<td>map</td>
<td>Mapping options</td>
</tr>
<tr>
<td>max-forwards</td>
<td>Change number of max-forwards for SIP Methods</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>midcall -signaling</td>
<td>Configure method to handle mid-call signaling</td>
</tr>
<tr>
<td>nat</td>
<td>SIP nat global config</td>
</tr>
<tr>
<td>no</td>
<td>Negate a command or set its defaults</td>
</tr>
<tr>
<td>notify</td>
<td>SIP Signaling Notify Configuration</td>
</tr>
<tr>
<td>offer</td>
<td>Configure settings for Offers made from the Gateway</td>
</tr>
<tr>
<td>options-ping</td>
<td>Send OPTION pings to remote end</td>
</tr>
<tr>
<td>outbound-proxy</td>
<td>Configure an Outbound Proxy Server</td>
</tr>
<tr>
<td>pass-thru</td>
<td>SIP pass-through global config</td>
</tr>
<tr>
<td>permit</td>
<td>Permit hostname for this gateway</td>
</tr>
<tr>
<td>preloaded-route</td>
<td>Use pre-loaded route header for outgoing calls, if available</td>
</tr>
<tr>
<td>privacy</td>
<td>Configure SIP UA privacy settings</td>
</tr>
<tr>
<td>privacy-policy</td>
<td>Set privacy behavior for outgoing SIP messages</td>
</tr>
<tr>
<td>random-contact</td>
<td>Use Random Contact for outgoing calls, if available</td>
</tr>
<tr>
<td>random-request-uri</td>
<td>Configure options for Request-URI having random value</td>
</tr>
<tr>
<td>reason-header</td>
<td>Configure settings for supporting SIP Reason Header</td>
</tr>
<tr>
<td>redirection</td>
<td>Enable call redirection (3xx) handling</td>
</tr>
<tr>
<td>refer-ood</td>
<td>Configure maximum number of out-of-dialog refer made to the Gateway</td>
</tr>
<tr>
<td>refer-to-passing</td>
<td>Refer-To needs to be passed through for transfer</td>
</tr>
<tr>
<td>registrar</td>
<td>Configure SIP registrar VoIP Interface</td>
</tr>
<tr>
<td>registration</td>
<td>Enable registration options</td>
</tr>
<tr>
<td>rel1xx</td>
<td>Type of reliable provisional response support</td>
</tr>
<tr>
<td>remote-party-id</td>
<td>Enable Remote-Party-ID support in SIP User Agent</td>
</tr>
<tr>
<td>requiri-passing</td>
<td>Request URI needs to be passed through</td>
</tr>
<tr>
<td>reset</td>
<td>SIP Reset Options</td>
</tr>
<tr>
<td>retry</td>
<td>Change default retries for each SIP Method</td>
</tr>
<tr>
<td>send</td>
<td>Configure outgoing message options</td>
</tr>
<tr>
<td>session</td>
<td>SIP Voice Protocol session config</td>
</tr>
</tbody>
</table>
How to Configure Multi-Tenants on SIP Trunks

Configuring Multi-Tenants on SIP Trunks

SUMMARY STEPS

1. enable
2. configure terminal
3. Use the following commands to configure multi-tenants:
   • voice class tenant <tag> in the global configuration mode

   Once you configure the voice class tenant <tag> command in the global mode, the configuration will move to the voice class tenant <tag> submode. You can configure all the sip-specific attributes in this submode.

   • voice-class sip tenant <tag> in the dial-peer configuration mode

4. authenticate
5. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sip-profiles</th>
<th>SIP Profiles global config</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip-server</td>
<td>Configure a SIP Server Interface</td>
</tr>
<tr>
<td>srtp</td>
<td>Allow SIP related SRTP options</td>
</tr>
<tr>
<td>srtp-auth</td>
<td>Allow to set preferred suites</td>
</tr>
<tr>
<td>tel-config</td>
<td>Tel format cfg for headers other than req -line in</td>
</tr>
<tr>
<td>timers</td>
<td>SIP Signaling Timers Configuration</td>
</tr>
<tr>
<td>update-callerid</td>
<td>Enable sending updates for callerid</td>
</tr>
<tr>
<td>url</td>
<td>Url configuration for request-line url in outgoing INVITE</td>
</tr>
<tr>
<td>video</td>
<td>Video related config for sip</td>
</tr>
<tr>
<td>warn-header</td>
<td>SIP Warning-Header global config</td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configure terminal</strong></td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

### Step 2

**Example:**

Device# configure terminal

### Step 3

Use the following commands to configure multi-tenants:

- **voice-class tenant <tag>** in the global configuration mode

  Once you configure the **voice-class tenant <tag>** command in the global mode, the configuration will move to the **voice-class tenant <tag>** submode. You can configure all the sip-specific attributes in this submode.

- **voice-class sip tenant <tag>** in the dial-peer configuration mode

**Example:**

In global configuration mode

```
! Configuring tenant 1
Device(config)# voice-class tenant 1
Device (config-class)# ?
aaa – sip-ua AAA related configuration
anat – Allow alternative network address types IPV4 and IPV6
asserted-id – Configure SIP-UA privacy identity settings
    ...
    ...
Video – video related function
Warn-header – SIP related config for SIP. SIP warning-header global config.
Device (config-voi-tenant)# end
--------
```

```
! Configuring tenant 2
Device(config)# voice-class tenant 2
Device (config-class)# ?
aaa – sip-ua AAA related configuration
anat – Allow alternative network address types IPV4 and IPV6
asserted-id – Configure SIP-UA privacy identity settings
    ...
    ...
outbound-proxy – Configure an Outbound Proxy Server
pass-thru – SIP pass-through global config
    ...
    ...
srtp – Allow SIP related SRTP options
Warn-header – SIP related config for SIP. SIP warning-header global config.
Device (config-voi-tenant)# end
```

**Example:**
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>In dial-peer configuration mode</td>
<td></td>
</tr>
</tbody>
</table>
| !Configuring tenant 1 under dial-peer 10  
Device (config)# dial-peer voice 10 voip  
Device (config-dial-peer)# voice-class sip tenant 1  
Device (config-dial-peer)# authenticate  
Device (config-dial-peer)# end | |
| !Configuring tenant 2 under dial-peer 20  
Device (config)# dial-peer voice 20 voip  
Device (config-dial-peer)# voice-class sip tenant 2  
Device (config-dial-peer)# authenticate  
Device (config-dial-peer)# end | |
| !An example for the use of the "no" form of command voice-class sip tenant  
Router(config)# dial-peer voice 3000 voip  
Router(config-dial-peer)# voice-class sip tenant 1  
Router(config-dial-peer)# no voice-class sip tenant 1 | When the no form is configured, the dial-peer is no longer associated with the tenant tag configuration. The attributes are now applied using the default order of dial-peer followed by the global configuration. |
| Step 4 authenticate  
Example:  
Device(config-dial-peer)# authenticate | Commits the configuration. |
| Step 5 end  
Example:  
Device(config-dial-peer)# end | Returns to privileged EXEC mode. |

**Example: SIP Trunk Registration in Multi-Tenant Configuration**

For SIP trunk registration, the `voice class tenant <tag>` command is not associated with any dial-peer configuration. All outgoing registrations are triggered to the Registrars when credentials are configured under `voice class tenant <tag>`.  

Router# show run | sec tenant

```
Voice class tenant 1
  registrar 1 ipv4:10.64.86.35:9051 expires 3600
  credentials username aaaa password 7 06070E204D realm aaaa.com
  outbound-proxy ipv4:10.64.86.35:9057
  bind control source-interface GigabitEthernet0/0
```
Voice class tenant 2
registrar 1 ipv4:9.65.75.45:9052 expires 3600
credentials username bbbb password 7 110B1B0715 realm bbbb.com
outbound-proxy ipv4:10.64.86.40:9040
bind control source-interface GigabitEthernet0/1

For multi-tenancy support on Cisco Unified Border Element, you can configure voice class tenants with different credentials, but having the same registrar. In that scenario, it is recommended that you configure the CLI commands sip-server and registrar under voice class tenant configuration. The following is a sample configuration:

voice class tenant 1
credentials number 1111 username test password 7 071B245B5D1D realm ipvoice.jp
authentication username test password 7 06120A3258
registrar ipv4:1.1.1.1 expires 120
sip-server ipv4:1.1.1.1

voice class tenant 2
credentials number 2222 username test password 7 09584B1E0A11 realm ipvoice.jp
authentication username test2 password 7 071B245F5A
registrar ipv4:1.1.1.1 expires 120
sip-server ipv4:1.1.1.1
PART IV

Codecs

• Codec Support and Restrictions, on page 385
• Codec Preference Lists, on page 389
Codec Support and Restrictions

This chapter provides advanced information about the support of and restrictions for certain codecs on CUBE. For basic information on how to configure codecs, refer to the Introduction to Codecs section.

- Feature Information for Codec Support on CUBE, on page 385
- ISAC Codec Support on CUBE, on page 386
- AAC-LD MP4A-LATM Codec Support on Cisco UBE, on page 386

Feature Information for Codec Support on CUBE

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 45: Feature Information for Codec Support on CUBE

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC-LD MP4A-LATM Codec Support on Cisco UBE</td>
<td>15.4(1)T Cisco IOS XE Release 3.12S</td>
<td>The AAC-LD MP4A-LATM codec is a wideband audio codec used by video endpoints. MP4A-LATM is an MPEG4 audio coding standard, where LATM is Low-Overhead MPEG-4 Audio Transport Multiplex. The Cisco Unified Border Element (Cisco UBE) supports MP4A-LATM to enable call flows involving endpoints that use this codec, especially for media recording. The following commands were introduced or modified: <code>codec mp4a-latm, codec preference tag mp4a-latm</code></td>
</tr>
<tr>
<td>ISAC Codec Support on CUBE</td>
<td>15.1(1)T</td>
<td>The ISAC Codec Support on CUBE The following commands were introduced by this feature: <code>codec isac, codec preference tag isac</code>.</td>
</tr>
</tbody>
</table>
ISAC Codec Support on CUBE

The iSAC codec is an adaptive VoIP codec especially designed to deliver wideband sound quality in both low- and high-bit rate applications. The iSAC codec automatically adjusts the bit-rate for the best quality or a fixed bit rate can be used if the network characteristics are known. This codec is designed for wideband VoIP communications. The iSAC codec offers better quality with reduced bandwidth for sideband applications.

Restrictions for ISAC Codec Support on CUBE

- Low complexity is not supported for the iSAC codec.

AAC-LD MP4A-LATM Codec Support on Cisco UBE

As part of this feature, Cisco UBE supports the following:

- Accept and send MP4A-LATM codec and corresponding FMTP profiles
- Configure MP4A-LATM under dial-peer or under voice-class codec as preferred codec
- Pass across real-time transport protocol (RTP) media for MP4A-LATM codec without any interworking
- Offer pre-configured FMTP profile for MP4A-LATM codec for DO-EO (Delayed-Offerto Early-Offer) calls
- Offer more than one FMTP profile (each with different payload type number) as mentioned by the offering endpoint, so that the answering endpoint can choose the best option.
- Offer only one instance of MP4A-LATM if media forking is applicable. The offered instance is the first one received in the offer.
- Calculate bandwidth for MP4A-LATM on the basis of either “b=TIAS” attribute or “bitrate” parameter in the FMTP attribute. If none of them are present in the session description protocol (SDP), the default maximum bandwidth, that is, 128 Kbps will be used for calculation.
- The following Cisco UBE features are supported with the MP4A-LATM codec:
  - Basic call (audio and video) flow-around and flow-through (FA and FT).
  - Voice Class Codec support in Cisco UBE with codec filtering
  - SRTP and SRCTP passthrough for SIP-to-SIP calls
  - Supplementary services
  - RSVP
  - Dynamic payload type interworking for DTMF and codec packets for SIP-to-SIP calls
  - Media Anti-Trombone with SIP signaling control on CUBE
  - Support for SIP UPDATE message per RFC 3311
  - RTP Media Loopback
  - Media forking for IP based calls using Zephyr recording server
  - Cisco UBE Mid-call Re-INVITE consumption
  - Signaling forking (Fastweb multile SIP Early Dialog Support, FA and FT)
  - Maximum bandwidth-based CAC
  - Media Policing
  - Box-to-Box High Availability (B2B HA)
  - Inbox High Availability (Inbox HA)
Restrictions for AAC-LD MP4A-LATM Codec Support on Cisco UBE

Cisco UBE does not support the following:

• Codec transcoding between MP4A-LATM and other codecs
• Dual-tone Multifrequency (DTMF) interworking with MP4A-LATM codec
• Non-SIP, that is, SIP to other service provider interface (SPI) interworking with MP4A-LATM codec
Codec Preference Lists

This chapter describes how to negotiate an audio codec from a list of codec associated with a preference. This chapter also describes how to disable codec filtering by configuring CUBE to send an outgoing offer with all configured audio codecs in the list assuming that the dspfarm supports all these codecs.

- Feature Information for Negotiation of an Audio Codec from a List of Codecs, on page 389
- Codecs Configured Using Preference Lists, on page 390
- Prerequisites for Codec Preference Lists, on page 390
- Restrictions for Codecs Preference Lists, on page 391
- How to Configure Codec Preference Lists, on page 391
- Troubleshooting Negotiation of an Audio Codec from a List of Codecs, on page 394
- Verifying Negotiation of an Audio Codec from a List of Codecs, on page 394

Feature Information for Negotiation of an Audio Codec from a List of Codecs

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

**Table 46: Feature Information for Negotiation of an Audio Codec from a List of Codecs on Each Leg of a SIP-to-SIP Call on the Cisco Unified Border Element**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negotiation of an Audio Codec from a List of Codecs on Each Leg of a SIP-to-SIP Call on the Cisco Unified Border Element</td>
<td>15.1(2)T</td>
<td>The Negotiation of an Audio Codec from a List of Codecs on Each Leg of a SIP-to-SIP Call on the Cisco Unified Border Element feature supports negotiation of an audio codec using the Voice Class Codec and Codec Transparent infrastructure on the Cisco UBE. The following command was introduced or modified: voice-class codec (dial peer).</td>
</tr>
</tbody>
</table>
## Codecs Configured Using Preference Lists

SIP-to-SIP calls configured using codecs using preference lists have the following features:

- Incoming and outgoing dial-peers can be configured with different preference lists.
- Both normal transcoding and high-density transcoding are supported with preference lists.
- Midcall codec changes for supplementary services are supported with preference lists. Transcoder resources are dynamically inserted or deleted when there is a codec or RTP-NTE to in-band DTMF interworking required.
- Reinvite-based supplementary services that are invoked from the Cisco Unified Communications Manager (CUCM), like call hold, call resume, Music On Hold (MOH), call transfer, and call forward are supported with preference lists.
- T.38 fax and fax passthrough switchover with preference lists are supported.
- Reinvite-based call hold and call resume for the Secure Real-Time Transfer protocol (SRTP) and Real-Time Transport Protocol (RTP) interworking on CUBE is supported with preference lists.
- High availability is supported for calls that use codecs with preference lists. But calls requiring the transcoder to be invoked are not checkpointed. During midcall renegotiation, if the call releases the transcoder, then the call is checkpointed.

### Prerequisites for Codec Preference Lists

- Transcoding configuration on the CUBE.
- The digital signal processor (DSP) requirements to support the transcoding feature on the CUBE.
Restrictions for Codecs Preference Lists

For All Calls (SIP-to-SIP, H323-to-H323, SIP-to-H323 calls)

• Video codecs are not supported with preference lists.
• Multiple audio streams are not supported.
• High-density transcoding is not supported when delayed offer to early offer is configured. Only low density transcoding is supported.
• Codec re-packetization feature is not supported when preference lists are configured.

For H323-to-H323 and SIP-to-H323 Calls

The below restrictions do not exist for SIP-to-SIP calls from 15.1(2)T and Cisco IOS XE Release 3.8S onwards.

• You can configure dissimilar preference lists on the incoming and outgoing dial peers.
• Incoming and outgoing dial-peers cannot be configured with the different preference lists.
• Transcoding is not supported when preference lists are used.
• Mid-call codec changes and supplementary services (call-hold / resume, call forward) do not work when a preference list is configured.
• Mid-call insertion or deletion of transcoder is not supported with preference lists.
• Rotary dial peers are not supported when preference lists are used.
• Both incoming and outgoing dial-peers need to be configured with the same codec voice classes.
• The preference of codecs configured in a codec voice classes is not be applied to the outgoing call-leg. Basically codec filtering is applied first and only the filtered codecs will be sent out in the outgoing offer from CUBE.
• T.38 fax, fax-passthru and modem-passthru is not be supported with preference lists.
• SRTP<->RTP is not supported with preference lists.
• When a codec voice class is configured, call establishment is un-predictable when a transcoder is involved in the call. The call succeeds only if the end points choose the first codec in the list of offered codecs.

Note
 Codec preference in the voice class codec on the outgoing call leg is not followed when the same codecs are available in the respective incoming invite with SDP with different codec preference. Cube prioritizes and follows the incoming invite with SDP codec preference when compared to the voice class codec preference on the outgoing dial-peer leg.

How to Configure Codec Preference Lists

Configuring Audio Codecs Using a Codec Voice Class and Preference Lists

Preferences can be used to determine which codecs will be selected over others.

A codec voice class is a construct within which a codec preference order can be defined. A codec voice class can then be applied to a dial peer, which then follows the preference order defined in the codec voice class.
SUMMARY STEPS

1. enable
2. configure terminal
3. voice class codec tag
4. Do the following for each audio codec you want to configure in the voice class:
   • codec preference value codec-type [bytes payload-size fixed-bytes]
   • codec preference value isac [mode {adaptive | independent} [bit-rate value framesize {30 | 60} [fixed]]]
   • codec preference value ilbc [mode frame-size [bytes payload-size]]
   • codec preference value mp4-latm [profile tag]
5. exit
6. dial-peer voice number voip
7. voice-class codec tag offer-all
8. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  Example:  
  Device> enable |
| **Step 2** configure terminal | Enters global configuration mode.  
  Example:  
  Device> configure terminal |
| **Step 3** voice class codec tag | Enters voice-class configuration mode for the specified codec voice class.  
  Example:  
  Device(config)# voice class codec 10 |
| **Step 4** Do the following for each audio codec you want to configure in the voice class:  
  • codec preference value codec-type [bytes payload-size fixed-bytes]  
  • codec preference value isac [mode {adaptive | independent} [bit-rate value framesize {30 | 60} [fixed]]]  
  • codec preference value ilbc [mode frame-size [bytes payload-size]]  
  • codec preference value mp4-latm [profile tag] | Configure a codec within the voice class and specifies a preference for the codec. This becomes part of a preference list |
| **Step 5** exit | Exits the current mode.  
  Example:  
  Device(config-class)# exit |
  • Enter your password if prompted. |
Disabling Codec Filtering

Cisco UBE is configured to filter common codecs for the subsets, by default. The filtered codecs are sent in the outgoing offer. You can configure the Cisco UBE to offer all the codecs configured on an outbound leg instead of offering only the filtered codecs.

Note

This configuration is applicable only for early offer calls from the Cisco UBE. For delayed offer calls, by default all codecs are offered irrespective of this configuration.

SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice tag voip
4. voice-class codec tag offer-all
5. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice tag voip</td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# dial-peer voice 10 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class codec tag offer-all</td>
<td>Adds all the configured voice class codec to the outgoing offer from the Cisco UBE.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# voice-class codec 10 offer-all</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits the dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

## Troubleshooting Negotiation of an Audio Codec from a List of Codecs

Use the following commands to debug any errors that you may encounter when you configure the Negotiation of an Audio Codec from a List of Codecs on Each Leg of a SIP-to-SIP Call on the Cisco Unified Border Element feature:

- debug ccsip all
- debug voip ccapi input
- debug sccp messages
- debug voip rtp session

For DSP-related debugs, use the following commands:

- debug voip dsmp all
- debug voip dsmp rtp both payload all
- debug voip ipipgw

## Verifying Negotiation of an Audio Codec from a List of Codecs

Perform this task to display information to verify Negotiation of an Audio Codec from a List of Codecs on Each Leg of a SIP-to-SIP Call on the Cisco Unified Border Element configuration. These show commands need not be entered in any specific order.
SUMMARY STEPS

1. enable
2. show call active voice brief
3. show voip rtp connections
4. show sccp connections
5. show dspfarm dsp active

DETAILED STEPS

Step 1  enable
Enables privileged EXEC mode.

Step 2  show call active voice brief
Displays a truncated version of call information for voice calls in progress.

Example:

Device# show call active voice brief
<ID>: <CallID> <start>ms.<index> +<connect> pid:<peer_id> <dir> <addr> <state> 
dur hh:mm:ss tx:<packets>/bytes rx:<packets>/bytes
IP <ip>:<udp> rtt:<time>ms lost:<lost>/<early>/<late>
delay:<last>/<min>/<max> ms <codec>
media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>
MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
last <buf event time>s dur:<Min>/<Max>s
FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
Tele <int> (callid) [channel_id] tx:<tot>/<v>/<fax>ms <codec> noise:<l> acom:<l> i/o:<l>/<l> dBm
MODEMRRELAY info:<rcvd>/<sent/> xid:<rcvd>/<sent/> total:<rcvd>/<sent/> <drops>
speeds(bps): local <rx>/<tx> remote <rx>/<tx>
Proxy <ip>:<audio udp>,<video udp>,<tcp0>,<tcp2>,<tcp3> endpt: <type>/<manf>
bw: <req>/<act> codec: <audio> (payload size)
tx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 2
Multicast call-legs: 0
Total call-legs: 4
1243 : 11 971490ms.1 +1 pid:1 Answer 1230000 connecting
dur 00:00:00 tx:415/66400 rx:17/2561
IP 192.0.2.1:19304 SRTP: off rtt:0ms lost:0/0/0 delay:0/0/0ms g711ulaw TextRelay: off
media inactive detected:n media cntrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
1243 : 12 971500ms.1 +1 pid:2 Originate connecting
dur 00:00:00 tx:5/10 rx:4/8
IP 9.44.26.4:16512 SRTP: off rtt:0ms lost:0/0/0 delay:0/0/0ms g729br8 TextRelay: off
media inactive detected:n media cntrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
0 : 13 971560ms.1 +0 pid:0 Originate connecting
dur 00:00:08 tx:415/66400 rx:17/2561
IP 192.0.2.2:2000 SRTP: off rtt:0ms lost:0/0/0 delay:0/0/0ms g711ulaw TextRelay: off
Cisco Unified Border Element Configuration Guide
Verifying Negotiation of an Audio Codec from a List of Codecs

media inactive detected: n media contrl rcvd: n/a timestamp: n/a
long duration call detected: n long duration call duration: n/a timestamp: n/a
0 : 15 971570ms.1 +0 pid:0 originate connecting
dur 0:0:0:0 rx:5/10 tx:5/6
IP 192.0.2.3:2000 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729br8 TextRelay: off
media inactive detected: n media contrl rcvd: n/a timestamp: n/a
long duration call detected: n long duration call duration: n/a timestamp: n/a
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 2
Multicast call-legs: 0
Total call-legs: 4

Step 3  show voip rtp connections
Displays Real-Time Transport Protocol (RTP) connections.

Example:

Device# show voip rtp connections
VoIP RTP active connections :
No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP
1 11 12 16662 19304 192.0.2.1 192.0.2.2
2 12 11 17404 16512 192.0.2.2 192.0.2.1
3 13 14 18422 2000 192.0.2.4 9.44.26.3
4 15 14 16576 2000 192.0.2.6 192.0.2.5
Found 4 active RTP connections

Step 4  show sccp connections
Displays information about the connections controlled by the Skinny Client Control Protocol (SCCP) transcoding and conferencing applications.

Example:

Device# show sccp connections
sess_id conn_id stype mode codec sport rport ripaddr
5 5 xcode sendrecv g729b 16576 2000 192.0.2.3
5 6 xcode sendrecv g711u 18422 2000 192.0.2.4
Total number of active session(s) 1, and connection(s) 2

Step 5  show dspfarm dsp active
Displays active DSP information about the DSP farm service.

Example:

Device# show dspfarm dsp active
SLOT DSP VERSION STATUS CHNL USE TYPE RSC_ID BRIDGE_ID PKTS_TXED PKTS_RXED
0 1 27.0.201 UP 1 USED xcode 1 0x9 5 8
0 1 27.0.201 UP 1 USED xcode 1 0x8 2558 17
Total number of DSFFARM DSP channel(s) 1
PART V

DSP Services

• Transcoding, on page 399
• Transrating, on page 417
• Call Progress Analysis Over IP-to-IP Media Session, on page 419
Transcoding

Transcoding is a process of converting one voice codec to another. For example, transcoding iLBC-G.711 or iLBC-G.729.

**LTI based Transcoding**

- Internal API is used to access Digital Signaling Processor (DSP) resources for transcoding.
- Transcoding resources (DSPFARM) and CUBE need to be on the same platform.
- Only DSPFARM profile configuration is required. Skinny Client Control Protocol (SCCP) configuration configuration is not required.
- No TCP socket is opened and no registration is used.
- DSPFARM profile is associated to a new application type CUBE.

```
Device(config)# dspfarm profile 1 transcode
Device(config-dspfarm-profile)# associate application CUBE
```

- With LTI transcoding, higher performance is achieved since there is no need for extra SCCP legs and associated RTP streams. The performance will be in line high density mode offered with SCCP based transcoding.
- **crypto pki trustpoint** configuration is not required for Secure Real-time Transport Protocol (SRTP) to Real-time Transport Protocol (RTP) calls.

---

**Note**

Cisco Aggregated Services Routers 1000 Series (ASR 1K), Cisco Integrated Services Generation 2 Routers (ISR G2), and Cisco 4000 Series Integrated Services Routers (ISR G3) support LTI-based Transcoding.

---

**SCCP based Transcoding**

- Skinny Client Control Protocol (SCCP) protocol is used for controlling Digital Signaling Processor (DSP) resources used for transcoding.
- Transcoding resources (DSPFARM) and CUBE can be on different platforms.
- SCCP client (For example, **sc cp ccm** configuration and SCCP server (telephony service) configuration is required apart from DSPFARM profile configuration.
- DSPFARM registers with Cisco Unified Border Element over TCP Socket, using SCCP.

```
Device(config)# dspfarm profile 1 transcode
Device(config-dspfarm-profile)# associate application SCCP
```
• High density transcoding needs to be enabled for higher performance. High density transcoding will flow-around through the transcoder
• Secure Real-time Transport Protocol (SRTP) to Real-time Transport Protocol (RTP) using transcoder requires **crypto pki trustpoint** configuration to establish the Transport Layer Security (TLS) connection with SCCP server.

**Note**
Integrated Services Routers Generation 1 series and Integrated Services Routers Generation 2 Series devices support SCCP-based Transcoding only.

- Configuring LTI-Based Transcoding, on page 400
- Configuration Examples for LTI Based Transcoding, on page 401
- Configuring SCCP-based Transcoding (ISR-G2 devices only), on page 404
- TLS for SCCP Connection for DSP Services, on page 406
- Configuring Secure Transcoding, on page 407
- Configuration Examples for SCCP Based Transcoding, on page 415

**Configuring LTI-Based Transcoding**

**Note**
We recommend that you configure LTI-based Transcoding for Cisco Aggregated Services Routers (ASR), Cisco Integrated Services Generation 2 Routers (ISR G2), and Cisco 4000 Series Integrated Services Routers (ISR G3).

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice-card voice-interface-slot-number
4. dspfarm services dspfarm
5. exit
6. dspfarm profile profile-identifier transcode
7. codec codec
8. maximum sessions sessions
9. associate application CUBE
10. exit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>configure terminal</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device&gt; configure terminal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enters global configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>voice-card voice-interface-slot-number</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# voice-card 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configures a voice card and enters voice-card configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>ds pfarm services ds pfarm</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enable voice-only DSPFARM services on the Voice Card.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>exit</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exits the voice-card configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>ds pfarm profile profile-identifier transcode</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# ds pfarm profile 1 transcode</td>
<td></td>
</tr>
</tbody>
</table>
| | Enters a DSP farm profile configuration mode and defines a profile for DSP farm services.  
| | • **profile-identifier**- Number that uniquely identifies a profile. Range: 1 to 65535.  
| | • **transcode**- Enables profile for transcoding. |
| **Step 7** | **codec codec**  |
| **Example:** | Device(config-dspfarm-profile)# codec ilbc |
| | The codec rate to be attempted for SCCP-controlled connections. You can specify just the codec type, and the DSP uses the default codec parameter, such as independent mode, 32 kbps bit-rate, and 30 ms framesize. |
| **Step 8** | **maximum sessions sessions**  |
| | Configures maximum number of sessions. |
| **Step 9** | **associate application CUBE**  |
| **Example:** | Configures an application to the profile for LTI based transcoding. |
| **Step 10** | **exit**  |
| | Exits interface configuration mode. |

**Configuration Examples for LTI Based Transcoding**

**Example: LTI-based Transcoding**

```
! Enabling ds pfarm services under voice-card  
Device(config)# voice-card 0/1  
Device(config-voicecard)# ds pfarm  
Device(config-voicecard)# ds pfarm services ds pfarm  
Device(config-voicecard)# exit  

! Configuring ds pfarm profile  
Device(config)# ds pfarm profile 1 transcode  
Device(config-dspfarm-profile)# codec g711ulaw  
Device(config-dspfarm-profile)# codec g711alaw
```
Device(config-dspfarm-profile)# codec g729r8
Device(config-dspfarm-profile)# maximum sessions 10
Device(config-dspfarm-profile)# associate application CUBE
Device(config-dspfarm-profile)# exit

! Starting Service Engine
Device(config)# interface ServiceEngine0/1/0
Device(config-if)# no shutdown
Device(config-if)# exit

Example: Secure LTI-based Transcoding

!Client trustpoints use HTTP to receive certificate from CA.
Device(config)#ip http server

!Generate an RSA Keypair. 
! (This step generates Private and Public keys. In this example, CUBE is just a label. It can be anything.)
crypto key generate rsa general-keys label CUBE modulus 1024
The name for the keys will be: CUBE
% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 0 seconds)

!Configure IOS CA Server. In this example, CA Server is named cube-ca.
crypto pki server cube-ca
database level complete
no database archive
grant auto
lifetime certificate 1800
Secure-CUBE(cs-server)#no shut
%Some server settings cannot be changed after CA certificate generation.
% Please enter a passphrase to protect the private key
% or Type Return to exit
Password:
Re-enter password:
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 0 seconds)
% Certificate Server enabled.

!Create PKI trustpoints for cube for TLS communication.
crypto pki trustpoint CUBE-TLS
enrollment url http://X.X.X.X:80
serial-number none
fqdn none
ip-address none
subject-name CN=Secure-CUBE
revocation-check none
rsakeypair CUBE

!Authenticate the trustpoint with CA server and accept certificate of CA
crypto pki authenticate CUBE-TLS
Certificate has the following attributes:
  Fingerprint MD5: BCEBB5A1 1AC882F7 24BE476D 06537711
  Fingerprint SHA1: CE2FEEA5 42515B33 3EF6A8F6 7E31D6DF 8E32BE6
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.

!Enroll the trustpoint with CA server.
!In this step the CUBE receives a signed certificate from CA.
Secure-CUBE(config)#crypto pki enroll CUBE-TLS
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
  password to the CA Administrator in order to revoke your certificate.
  For security reasons your password will not be saved in the configuration.
  Please make a note of it.
Password:
Re-enter password:
%
% The subject name in the certificate will include: CN=Secure-CUBE
% The fully-qualified domain name will not be included in the certificate
Request certificate from CA? [yes/no]: yes
%
% Certificate request sent to Certificate Authority
% The 'show crypto pki certificate verbose CUBE-TLS' command will show the
  fingerprint.

!Configure TCP TLS as transport protocol
voice service voip
  sip
session transport tcp tls

!Assign trustpoint for sip-ua, this trustpoint is used for all SIP signaling between CUBE
  and CUCM.
sip-ua
crypto signaling remote-addr <cucm pub ip address> 255.255.255.255 trustpoint CUBE-TLS
crypto signaling remote-addr <cucm sub ip address> 255.255.255.255 trustpoint CUBE-TLS

!or or default trustpoint can be configured for all SIP signaling from CUBE.
sip-ua
crypto signaling default trustpoint CUBE-TLS

!Enable SRTP.
Voice service voip
srtp fallback

!Configure secure transcoder is required.

dspfarm profile 1 transcode universal security
codec g711ulaw
codec g711alaw
codec g729ar8
codec g729abr8
maximum sessions 10
associate application CUBE

Configuring SCCP-based Transcoding (ISR-G2 devices only)

SUMMARY STEPS

1. enable
2. configure terminal
3. voice-card voice-interface-slot-number
4. dspfarm
5. dsp service dspfarm
6. exit
7. telephony-service
8. sdspfarm units units
9. sdspfarm transcode sessions units
10. sdspfarm tag value Device-Name
11. max-phones max-phones-to-be-supported
12. max-dn max-directorynumbers-to-be-supported
13. ip source-address CUBE-internal-ipv4-address [port port-number]
14. exit
15. sccp local interface-type number
16. sccp ccm CUBE-internal-ipv4-address identifier identifier-number version version-number
17. sccp
18. sccp ccm group group-id
19. associate ccm CCM-identifier priority priority
20. associate profile profile-identifier register Device-Name
21. exit
22. dspfarm profile profile-id transcode
23. codec codec
24. maximum sessions sessions
25. associate application sccp
26. exit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device&gt; configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>voice-card voice-interface-slot-number</code></td>
<td>Configures a voice card and enters voice-card configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device(config)# voice-card 1</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>dspfarm</code></td>
<td>Enable voice card for DSP.</td>
</tr>
<tr>
<td>Step 5</td>
<td><code>dsp service dspfarm</code></td>
<td>Enable voice-only dspfarm services on the Voice Card.</td>
</tr>
<tr>
<td>Step 6</td>
<td><code>exit</code></td>
<td>Exits the voice-card configuration mode.</td>
</tr>
<tr>
<td>Step 7</td>
<td><code>telephony-service</code></td>
<td>Enters telephony-service configuration mode.</td>
</tr>
<tr>
<td>Step 8</td>
<td><code>sdspfarm units units</code></td>
<td>Define maximum number of dspfarm units.</td>
</tr>
<tr>
<td>Step 9</td>
<td><code>sdspfarm transcoding sessions units</code></td>
<td>Define maximum number of dspfarm transcoding session.</td>
</tr>
<tr>
<td>Step 10</td>
<td><code>sdspfarm tag value Device-Name</code></td>
<td>Configures a name for the transcoder.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device(config-telephony)# sdspfarm tag 1 CUBE-XCODE</code></td>
<td></td>
</tr>
<tr>
<td>Step 11</td>
<td><code>max-ephones max-phones-to-be-supported</code></td>
<td>Configures the maximum number of phones that are to be supported.</td>
</tr>
<tr>
<td>Step 12</td>
<td><code>max-dn max-directorynumbers-to-be-supported</code></td>
<td>Configures the maximum number of directories to be supported.</td>
</tr>
<tr>
<td>Step 13</td>
<td><code>ip source-address CUBE-internal-ipv4-address [port port-number]</code></td>
<td>Defines an IP address and port number for the telephony service.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device(config-telephony)# ip source-address 10.1.1.1 port 2000</code></td>
<td></td>
</tr>
<tr>
<td>Step 14</td>
<td><code>exit</code></td>
<td>Exits the telephony-service configuration mode.</td>
</tr>
<tr>
<td>Step 15</td>
<td><code>sccp local interface-type number</code></td>
<td>Configures the local gateway related parameters values.</td>
</tr>
<tr>
<td>Step 16</td>
<td><code>sccp ccm CUBE-internal-ipv4-address identifier identifier-number version version-number</code></td>
<td>Configures call manager related parameter values.</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>18</td>
<td>sccp ccm group group-id</td>
<td>Configures Call Manager Group and enters SCCP CCM configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)#sccp ccm group 1</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>associate ccm CCM-identifier priority priority</td>
<td>Configures Call Manager Group and enters SCCP CCM configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-sccp-ccm)# associate ccm 1 priority 1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>associate profile profile-identifier register Device-Name</td>
<td>Specifies the device name that needs to register.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-sccp-ccm)# associate profile 1 register CUBE-XCODE</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>exit</td>
<td>Exits SCCP CCM configuration mode.</td>
</tr>
<tr>
<td>22</td>
<td>dspfarm profile profile-id transcode</td>
<td>Configures a Transcoding profile and enters DSP profile configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# dspfarm profile 1 transcode</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>codec codec</td>
<td>The codec rate to be attempted for SCCP-controlled connections.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-dspfarm-profile)# codec ilbc</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>maximum sessions sessions</td>
<td>Configures maximum number of sessions.</td>
</tr>
<tr>
<td>25</td>
<td>associate application sccp</td>
<td>Configures an application to the profile for SCCP-based transcoding.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-dspfarm-profile)# associate application sccp</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>exit</td>
<td>Exits the telephony-service configuration mode.</td>
</tr>
</tbody>
</table>

**TLS for SCCP Connection for DSP Services**

The Cisco Unified Border Element supports Transport Layer Security (TLS) to be enabled or disabled between the Skinny Call Control Protocol (SCCP) server and the SCCP client. By default, TLS is enabled, which provides added protection at the transport level and ensures that SRTP keys are not easily accessible. Once TLS is disabled, the SRTP keys are not protected.

SRTP-RTP interworking is available with normal and universal transcoders. The transcoder on the Cisco Unified Border Element is invoked using SCCP messaging between the SCCP server and the SCCP client. SCCP messages carry the SRTP keys to the digital signal processor (DSP) farm at the SCCP client. The transcoder can be within the same router or can be located in a separate router. TLS should be disabled only when the transcoder is located in the same router. To disable TLS, configure the **no** form of the **tls** command in DSPFARM profile configuration mode. Disabling TLS improves CPU performance.
Configuring Secure Transcoding

Configuring the Certificate Authority

Perform the steps described in this section to configure the certificate authority.

SUMMARY STEPS

1. enable
2. configure terminal
3. ip http server
4. crypto pki server cs-label
5. database level complete
6. grant auto
7. no shutdown
8. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> ip http server</td>
<td>Enables the HTTP server on your IPv4 or IPv6 system,</td>
</tr>
<tr>
<td>Example:</td>
<td>including the Cisco web browser user interface.</td>
</tr>
<tr>
<td>Device(config)# ip http server</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> crypto pki server cs-label</td>
<td>Enables a Cisco IOS certificate server and enters certificate</td>
</tr>
<tr>
<td>Example:</td>
<td>server configuration mode.</td>
</tr>
<tr>
<td>Device(config)# crypto pki server 3854-cube</td>
<td>In the example, 3854-cube is specified as the name of the certificate server.</td>
</tr>
<tr>
<td><strong>Step 5</strong> database level complete</td>
<td>Controls what type of data is stored in the certificate</td>
</tr>
<tr>
<td>Example:</td>
<td>enrollment database.</td>
</tr>
<tr>
<td>Device(cs-server)# database level complete</td>
<td>• In the example, each issued certificate is written to the database.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 6</strong> grant auto</td>
<td>Specifies automatic certificate enrollment.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cs-server)# grant auto</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> no shutdown</td>
<td>Reenables the certificate server.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cs-server)# no shutdown</td>
<td>- Create and enter a new password when prompted.</td>
</tr>
<tr>
<td><strong>Step 8</strong> exit</td>
<td>Exits certificate server configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cs-server)# exit</td>
<td></td>
</tr>
</tbody>
</table>

### Configuring a Trustpoint for the Secure Universal Transcoder

Perform the task in this section to configure, authenticate, and enroll a trustpoint for the secure universal transcoder.

**Before you begin**

Before you configure a trustpoint for the secure universal transcoder, you should configure the certificate authority, as described in the Configuring the Certificate Authority, on page 407.

### SUMMARY STEPS

1. enable
2. configure terminal
3. crypto pki trustpoint name
4. enrollment url url
5. serial-number
6. revocation-check method
7. rsakeypair key-label
8. end
9. crypto pki authenticate name
10. crypto pki enroll name
11. exit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>- Enter your password if prompted.</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>crypto pki trustpoint name</td>
<td>Declares the trustpoint that the router uses and enters ca-trustpoint configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# crypto pki trustpoint secdsp</td>
<td>• In the example, the trustpoint is named secdsp.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>enrollment url url</td>
<td>Specifies the enrollment parameters of a certification authority (CA).</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(ca-trustpoint)# enrollment url <a href="http://10.13.2.52:80">http://10.13.2.52:80</a></td>
<td>• In the example, the URL is defined as <a href="http://10.13.2.52:80">http://10.13.2.52:80</a>.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>serial-number</td>
<td>Specifies whether the router serial number should be included in the certificate request.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(ca-trustpoint)# serial-number</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>revocation-check method</td>
<td>Checks the revocation status of a certificate.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(ca-trustpoint)# revocation-check crl</td>
<td>• In the example, the certificate revocation list checks the revocation status.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td>rsakeypair key-label</td>
<td>Specifies which key pair to associate with the certificate.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(ca-trustpoint)# rsakeypair 3845-cube</td>
<td>• In the example, the key pair 3845-cube generated during enrollment is associated with the certificate.</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Exits ca-trustpoint configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(ca-trustpoint)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td></td>
</tr>
<tr>
<td>crypto pki authenticate name</td>
<td>Authenticated the CA.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# crypto pki authenticate secdsp</td>
<td>• Accept the trustpoint CA certificate if prompted.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td></td>
</tr>
<tr>
<td>crypto pki enroll name</td>
<td>Obtains the certificate for the router from the CA.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# crypto pki enroll secdsp</td>
<td>• Create and enter a new password if prompted.</td>
</tr>
<tr>
<td></td>
<td>• Request a certificate from the CA if prompted.</td>
</tr>
</tbody>
</table>
### Configuring DSPFARM Services

For configuration steps, see [Configuring LTI-Based Transcoding](#).

### Associating SCCP to the Secure DSPFARM Profile

Perform the task in this section to associate SCCP to the secure DSPFARM profile.

**Before you begin**

Before you associate SCCP to the secure DSPFARM profile, you should configure DSPFARM services, as described in the “Configuring DSPFARM Services”.

#### SUMMARY STEPS

1. enable
2. configure terminal
3. sccp local interface-type interface-number
4. sccp ccm ip-address identifier identifier-number version version-number
5. sccp
6. associate ccm identifier-number priority priority-number
7. associate profile profile-identifier register device-name
8. dspfarm profile profile-identifier transcode universal security
9. trustpoint trustpoint-label
10. codec codec-type
11. Repeat Step 10 to configure required codecs.
12. maximum sessions number
13. associate application sccp
14. no shutdown
15. exit

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>* Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>

---

**Purpose**

**Command or Action**

<table>
<thead>
<tr>
<th>Step 11</th>
<th>exit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# exit</td>
<td></td>
</tr>
</tbody>
</table>

**Purpose**

Exits global configuration mode.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>configure terminal</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td>En ters global configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>sccp local interface-type interface-number</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# sccp local GigabitEthernet 0/0</td>
</tr>
<tr>
<td>Selects the local interface that SCCP applications (transcoding and conferencing) use to register with Cisco CallManager.</td>
<td></td>
</tr>
<tr>
<td>- In the example, the following parameters are set:</td>
<td></td>
</tr>
<tr>
<td>- GigabitEthernet is defined as the interface type that the SCCP application uses to register with Cisco CallManager.</td>
<td></td>
</tr>
<tr>
<td>- The interface number that the SCCP application uses to register with Cisco CallManager is specified as 0/0.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>sccp ccm ip-address identifier identifier-number version version-number</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# sccp ccm 10.13.2.52 identifier 1 version 5.0.1</td>
</tr>
<tr>
<td>Adds a Cisco Unified Communications Manager server to the list of available servers.</td>
<td></td>
</tr>
<tr>
<td>- In the example, the following parameters are set:</td>
<td></td>
</tr>
<tr>
<td>- 10.13.2.52 is configured as the IP address of the Cisco Unified Communications Manager server.</td>
<td></td>
</tr>
<tr>
<td>- The number 1 identifies the Cisco Unified Communications Manager server.</td>
<td></td>
</tr>
<tr>
<td>- The Cisco Unified Communications Manager version is identified as 5.0.1.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>sccp</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# sccp</td>
</tr>
<tr>
<td>Enables SCCP and related applications (transcoding and conferencing) and enters SCCP Cisco CallManager configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>associate ccm identifier-number priority priority-number</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-sccp-ccm)# associate ccm 1 priority 1</td>
</tr>
<tr>
<td>Associates a Cisco Unified CallManager with a Cisco CallManager group and establishes its priority within the group.</td>
<td></td>
</tr>
<tr>
<td>- In the example, the following parameters are set:</td>
<td></td>
</tr>
<tr>
<td>- The number 1 identifies the Cisco Unified CallManager.</td>
<td></td>
</tr>
<tr>
<td>- The Cisco Unified CallManager is configured with the highest priority within the Cisco CallManager group.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>associate profile profile-identifier register device-name</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Associates a DSPFARM profile with a Cisco CallManager group.</td>
</tr>
<tr>
<td>- In the example, the following parameters are set:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device(config-sccp-ccm)# associate profile 1 register sxcoder</td>
<td>- The number 1 identifies the DSPFARM profile. - Sxcoder is configured as the user-specified device name in Cisco Unified CallManager.</td>
</tr>
</tbody>
</table>

**Step 8**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dspfarm profile</strong> profile-identifier <strong>transcode universal security</strong></td>
<td>Defines a profile for DSPFARM services and enters DSPFARM profile configuration mode. - In the example, the following parameters are set: - Profile 1 is enabled for transcoding - Profile 1 is enabled for secure DSPFARM services.</td>
</tr>
</tbody>
</table>

**Example:**

Device(config-sccp-ccm)# dspfarm profile 1 transcode universal security

**Step 9**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>trustpoint</strong> trustpoint-label</td>
<td>Associates a trustpoint with a DSPFARM profile. - In the example, the trustpoint to be associated with the DSPFARM profile is labeled secdsp.</td>
</tr>
</tbody>
</table>

**Example:**

Device(config-dspfarm-profile)# trustpoint secdsp

**Step 10**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>codec</strong> codec-type</td>
<td>Specifies the codecs that are supported by a DSPFARM profile. - In the example, the g711ulaw codec is specified.</td>
</tr>
</tbody>
</table>

**Example:**

Device(config-dspfarm-profile)# codec g711ulaw

**Step 11**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeat Step 10 to configure required codecs.</td>
<td>--</td>
</tr>
</tbody>
</table>

**Step 12**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>maximum sessions</strong> number</td>
<td>Specifies the maximum number of sessions that are supported by the profile. - In the example, a maximum of 84 sessions are supported by the profile. The maximum number of sessions depends on the number of DSPs available for transcoding.</td>
</tr>
</tbody>
</table>

**Example:**

Device(config-dspfarm-profile)# maximum sessions 84

**Step 13**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>associate application</strong> sccp</td>
<td>Associates SCCP to the DSPFARM profile.</td>
</tr>
</tbody>
</table>

**Example:**

Device(config-dspfarm-profile)# associate application sccp

**Step 14**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>no shutdown</strong></td>
<td>Allocates DSPFARM resources and associates them with the application.</td>
</tr>
</tbody>
</table>

**Example:**

Device(config-dspfarm-profile)# no shutdown

**Step 15**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>exit</strong></td>
<td>Exits DSPFARM profile configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

Device(config-dspfarm-profile)# exit
Registering the Secure Universal Transcoder to the CUBE

Perform the task in this section to register the secure universal transcoder to the Cisco Unified Border Element. The Cisco Unified Border Element Support for SRTP-RTP Interworking feature supports both secure transcoders and secure universal transcoders.

Before you begin

Before you register the secure universal transcoder to the Cisco Unified Border Element, you should associated SCCP to the secure DSPFARM profile, as described in the Associating SCCP to the Secure DSPFARM Profile, on page 410.

SUMMARY STEPS

1. enable
2. configure terminal
3. telephony-service
4. sdspfarm transcode sessions number
5. sdspfarm tag number device-name
6. em logout time1 time2 time3
7. max-ephones max-ephones
8. max-dn max-directory-numbers
9. ip source-address ip-address
10. secure-signaling trustpoint label
11. tftp-server-credentials trustpoint label
12. create cnf-files
13. no sccp
14. sccp
15. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device&gt; configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> telephony-service</td>
<td>Enters telephony-service configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# telephony-service</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| 4    | **sdspfarm transcode sessions** *number* | Specifies the maximum number of transcoding sessions allowed per Cisco CallManager Express router.  
  - In the example, a maximum of 84 DSPFARM sessions are specified. |
|      | Example:         |         |
|      | Device(config-telephony)# *sdspfarm transcode sessions 84* |         |
| 5    | **sdspfarm tag** *number*  *device-name* | Permits a DSPFARM to be to registered to Cisco Unified CallManager Express and associates it with an SCCP client interface's MAC address.  
  - In the example, DSPFARM 1 is associated with the scoder device. |
|      | Example:         |         |
|      | Device(config-telephony)# *sdspfarm tag 1 scoder* |         |
| 6    | **em logout** *time1*  *time2*  *time3* | Configures three time-of-day-based timers for automatically logging out all Extension Mobility feature users.  
  - In the example, all users are logged out from Extension Mobility after 00:00. |
|      | Example:         |         |
|      | Device(config-telephony)# *em logout 0:0 0:0 0:0* |         |
| 7    | **max-ephones** *max-ephones* | Sets the maximum number of Cisco IP phones to be supported by a Cisco CallManager Express router.  
  - In the example, a maximum of four phones are supported by the Cisco CallManager Express router. |
|      | Example:         |         |
|      | Device(config-telephony)# *max-ephones 4* |         |
| 8    | **max-dn** *max-directory-numbers* | Sets the maximum number of extensions (ephone-dns) to be supported by a Cisco Unified CallManager Express router.  
  - In the example, a maximum of four extensions is allowed. |
|      | Example:         |         |
|      | Device(config-telephony)# *max-dn 4* |         |
| 9    | **ip source-address** *ip-address* | Identifies the IP address and port through which IP phones communicate with a Cisco Unified CallManager Express router.  
  - In the example, 10.13.2.52 is configured as the router IP address. |
|      | Example:         |         |
|      | Device(config-telephony)# *ip source-address 10.13.2.52* |         |
| 10   | **secure-signaling trustpoint** *label* | Specifies the name of the Public Key Infrastructure (PKI) trustpoint with the certificate to be used for TLS handshakes with IP phones on TCP port 2443.  
  - In the example, PKI trustpoint secdsp is configured. |
|      | Example:         |         |
|      | Device(config-telephony)# *secure-signaling trustpoint secdsp* |         |
| 11   | **tftp-server-credentials trustpoint** *label* | Specifies the PKI trustpoint that signs the phone configuration files.  
  - In the example, PKI trustpoint scme is configured. |
|      | Example:         |         |
### Purpose

Command or Action | Purpose
--- | ---
Device(config-telephony)# tftp-server-credentials trustpoint scme | Builds the XML configuration files that are required for IP phones in Cisco Unified CallManager Express.

**Step 12**

**create cnf-files**

Example:

Device(config-telephony)# create cnf-files

Disables SCCP and its related applications (transcoding and conferencing) and exits telephony-service configuration mode.

**Step 13**

**no sccp**

Example:

Device(config-telephony)# no sccp

Enables SCCP and related applications (transcoding and conferencing).

**Step 14**

**sccp**

Example:

Device(config)# sccp

Exits global configuration mode.

**Step 15**

**end**

Example:

Device(config)# end

---

### Configuration Examples for SCCP Based Transcoding

**Example: SCCP-based Transcoding**

```bash
! Enabling dspfarm services under voice-card
Device(config)# voice-card 1
Device(config-voicecard)# dspfarm
Device(config-voicecard)# dsp services dspfarm
Device(config-voicecard)# exit

! Configuring Telephony Service
Device(config)# telephony-service
Device(config-telephony)# sdspparm units 1
Device(config-telephony)# sdspparm transcode sessions 128
Device(config-telephony)# sdspparm tag 1 CUBE-XCODE
Device(config-telephony)# max-ephones 10
Device(config-telephony)# max-dn 10
Device(config-telephony)# ip source-address 10.1.1.1 port 2000
Device(config-telephony)# exit

! Configuring SCCP
Device(config)# no sccp
Device(config)# sccp local GigabitEthernet0/0
Device(config)# sccp ccm 10.1.1.1 identifier 1 version 4.0
```
Device(config)# sccp
Device(config)# sccp ccm group 1
Device(config-sccp-ccm)# associate ccm 1 priority 1
Device(config-sccp-ccm)# associate profile 1 register CUBE-XC0DE
Device(config-sccp-ccm)# exit

! Configuring dspfarm profile
Device(config)# dspfarm profile 1 transcode
Device(config-dspfarm-profile)# codec g711ulaw
Device(config-dspfarm-profile)# codec g711alaw
Device(config-dspfarm-profile)# codec g729u8
Device(config-dspfarm-profile)# maximum sessions 10

Device(config-dspfarm-profile)# associate application SCCP
Device(config-dspfarm-profile)# exit
Transrating

Transrating is a process of configuring a different packetization for a voice codec. For example, transrating G.729 20ms to G.729 30ms.

- Voice Packetization, on page 417
- Configuring Transrating for a Codec, on page 418

Voice Packetization

After the voice wavelength is digitized, the DSP collects the digitized data for an amount of time until there is enough data to fill the payload of a single packet.

With G.711, either 20 ms or 30 ms worth of voice is transmitted in a single packet. 20 ms worth of voice corresponds to 160 samples per packet. With 20 ms worth of voice per packet, 50 packets are created per second: 1 sec / 20 ms = 50.

The packetization rate has a direct effect on the total amount of bandwidth needed. More packets require more headers, and each header adds 40 bytes to the packet. The Table 12: Codec and Bandwidth Information, on page 54 table shows the effect of packetization rates on bandwidth utilization.

Codecs such as G.729 also compress the digitized output. G.729 creates a codeword for every 10 ms of voice. This “codeword” is a predefined representation of a 10-ms sample of human voice. Two codewords are contained in each packet at 50 packets per second or three codewords at 33.3 packets per second. Because the codewords need fewer bits, the overall bandwidth required is reduced.

<table>
<thead>
<tr>
<th>Supported Codecs</th>
<th>Packetization (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.711 a-law 64 Kbps</td>
<td>10, 20, 30</td>
</tr>
<tr>
<td>G.711 law 64 Kbps</td>
<td>10, 20, 30</td>
</tr>
<tr>
<td>G.723 5.3/6/3 Kbps</td>
<td>30, 60</td>
</tr>
<tr>
<td>G.729, G.729A, G.729B, G.729AB 8 Kbps</td>
<td>10, 20, 30, 40, 50, 60</td>
</tr>
<tr>
<td>G.722—64 Kbps</td>
<td>10, 20, 30</td>
</tr>
</tbody>
</table>
## Configuring Transrating for a Codec

### SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice number voip
4. codec codec-name bytes voice-payload-size [fixed-bytes]
5. end

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice number voip</td>
<td>Enters dial peer configuration mode for the specified VoIP dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# dial-peer voice 1 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> codec codec-name bytes voice-payload-size [fixed-bytes]</td>
<td>Configures a different packetizations for a voice codec.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# codec g729r8 bytes 30 fixed-byte</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 32

Call Progress Analysis Over IP-to-IP Media Session

The Call Progress Analysis Over IP-IP Media Session feature enables the detection of automated answering systems and live human voices on outbound calls and communicates the detected information to the external application. Typically, call progress analysis (CPA) is extensively used in contact center deployments in conjunction with the outbound Session Initiation Protocol (SIP) dialer, where CPA is enabled on the Cisco Unified Border Element (Cisco UBE), and digital signal processors (DSP) perform the CPA functionality.

- Feature Information for Call Progress Analysis Over IP-IP Media Session, on page 419
- Restrictions for Call Progress Analysis Over IP-to-IP Media Session, on page 420
- Information About Call Progress Analysis Over IP-IP Media Session, on page 421
- How to Configure Call Progress Analysis Over IP-to-IP Media Session, on page 422
- Configuration Examples for the Call Progress Analysis Over IP-to-IP Media Session, on page 425

Feature Information for Call Progress Analysis Over IP-IP Media Session

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Restrictions for Call Progress Analysis Over IP-to-IP Media Session

- Only SIP-to-SIP Early Offer (EO-to-EO) call flows are supported.
- Session Description Protocol (SDP) passthrough and flow-around media calls are not supported.
- Only the G711 flavor of codec is supported.
- High Availability (HA) is not supported.
- Skinny Client Control Protocol (SCCP)-based digital signal processor (DSP) farm is not supported.
- CPA cannot not be detected if Dialer uses Inband as DTMF relay mechanism, that is, Inband to RTP-NTE DTMF inter-working is not supported with CPA.
- CPA call record is not supported for "180 without SDP" and "Direct Call Connect (without 18x)" call flows from Service Provider.
Information About Call Progress Analysis Over IP-IP Media Session

Call Progress Analysis

Call progress analysis (CPA) is a DSP algorithm that analyzes the Real-Time Transport Protocol (RTP) voice stream to look for special information tones (SIT), fax or modem tones, human speech, and answering machine tones. CPA also passes the voice information to Cisco IOS or Cisco Unified Border Element (Cisco UBE).

CPA is initiated on receiving a new SIP INVITE with x-cisco-cpa content. While a call is in progress, the DSP or the Xcoder analyzes the incoming voice or media stream. The DSP identifies the type of voice stream based on statistical voice patterns or specific tone frequencies and provides the information to the Cisco UBE. The Cisco UBE notifies the dialer with a SIP UPDATE with x-cisco-cpa content along with the detected event. Based on the report, the caller (dialer) can decide to either transfer the call or terminate the call.

To use the CPA functionality, you must enable CPA and configure CPA timing and threshold parameters.

Table 49: x-cisco-cpa content meaning

<table>
<thead>
<tr>
<th>SIP Message</th>
<th>Direction of Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>18x or 200</td>
<td>Cisco IOS to dialer</td>
<td>Cisco UBE informs the dialer if CPA is enabled for a call or not.</td>
</tr>
<tr>
<td>New INVITE</td>
<td>Dialer to Cisco IOS</td>
<td>Dialer requests Cisco IOS or the Cisco UBE to activate the CPA algorithm for this session.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Cisco IOS to dialer</td>
<td>Cisco IOS or the Cisco UBE notifies the dialer about the detected event.</td>
</tr>
</tbody>
</table>

CPA Events

Table 50: CPA Event Detection List

<table>
<thead>
<tr>
<th>CPA Event</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asm</td>
<td>Answer machine</td>
</tr>
<tr>
<td>AsmT</td>
<td>Answer machine terminate tone</td>
</tr>
<tr>
<td>CpaS</td>
<td>Start of the Call Progress Analysis</td>
</tr>
<tr>
<td>FT</td>
<td>Fax/Modem tone</td>
</tr>
<tr>
<td>LS</td>
<td>Live human speech</td>
</tr>
<tr>
<td>LV</td>
<td>Low volume or dead air call</td>
</tr>
</tbody>
</table>
How to Configure Call Progress Analysis Over IP-to-IP Media Session

Enabling CPA and Setting the CPA Parameters

Perform the following task to enable CPA and set the CPA timing and threshold parameters:

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `dspfarm profile profile-identifier transcode`
4. `call-progress-analysis`
5. `exit`
6. `voice service voip`
7. `cpa timing live-person max-duration`
8. `cpa timing term-tone max-duration`
9. `cpa threshold active-signal signal-threshold`
10. `end`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dsfpfarm profile profile-identifier transcode</td>
<td>Enters DSP farm profile configuration mode, defines a profile for DSP farm services, and enables the profile for transcoding.</td>
</tr>
<tr>
<td>Example: Device(config)# dsfpfarm profile 15 transcode</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> call-progress-analysis</td>
<td>Enables call progress analysis (CPA) on Cisco UBE.</td>
</tr>
<tr>
<td>Example: Device(config-dspfarm-profile)# call-progress-analysis</td>
<td>• You must configure this command to activate the CPA feature and set CPA parameters.</td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits DSP farm profile configuration mode and enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config-dspfarm-profile)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> voice service voip</td>
<td>Enters voice service configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> cpa timing live-person max-duration</td>
<td>(Optional) Sets the maximum waiting time (in milliseconds) that the CPA algorithm uses to determine if a call is answered by a live human.</td>
</tr>
<tr>
<td>Example: Device(conf-voi-serv)# cpa timing live-person 2501</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> cpa timing term-tone max-duration</td>
<td>(Optional) Sets the maximum waiting time (in milliseconds) that the CPA algorithm uses to wait for the answering machine termination tone after the answering machine is detected.</td>
</tr>
<tr>
<td>Example: Device(conf-voi-serv)# cpa timing term-tone 15500</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> cpa threshold active-signal signal-threshold</td>
<td>(Optional) Sets the threshold (in decibels) of an active signal that is related to the measured noise floor level.</td>
</tr>
<tr>
<td>Example: Device(conf-voi-serv)# cpa threshold active-signal 18db</td>
<td>• If a signal threshold configured by this command is greater than the measured noise floor level, then the signal is considered as active. The active signal thresholds that you can configure are 9, 12, 15, 18, and 21 decibels.</td>
</tr>
<tr>
<td><strong>Step 10</strong> end</td>
<td>Exits voice service configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device(conf-voi-serv)# end</td>
<td></td>
</tr>
</tbody>
</table>
Verifying the Call Progress Analysis Over IP-to-IP Media Session

Perform this task to verify that call progress analysis has been configured for a digital signal processor (DSP) farm profile.

**SUMMARY STEPS**

1. `enable`
2. `show dspfarm profile profile-identifier`

**DETAILED STEPS**

**Step 1**  
`enable`  
Enables privileged EXEC mode.  
**Example:**  
Device> `enable`

**Step 2**  
`show dspfarm profile profile-identifier`  
Displays the configured DSP farm profile information for a selected Cisco Call Manager group. In the following sample output, the Call Progress Analysis field shows that CPA is enabled.  
**Example:**  
Device# `show dspfarm profile 3`

```
Profile ID = 3, Service=Universal TRANSCODING, Resource ID = 3  
Profile Description:  
Profile Service Mode : Non Secure  
Profile Admin State : UP  
Profile Operation State : ACTIVE  
Application : CUBE  
Status : ASSOCIATED  
Resource Provider : FLEX_DSPRM  
Status : UP  
Number of Resource Configured : 4  
Number of Resources Out of Service : 0  
Number of Resources Active : 0  
Codec Configuration: num_of_codecs:4  
Codec : g711ulaw, Maximum Packetization Period : 30  
Codec : g711alaw, Maximum Packetization Period : 30  
Codec : g729ar8, Maximum Packetization Period : 60  
Codec : g729abr8, Maximum Packetization Period : 60  
Noise Reduction : ENABLED  
Call Progress Analysis : ENABLED
```

**Troubleshooting Tips**

Use the following commands to troubleshoot the call progress analysis for SIP-to-SIP calls:

- `debug ccsip all`
- `debug voip ccap inout`
• debug voip hpi all
• debug voip ipipgw
• debug voip media resource provisioning all

Configuration Examples for the Call Progress Analysis Over IP-to-IP Media Session

Example: Enabling CPA and Setting the CPA Parameters

The following example shows how to enable CPA and set a few timing and threshold parameters. Depending on your requirements, you can configure more timing and threshold parameters.

Device> enable
Device# configure terminal
Device(config)# dspfarm profile 15 transcode
Device(config-dspfarm-profile)# call-progress-analysis
Device(config-dspfarm-profile)# exit
Device(config)# voice service voip
Device(conf-voi-serv)# cpa timing live-person 2501
Device(conf-voi-serv)# cpa timing term-tone 15500
Device(conf-voi-serv)# cpa threshold active-signal 18db
Device(conf-voi-serv)# end
Example: Enabling CPA and Setting the CPA Parameters
PART VI

Video

• Video Suppression, on page 429
CHAPTER 33

Video Suppression

The video suppression feature allows pass-through of only audio and image (for T.38 Fax) media types in SDP and drops all other media capabilities.

• Feature Information for Video Suppression, on page 429
• Restrictions, on page 429
• Information About Video Suppression, on page 430
• Configuring Video Suppression, on page 430
• Troubleshooting Tips, on page 431

Feature Information for Video Suppression

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Video Suppression</td>
<td>Cisco IOS 15.6(2)T</td>
<td>This feature allows pass-through of only audio and image media types and drops all</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE Denali 16.3.1</td>
<td>other media types in SDP. The following commands are introduced: audio forced, voice-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>class sip audio forced</td>
</tr>
</tbody>
</table>

Restrictions

• Supports only SIP-SIP calls.
• Video suppression is not supported in SDP pass-through mode.
• Video suppression feature removes both video and application m-lines in the incoming SDP. It is not possible to remove application m-line alone and pass across video m-line parameters.

Information About Video Suppression

Video suppression feature enables CUBE to interwork with the networks that support only audio and image media types in SDP and the networks that support video and application media types in addition to audio and image media types.

By default video suppression feature is disabled on CUBE and hence the video capabilities are passed through in SDP. Passing across the video capabilities could cause interoperability issues if one of the networks do not support video capabilities.

By enabling video suppression feature, you can configure CUBE to pass-through audio and image only, and drop all other capabilities such as video and application m-lines. This helps enterprises to interwork with audio capable networks and video capable networks smoothly.

You can enable video suppression at dial-peer level and at global configuration level.

Feature Behavior

• If video suppression is enabled on any of the dial-peers (inbound or outbound), video capabilities are not offered for that particular call.

• Configuring `voice-class sip audio forced [system]` command at a dial-peer level makes use of global configuration level settings for allowing only audio and image media.

• Video suppression feature will work as expected even when codec transparent feature is configured.

Configuring Video Suppression

SUMMARY STEPS

1. enable
2. configure terminal
3. Enter one of the following commands:
   • In the dial-peer configuration mode
     `voice-class sip audio forced`
   • In the global VoIP SIP configuration mode
     `audio forced`
4. end
**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>Enter one of the following commands:</td>
<td>Enables pass-through of only audio and image media types in SDP.</td>
</tr>
<tr>
<td>• In the dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>voice-class sip audio forced</td>
<td></td>
</tr>
<tr>
<td>• In the global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>audio forced</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>!Applying audio-forced to one dial peer only</td>
<td></td>
</tr>
<tr>
<td>Device (config)# dial-peer voice 10 voip</td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer)# voice-class sip audio forced</td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>! Applying audio forced globally</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device (config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td>Device (config-voi-sip)# audio forced</td>
<td></td>
</tr>
<tr>
<td>Device (config-voi-sip)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Exits present configuration mode and enters privileged EXEC mode.</td>
</tr>
</tbody>
</table>

**Troubleshooting Tips**

The following commands are useful for debugging:

- show voip rtp connections
- show call active voice brief
- show call active video brief
- debug voip dialpeer
- debug ccsip all
- debug voip ccaapi inout
PART VII

Media Services

• Configuring RTCP Report Generation, on page 435
CHAPTER 34

Configuring RTCP Report Generation

The assisted Real-time Transport Control Protocol (RTCP) feature adds the ability for Cisco Unified Border Element (Cisco UBE) to generate standard RTCP keepalive reports on behalf of endpoints. RTCP reports determine the liveliness of a media session during prolonged periods of silence, such as call hold or mute. Therefore, it is important for the Cisco UBE to generate RTCP reports irrespective of whether the endpoints send or receive media.

Cisco UBE generates RTCP report only when inbound and outbound call legs are SIP, or SIP to H.323, or H.323 to SIP.

- Finding Feature Information, on page 435
- Prerequisites, on page 435
- Restrictions, on page 436
- Configuring RTCP Report Generation on Cisco UBE, on page 436
- Troubleshooting Tips, on page 437
- Feature Information for Configuring RTCP Report Generation, on page 438

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites

Cisco Unified Border Element

- Cisco IOS Release 15.1(2)T or a later release must be installed and running on your Cisco Unified Border Element.
Cisco Unified Border Element (Enterprise)

- Cisco IOS XE Release 3.17S or a later release must be installed and running on your Cisco ASR 1000 Series Router and Cisco ISR 4000 Series Router.

Restrictions

- RTCP report generation over IPv6 is not supported.
- RTCP report generation is not supported for Secure Real-time Transport Protocol (SRTP) or SRT Control Protocol (SRTCP) pass-through as Cisco UBE is not aware of the media encryption or decryption keys.
- RTCP report generation is not supported for loopback calls, T.38 fax, and modem relay calls.
- RTCP or SRTCP report generation is not supported when Cisco UBE inserts a Digital Signal Processor (DSP) for RTP-SRTP interworking on RTP and SRTP call legs.
- RTCP report generation is not supported when there is a call hold with an invalid media address such as 0.0.0.0 in Session Description Protocol (SDP) or Open Logical Channel (OLC).
- RTCP report generation is not supported for RTCP multiplexed with RTP on the same address and port.
- RTCP report generation is not supported on enterprise aggregation services routers (ASRs) and 4000 series integrated services routers (ISRs) when Media Termination Points are collocated with the Cisco Unified Border Element. It affects RFC2833 and RFC4733 DTMF generation when MTP is used for DTMF conversion from Out-of-Band (OOB) to RFC2833 or RFC4733.
- RTCP packet generation is not supported on the SIP leg when the H.323 leg puts the SIP leg on hold in a Slow Start to Delayed-Offer call.

Configuring RTCP Report Generation on Cisco UBE

RTCP keepalive packets indicate session liveliness. When configured on Cisco UBE, RTCP keepalive packets are sent on both inbound and outbound SIP or H.323 call legs.

Perform this task to configure RTCP report generation on Cisco UBE.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. allow-connections from-type to-type
5. rtcp keepalive
6. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
</tbody>
</table>
### Purpose

#### Command or Action

**Example:**

Router> enable

- Enters global configuration mode.

**Step 2**

configure terminal

**Example:**

Router# configure terminal

- Enters voice service configuration mode.

**Step 3**

voice service voip

**Example:**

Router(config)# voice service voip

- Allows connection between SIP endpoints in a VoIP network.

**Step 4**

allow-connections from-type to to-type

**Example:**

Router(config-voi-serv)# allow-connections sip to sip

**Step 5**

rtp keepalive

**Example:**

Router(config-voi-serv)# rtp keepalive

- Configures RTCP keepalive report generation.

**Step 6**

end

**Example:**

Router(config-voi-serv)# end

- Exits voice service configuration mode and returns to privileged EXEC mode.

### Troubleshooting Tips

Use the following debug commands for debugging related to RTCP keepalive packets:

- **debug voip rtp packet** -- Shows details related to RTCP keepalive packets such as RTCP sending and receiving paths, Call ID, Globally Unique Identifier (GUID), packet header, and so on.

```
Router# debug voip rtp packet
01:06:27.450: //6/xxxxxxxxxxxx/RTCP//Event/voip_rtp_send_rtcp_keepalive: Generate RTCP Keepalive
*Mar 17 01:06:27.450: rtcp_send_report: Attributes
    (src ip=192.168.30.3, src port=17101, dst ip=192.168.30.4, dst port=18619
    bye=0, initial=1, ssrc=0x07111E02, keepalive=1)
    (rtcp=0x2E5AF214, ssrc=0x07111E02, source->ssrc=0x00001E03, total_len=36)
    2E5AF210: 80C90001 07111E02 81CA0006 .I.......J..
    2E5AF220: 07111E02 010F302E 302E3040 392E3435 ......0.0.0@9.45
    2E5AF230: 2E33302E 33000000 00 .30.3....
```
Under moderate traffic loads, the `debug voip rtp packet` command produces a high volume of output and the command should be enabled only when the call volume is very low.

- **debug voip rtp packet** -- Shows details about VoIP RTP packet debugging trace.

  ```
  Router# debug voip rtp packet
  VOIP RTP All Packets debugging is on
  ```

- **debug voip rtp session** -- Shows all RTP session debug information.

  ```
  Router# debug voip rtp session
  VOIP RTP All Events debugging is on
  ```

- **debug voip rtp error** -- Shows details about debugging trace for RTP packet error cases.

  ```
  Router# debug voip rtp error
  VOIP RTP Errors debugging is on
  ```

- **debug ip rtp protocol** -- Shows details about RTP protocol debugging trace.

  ```
  Router# debug ip rtp protocol
  RTP protocol debugging is on
  ```

- **debug voip rtcp session** -- Shows all RTCP session debug information.

  ```
  Router# debug voip rtcp session
  VOIP RTCP Events debugging is on
  ```

- **debug voip rtcp error** -- Shows details about debugging trace for RTCP packet error cases.

  ```
  Router# debug voip rtcp error
  VOIP RTCP Errors debugging is on
  ```

### Feature Information for Configuring RTCP Report Generation

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

Feature History Table entry for the Cisco Unified Border Element.
### Table 52: Feature Information for Configuring RTCP Report Generation

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assisted RTCP</td>
<td>15.1(2)T</td>
<td>This feature adds the ability for Cisco UBE to generate standard RTCP keepalive reports on behalf of endpoints and ensures the liveliness of a media session during prolonged periods of silence, such as call hold. The following commands were introduced or modified in this release: <code>rtcp keepalive</code>, <code>debug voip rtp</code>, <code>debug voip rtp</code>, <code>debug ip rtp protocol</code>, and <code>ip rtcp report interval</code>.</td>
</tr>
</tbody>
</table>

Feature History Table entry for the Cisco Unified Border Element (Enterprise).

### Table 53: Feature Information for Configuring RTCP Report Generation

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assisted RTCP</td>
<td>IOS XE Release 3.17S</td>
<td>This feature adds the ability for Cisco UBE to generate standard RTCP keepalive reports on behalf of endpoints and ensures the liveliness of a media session during prolonged periods of silence, such as call hold. The following commands were introduced or modified in this release: <code>rtcp keepalive</code>, <code>debug voip rtp</code>, <code>debug voip rtp</code>, <code>debug ip rtp protocol</code>, and <code>ip rtcp report interval</code>.</td>
</tr>
</tbody>
</table>
Feature Information for Configuring RTCP Report Generation
PART VIII

Media Recording

• Network-Based Recording, on page 443
• SIPREC (SIP Recording), on page 469
• Video Recording - Additional Configurations, on page 493
• Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 499
• Cisco Unified Communications Gateway Services--Extended Media Forking, on page 507
CHAPTER 35

Network-Based Recording

The Network-Based Recording feature supports software-based forking for Real-time Transport Protocol (RTP) streams. Media forking provides the ability to create midcall multiple streams (or branches) of audio and video associated with a single call and then send the streams of data to different destinations. To enable network-based recording using Cisco Unified Border Element (CUBE), you can configure specific commands or use a call agent. CUBE acts as a recording client and MediaSense Session Initiation Protocol (SIP) recorder acts a recording server.

- Feature Information for Network-Based Recording, on page 443
- Restrictions for Network-Based Recording, on page 444
- Information About Network-Based Recording Using CUBE, on page 445
- How to Configure Network-Based Recording, on page 448
- Additional References for Network-Based Recording, on page 468

Feature Information for Network-Based Recording

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 54: Feature Information for Network-Based Recording

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Readiness Criteria (SRC)—Modified the command <strong>show sip-ua calls</strong>.</td>
<td>Cisco IOS XE Gibraltar Release 16.11.1a</td>
<td>Command <strong>show sip-ua calls</strong> is modified to display local crypto key and remote crypto key.</td>
</tr>
<tr>
<td>Audio-only Stream Forking of Video Call</td>
<td>Cisco IOS 15.4(3)M Cisco IOS XE 3.13S</td>
<td>The Audio-only Stream Forking of Video Call feature supports CUBE-based forking and recording of only audio calls in a call that includes both audio and video. The following commands were introduced: <strong>media-type audio</strong>.</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Network-Based Recording of Video Calls Using CUBE</td>
<td>Cisco IOS 15.3(3)M, Cisco IOS XE 3.10S</td>
<td>The Network-Based Recording of Video Calls using CUBE feature supports forking and recording of video calls.</td>
</tr>
<tr>
<td>Network-Based Recording of Audio Calls Using CUBE</td>
<td>Cisco IOS 15.2(1)T, Cisco IOS XE 3.8S</td>
<td>The Network-Based Recording of Audio Calls using CUBE feature supports forking for RTP streams. The following commands were introduced or modified: media class, media profile recorder, media-recording, recorder parameter, recorder profile, show voip recmsp session.</td>
</tr>
</tbody>
</table>

**Restrictions for Network-Based Recording**

- Network-based recording is not supported for the following calls:
  - Calls that do not use Session Initiation Protocol (SIP). Must be a SIP-to-SIP call flow
  - Flow-around calls
  - Session Description Protocol (SDP) pass-through calls
  - Real-time Transport Protocol (RTP) loopback calls
  - High-density transcoder calls
  - IPv6-to-IPv6 calls
  - IPv6-to-IPv4 calls with IPv4 endpoint.
  - Secure Real-time Transport Protocol (SRTP) passthrough calls
  - SRTP-RTP calls with forking for SRTP leg (forking is supported for the RTP leg)
  - Resource Reservation Protocol (RSVP)
  - Multicast music on hold (MOH)

- Any media service parameter change via Re-INVITE or UPDATE from Recording server is not supported Midcall renegotiation and supplementary services can be done through the primary call only.

- Media service parameter change via Re-INVITE or UPDATE message from the recording server is not supported

- Recording is not supported if CUBE is running a TCL IVR application with the exception of survivability.tcl, which is supported with network based recording.

- Media mixing on forked streams is not supported

- Digital Signal Processing (DSP) resources are not supported on forked legs
Restrictions for Video Recording

- If the main call has multiple video streams (m-lines), the video streams other than the first video m-line are not forked.
- Application media streams of the primary call are not forked to the recording server.
- Forking is not supported if the anchor leg or recording server is on IPv6.
- High availability is not supported on forked video calls.

Information About Network-Based Recording Using CUBE

Deployment Scenarios for CUBE-based Recording

CUBE as a recording client has the following functions:

- Acts as a SIP user agent and sets up a recording session (SIP dialog) with the recording server.
- Acts as the source of the recorded media and forwards the recorded media to the recording server.
- Sends information to a server that helps the recording server associate the call with media streams and identifies the participants of the call. This information sent to the recording server is called metadata.

Note

CUBE simply forwards the RTP streams it receives to the SIP recorder. It does not support omitting any pre-agent VRU activity from the recording.

If you want to omit the VRU segment from a recording, you must use the Unified CVP to route the agent segment of the call back through CUBE. To do this, you need to separate ingress and media forking function from one another, which means you must either route the call through the ingress router a second time, or route it through a second router.

Given below is a typical deployment scenario of a CUBE-based recording solution. The information flow is described below:

*Figure 38: Deployment Scenario for CUBE-based Recording Solution*

1. Incoming call from SIP trunk.
2. Outbound call to a Contact Centre
3. Media between endpoints flowthrough CUBE

4. CUBE sets up a new SIP session with MediaSense based on policy.

5. CUBE forks RTP media to MediaSense. For an audio call, audio is forked. For a video call, both audio and video are forked. For an audio-only configuration in a audio-video call, only audio is forked. There will be two or four m-lines to the recording server, based on the type of recording.

The metadata carried in the SIP session between the recording client and the recording server is to:
- Carry the communication session data that describes the call.
- Send the metadata to the recording server. The recording server uses the metadata to associate communication sessions involving two or more participants with media streams.

The call leg that is created between the recording client and the recording server is known as the recording session.

**Open Recording Architecture**

The Open Recording Architecture (ORA) comprises of elements, such as application management server and SIP bridge, to support IP-based recording. The ORA IP enables recording by solving topology issues, which accelerates the adoption of Cisco unified communication solutions.

Following are the three layers of the ORA architecture:
Network Layer

The ORA network layer is comprised of call control systems, media sources, and IP foundation components, such as routers and switches.

Capture and Media Processing Layer

The ORA capture and media processing layer includes core functions of ORA—terminating media streams, storage of media and metadata, and speech analytics that can provide real-time events for applications.

Application Layer

The ORA application layer supports in-call and post-call applications through open programming interfaces. In-call applications include applications that make real-time business decisions (for example, whether to record a particular call or not), control pause and resume from Interactive Voice Response (IVR) or agent desktop systems, and perform metadata tagging and encryption key exchange at the call setup. Post-call applications include the following:

- Traditional compliance search, replay, and quality monitoring.
- Advanced capabilities, such as speech analytics, transcription, and phonetic search.
- Custom enterprise integration.
- Enterprise-wide policy management.

Media Forking Topologies

The following topologies support media forking:

Media Forking with Cisco UCM

The figure below illustrates media forking with Cisco Unified CallManager (Cisco UCM) topology. This topology supports replication of media packets to allow recording by the caller agent. It also enables CUBE to establish full-duplex communication with the recording server. In this topology, SIP recording trunk is enhanced to have additional call metadata.
Media Forking without Cisco UCM

The topology below shows media forking without the Cisco UCM topology. This topology supports static configuration on CUBE and the replication of media packets to allow recording caller-agent and full-duplex interactions at an IP call recording server.

SIP Recorder Interface

SIP is used as a protocol between CUBE and the MediaSense SIP server. Extensions are made to SIP to carry the recording session information needed for the recording server. This information carried in SIP sessions between the recording client and the recording server is called metadata.

Metadata

Metadata is the information that is passed by the recording client to the recording server in a SIP session. Metadata describes the communication session and its media streams.

Metadata is used by the recording server to:

- Identify participants of the call.
- Associate media streams with the participant information. Each participant can have one or more media streams, such as audio and video.
- Identify the participant change due to transfers during the call.

The recording server uses the metadata information along with other SIP message information, such as dialog ID and time and date header, to derive a unique key. The recording server uses this key to store media streams and associate the participant information with the media streams.

How to Configure Network-Based Recording

Configuring Network-Based Recording (with Media Profile Recorder)

SUMMARY STEPS

1. enable
2. configure terminal
3. media profile recorder profile-tag
4. (Optional) media-type audio
5. media-recording dial-peer-tag [dial-peer-tag2...dial-peer-tag5]
### Media Recording

#### Configuring Network-Based Recording (with Media Profile Recorder)

6. exit
7. media class tag
8. recorder profile tag
9. exit
10. dial-peer voice dummy-recorder-dial-peer-tag voip
11. media-class tag
12. destination-pattern [+ string] [T]
13. session protocol sipv2
14. session target ipv4:[recording-server-destination-address | recording-server-dns]
15. session transport tcp
16. end

### Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> media profile recorder profile-tag</td>
<td>Configures the media profile recorder and enters media profile</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>configuration mode.</td>
</tr>
<tr>
<td>Device(config)# media profile recorder 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> (Optional) media-type audio</td>
<td>Configures recording of audio only in a call with both audio and</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>video. If this configuration is not done, both audio and video are</td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# media-type audio</td>
<td>recorded.</td>
</tr>
<tr>
<td><strong>Step 5</strong> media-recording dial-peer-tag [dial-peer-tag2...dial-peer-tag5]</td>
<td>Configures the dial-peers that need to be configured.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Note</strong> You can specify a maximum of five dial-peer tags.</td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# media-recording 8000 8001 8002</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# exit</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><code>media class tag</code></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><code>recorder profile tag</code></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><code>exit</code></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><code>dial-peer voice dummy-recorder-dial-peer-tag voip</code></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td><code>media-class tag</code></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td><code>destination-pattern [+] string [T]</code></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

The predefined valid entries for `string` are the digits 0 to 9, the letters A to F and, the following special characters:

- The asterisk (*) and pound sign (#) that appear on standard touch-tone dial pads.
- Plus sign (+), which indicates that the preceding digit occurred one or more times.
- Backslash symbol (\), which is followed by a single character, and matches that character.

Media Forking functionality does not work with the wildcard entries other than the predefined set.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 13</strong></td>
<td></td>
</tr>
<tr>
<td>session protocol sipv2</td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td></td>
</tr>
<tr>
<td>session target ipv4:[recording-server-destination-address</td>
<td>Specifies a network-specific address for a dial peer.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session target ipv4:10.42.29.7</td>
<td>Keyword and argument are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ipv4: destination address --IP address of the dial peer, in this format: xxx.xxx.xxx.xxx</td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td></td>
</tr>
<tr>
<td>session transport tcp</td>
<td>Configures a VoIP dial peer to use Transmission Control Protocol (TCP).</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session transport tcp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Configuring Network-Based Recording (without Media Profile Recorder)

**SUMMARY STEPS**

1. enable
2. configure terminal
3. media class tag
4. recorder parameter
5. (Optional) media-type audio
6. media-recording dial-peer-tag
7. exit
8. exit
9. dial-peer voice dummy-recorder-dial-peer-tag voip
10. media-class tag
11. destination-pattern [+ string [T]
12. session protocol sipv2
13. session target ipv4:[recording-server-destination-address | recording-server-dns]
14. session transport tcp
15. end
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>media class tag</td>
<td>Configures the media class and enters media class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# media class 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>recorder parameter</td>
<td>Enters media class recorder parameter configuration mode to enable you to configure recorder-specific parameters.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(cg-mediaclass)# recorder parameter</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>media-type audio</td>
<td>Configures recording of audio only in a call with both audio and video.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaprofile)# media-type audio</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>media-recording dial-peer-tag</td>
<td>Configures voice-class recording parameters.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaclass-recorder)# media-recording 8000, 8001, 8002</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>exit</td>
<td>Exits media class recorder parameter configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaclass-recorder)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>exit</td>
<td>Exits media class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaclass)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>dial-peer voice dummy-recorder-dial-peer-tag voip</td>
<td>Configures a recorder dial peer and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 8000 voip</td>
<td></td>
</tr>
</tbody>
</table>
## Media Recording

### Configuring Network-Based Recording (without Media Profile Recorder)

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>media-class <em>tag</em></td>
<td>Configures media class on a dial peer.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# media-class 100</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>destination-pattern [*] <em>string</em> [<em>T]</em></td>
<td>Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The predefined valid entries for <em>string</em> are the digits 0 to 9, the letters A to F and, the following special characters:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The asterisk (*) and pound sign (#) that appear on standard touch-tone dial pads.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Plus sign (+), which indicates that the preceding digit occurred one or more times.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Backslash symbol (), which is followed by a single character, and matches that character.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Media Forking functionality does not work with the wildcard entries other than the predefined set.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# destination-pattern 595959</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>session protocol sipv2</td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>session target ipv4: [recording-server-destination-address</td>
<td>Specifies a network-specific address for a dial peer. Keyword and argument are as follows:</td>
</tr>
<tr>
<td></td>
<td>recording-server-dns]</td>
<td>• <strong>ipv4:</strong> <em>destination address</em> --IP address of the dial peer, in this format: xxx.xxx.xxx.xxx</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session target ipv4:10.42.29.7</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>session transport tcp</td>
<td>Configures a VoIP dial peer to use Transmission Control Protocol (TCP).</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session transport tcp</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>
Verifying the Network-Based Recording Using CUBE

Perform this task to verify the configuration of the Network-Based Recording Using CUBE. The `show` and `debug` commands can be entered in any order.

**SUMMARY STEPS**

1. `enable`
2. `show voip rtp connections`
3. `show voip recmsp session`
4. `show voip recmsp session detail call-id call-id`
5. `show voip rtp forking`
6. `show call active video compact`
7. `show call active video compact`
8. `show sip-ua calls`
9. `show call active video brief`
10. `debug csip messages` (for audio calls)
11. `debug csip messages` (for video calls)
12. `debug csip messages` (for audio-only recording in a call with both audio and video)
13. Enter one of the following:
   - `debug csip all`
   - `debug voip recmsp all`
   - `debug voip ccap all`
   - `debug voip fpi all` (for ASR devices only)

**DETAILED STEPS**

---

**Step 1**

`enable`

Enables privileged EXEC mode.

**Example:**

```
Device> enable
```

**Step 2**

`show voip rtp connections`

Displays Real-Time Transport Protocol (RTP) connections. Two extra connections are displayed for forked legs.

**Example:**

```
Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 8091, Ports Reserved: 101, Ports in Use: 8
Port range not configured, Min: 16384, Max: 32767

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Ports Available</th>
<th>Ports Reserved</th>
<th>Ports In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Address-Range</td>
<td>8091</td>
<td>101</td>
<td>8</td>
</tr>
</tbody>
</table>
```
VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No.</th>
<th>CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>16384</td>
<td>20918</td>
<td>10.104.45.191</td>
<td>10.104.8.94</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>16386</td>
<td>17412</td>
<td>10.104.45.191</td>
<td>10.104.8.98</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>16388</td>
<td>29652</td>
<td>10.104.45.191</td>
<td>10.104.8.98</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>16390</td>
<td>20036</td>
<td>10.104.45.191</td>
<td>10.104.8.94</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>16392</td>
<td>58368</td>
<td>10.104.45.191</td>
<td>10.104.105.232</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>5</td>
<td>16394</td>
<td>53828</td>
<td>10.104.45.191</td>
<td>10.104.105.232</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>5</td>
<td>16396</td>
<td>39318</td>
<td>10.104.45.191</td>
<td>10.104.105.232</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>5</td>
<td>16398</td>
<td>41114</td>
<td>10.104.45.191</td>
<td>10.104.105.232</td>
</tr>
</tbody>
</table>

Found 8 active RTP connections

Step 3  
**show voip recmsp session**

Displays active recording Media Service Provider (MSP) session information internal to CUBE.

**Example:**

```
Device# show voip recmsp session

RECMSP active sessions:
MSP Call-ID AnchorLeg Call-ID ForkedLeg Call-ID
143 141 145

Found 1 active sessions
```

Step 4  
**show voip recmsp session detail call-id call-id**

Displays detailed information about the recording MSP Call ID.

**Example:**

```
Device# show voip recmsp session detail call-id 145

RECMSP active sessions:
Detailed Information
--------------------------
Recording MSP Leg Details:
Call ID: 143
GUID : 7C5946D38ECD

AnchorLeg Details:
Call ID: 141
Forking Stream type: voice-nearend
Participant: 708090

Non-anchor Leg Details:
Call ID: 140
Forking Stream type: voice-farend
Participant: 10000

Forked Leg Details:
Call ID: 145
Near End Stream CallID 145
Stream State ACTIVE
Far End stream CallID 146
```
Stream State ACTIVE
Found 1 active sessions

Device# show voip recmsp session detail call-id 5

RECMSP active sessions:
Detailed Information
-------------------------
Recording MSP Leg Details:
Call ID: 5
GUID : 1E01B6000000

AnchorLeg Details:
Call ID: 1
Forking Stream type: voice-nearend
Forking Stream type: video-nearend
Participant: 1777

Non-anchor Leg Details:
Call ID: 2
Forking Stream type: voice-farend
Forking Stream type: video-farend
Participant: 1888

Forked Leg Details:
Call ID: 6
Voice Near End Stream CallID 6
Stream State ACTIVE
Voice Far End stream CallID 7
Stream State ACTIVE
Video Near End stream CallID 8
Stream State ACTIVE
Video Far End stream CallID 9
Stream State ACTIVE
Found 1 active sessions

<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream State</td>
<td>Displays the state of the call. This can be ACTIVE or HOLD.</td>
</tr>
<tr>
<td>Msp Call-Id</td>
<td>Displays an internal Media service provider call ID and forking related statistics for an active forked call.</td>
</tr>
<tr>
<td>Anchor Leg Call-id</td>
<td>Displays an internal anchor leg ID, which is the dial peer where forking enabled. The output displays the participant number and stream type. Stream type voice-nearend indicates the called party side.</td>
</tr>
<tr>
<td>Non-Anchor Call-id</td>
<td>Displays an internal non-anchor leg ID, which is the dial peer where forking is not enabled. The output displays the participant number and stream type. Stream type voice-nearend indicates the called party side.</td>
</tr>
<tr>
<td>Forked Call-id</td>
<td>This forking leg call-id will show near-end and far-end stream call-id details with state of the stream. Displays an internal foked leg ID. The output displays near-end and far-end details of a stream.</td>
</tr>
</tbody>
</table>

Step 5

**show voip rtp forking**

Displays RTP media-forking connections.

**Example:**
Device# show voip rtp forking
VoIP RTP active forks:
  Fork 1
  stream type voice-only (0): count 0
  stream type voice+dtmf (1): count 0
  stream type dtmf-only (2): count 0
  stream type voice-nearend (3): count 1
    remote ip 10.42.29.7, remote port 38526, local port 18648
    codec g711ulaw, logical ssrc 0x53
    packets sent 29687, packets received 0
  stream type voice+dtmf-nearend (4): count 0
  stream type voice-farend (5): count 1
    remote ip 10.42.29.7, remote port 50482, local port 17780
    codec g711ulaw, logical ssrc 0x55
    packets sent 29686, packets received 0
  stream type voice+dtmf-farend (6): count 0
  stream type video (7): count 0

<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>remote ip 10.42.29.7, remote port 38526, local port 18648</td>
<td>Recording server IP, recording server port, and local CUBE device port where data for stream 1 was first sent from.</td>
</tr>
<tr>
<td>remote ip 10.42.29.7, remote port 50482, local port 17780</td>
<td>Recording server IP, recording server port, and local CUBE device port where data for stream 2 was first sent from.</td>
</tr>
<tr>
<td>packets sent 29686</td>
<td>Number of packets sent to the recorder</td>
</tr>
<tr>
<td>codec g711ulaw</td>
<td>Codec negotiated for the recording leg.</td>
</tr>
</tbody>
</table>

**Step 6**
show call active voice compact

Displays a compact version of voice calls in progress. An additional call leg is displayed for media forking.

**Example:**
```
Device# show call active voice compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 3
  140 ANS T644 g711ulaw VOIP P10000 10.42.30.32:18638
  141 ORG T644 g711ulaw VOIP P708090 10.42.30.189:26184
  145 ORG T643 g711ulaw VOIP P595959 10.42.29.7:38526
```

**Step 7**
show call active video compact

Displays a compact version of video calls in progress.

**Example:**
```
Device# show call active video compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 3
  1 ANS T14 H264 VOIP-VIDEO P1777 10.104.8.94:20036
  2 ORG T14 H264 VOIP-VIDEO P1888 10.104.8.98:29652
  6 ORG T13 H264 VOIP-VIDEO P1234 10.104.105.232:39318
```

**Step 8**
show sip-ua calls

Displays active user agent client (UAC) and user agent server (UAS) information on SIP calls.
Example:

Device# show sip-ua calls
Total SIP call legs:2, User Agent Client:1, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID : C9A3AA00-B49A11E8-8018A74B-CD0B0450@10.0.0.1
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 1234
Called Number : 9876
Called URI : sip:9876@10.0.0.2:9800
Bit Flags : 0xC04018 0x90000100 0x80
CC Call ID : 13
Local UUID : 7d14e2d622ec504f9aa4ba029dd136
Remote UUID : 2522eaa82f505c868037da95438fc49b
Source IP Address (Sig ) : 10.0.0.1
Destn SIP Req Addr:Port : [10.0.0.2]:9800
Destn SIP Resp Addr:Port : [10.0.0.2]:9800
Destination Name : 10.0.0.2
Number of Media Streams : 2
Number of Active Streams : 2
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 13
Stream Type : voice-only (0)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port : [10.0.0.1]:8022
Media Dest IP Addr:Port : [10.0.0.2]:6008
Local Crypto Suite : AES_CM_128_HMAC_SHA1_80 (AEAD_AES_256_GCM
AEAD_AES_128_GCM
AES_CM_128_HMAC_SHA1_80
AES_CM_128_HMAC_SHA1_32 )
Remote Crypto Suite : AES_CM_128_HMAC_SHA1_80
Local Crypto Key : bTqZxZdgFJddAh1a9wJGV3aKxo5vPV+Z1234tVb2
Remote Crypto Key : bTqZxZdgFJddAh1a9wJGV3aKxo5vPV+Z9876tVb2
Media Stream 2
State of the stream : STREAM_ACTIVE
Stream Call ID : 14
Stream Type : video (7)
Stream Media Addr Type : 1
Negotiated Codec : h264 (0 bytes)
Codec Payload Type : 97
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port : [10.0.0.1]:8020
Media Dest IP Addr:Port : [10.0.0.2]:9802
Local Crypto Suite : AES_CM_128_HMAC_SHA1_80
  Aead_AES_256_GCM
  Aead_AES_128_GCM
  AES_CM_128_HMAC_SHA1_80
Remote Crypto Suite : AES_CM_128_HMAC_SHA1_80
Local Crypto Key : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z2345tVb2
Remote Crypto Key : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z8765tVb2
Mid-Call Re-Assocation Count: 0
SRTP-RTP Re-Assocation DSP Query Count: 0

Options-Ping   ENABLED:NO   ACTIVE:NO
Number of SIP User Agent Client(UAC) calls: 1

SIP UAS CALL INFO
Call 1
SIP Call ID : 1-12049@10.0.0.2
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 1234
Called Number : 9876
Called URI : sip:9876@10.0.0.1:5060
Bit Flags : 0x0C0401C 0x10000100 0x4
CC Call ID : 11
Local UUID : 2522ea82f505c868037da95438fc49b
Remote UUID : 7d14e2d622ec504f9aa4ba029dd136
Source IP Address (Sig) : 10.0.0.1
Destn SIP Req Addr:Port : [10.0.0.2]:5060
Destn SIP Resp Addr:Port : [10.0.0.2]:5060
Destination Name : 10.0.0.2
Number of Media Streams : 2
Number of Active Streams: 2
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 11
Stream Type : voice-only (0)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [10.0.0.1]:8016
Media Dest IP Addr:Port: [10.0.0.2]:6009
Local Crypto Suite : AES_CM_128_HMAC_SHA1_80
Remote Crypto Suite : AES_CM_128_HMAC_SHA1_80
Local Crypto Key : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z2345tVb2
Remote Crypto Key : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z8765tVb2
Media Stream 2
State of the stream : STREAM_ACTIVE
Stream Call ID : 12
Stream Type : video (7)
Stream Media Addr Type : 1
Negotiated Codec : h264 (0 bytes)
Codec Payload Type : 97
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
**Step 9**

`show call active video brief`

Displays a truncated version of video calls in progress.

**Example:**

```
Device# show call active video brief

Telephony call-legs: 0
SIP call-legs: 3
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 3

0 : 1 87424920ms.1 (*12:23:53.757 IST Wed Jul 17 2013) +1050 pid:1 Answer 1777 active
dur 00:00:46 tx:5250/1857831 rx:5293/1930598 dscp:0 media:0 audio tos:0xB8 video tos:0x88
IP 10.104.8.94:20036 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off
Transcoded: No
0 : 2 87424930ms.1 (*12:23:53.683 IST Wed Jul 17 2013) +1040 pid:2 Originate 1888 active
dur 00:00:46 tx:5293/1930598 rx:5250/1857831 dscp:0 media:0 audio tos:0xB8 video tos:0x88
IP 10.104.8.98:29652 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off
Transcoded: No
... 0 : 6 87425990ms.1 (*12:23:54.643 IST Wed Jul 17 2013) +680 pid:1234 Originate 1234 active
dur 00:00:46 tx:10398/3732871 rx:0/0 dscp:0 media:0 audio tos:0xB8 video tos:0x0
IP 10.104.105.232:39318 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off
Transcoded: No
...
```

**Step 10**

`debug ccsip messages` (for audio calls)

```
Sent:
INVITE sip:22222@10.42.29.7:5060 SIP/2.0
Via: SIP/2.0/TCP 10.42.30.10:5060;branch=z9hG4bXB622CF
X-Cisco-Recording-Participant: sip:708900@10.42.30.5:media-index="0"
X-Cisco-Recording-Participant: sip:10000@10.42.30.32:media-index="1"
From: <sip:10.42.30.10;tag=5096700-1E1A>
To: <sip:595999@10.42.29.7>
Date: Fri, 18 Mar 2011 07:01:50 GMT
Call-ID: 666C8F613-5064110E-80EAE01B-4C27A620.10.42.30.10
Supported: 100rel,timer,resource-priority,replaces,sdp-anat
Min-SE: 1800
Cisco-Guid: 1334370502-1348997600-2396699092-3395863316
```
User-Agent: Cisco-SIPGateway/IOS-15.2(0.0.2)PIA16
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1300431710
Contact: <sip:10.42.30.10:5060;transport=tcp>
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 449
v=0
o=CiscoSystemsSIP-GW-UserAgent 3021 3526 IN IP4 10.42.30.10
s=SIP Call
c=IN IP4 10.42.30.10
m=audio 24544 RTP/AVP 0 101 19
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=rtpmap:19 CN/8000
a=ptime:20
a=sendonly
m=audio 31166 RTP/AVP 0 101 19
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=rtpmap:19 CN/8000
a=ptime:20
a=sendonly
Received:
SIP/2.0 200 Ok
Via: SIP/2.0/TCP 10.104.46.198:5060;branch=z9hG4bK13262B
To: <sip:23232323010.104.46.201>;tag=ds457251f
From: <sip:10.104.46.198>;tag=110B66-1CBC
Call-ID: 7142FB-9A5011E0-801EF71A-59B4D258@10.104.46.198
CSeq: 101 INVITE
Content-Length: 206
Contact: <sip:23232323010.104.46.201:5060;transport=tcp>
Content-Type: application/sdp
Allow: INVITE, BYE, CANCEL, ACK, NOTIFY, INFO, UPDATE
Server: Cisco-ORA/8.5
v=0
o=CiscoORA 2187 1 IN IP4 10.104.46.201
s=IP Call
c=IN IP4 10.104.46.201
t=0 0
m=audio 54100 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=recvonly
m=audio 39674 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=recvonly
Sent:
ACK sip:23232323010.104.46.201:5060;transport=tcp SIP/2.0
Via: SIP/2.0/TCP 10.104.46.198:5060;branch=z9hG4bK141B87
From: <sip:10.104.46.198>;tag=110B66-1CBC
To: <sip:23232323010.104.46.201>;tag=ds457251f
Date: Mon, 20 Jun 2011 08:42:01 GMT
Call-ID: 7142FB-9A5011E0-801EF71A-59B4D258@10.104.46.198
Max-Forwards: 70
CSeq: 101 ACK
Allow-Events: telephone-event
Content-Length: 0

<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE sip:22222@10.42.29.7:5060 SIP/2.0</td>
<td>22222 is the destination pattern or the number of recording server and is configured under the recorder dial peer.</td>
</tr>
<tr>
<td>X-Cisco-Recording-Participant: sip:708090@10.42.30.5;media-index=&quot;0&quot;</td>
<td>Cisco proprietary header with originating and terminating participant number and IP address used to communicate to the recording server</td>
</tr>
<tr>
<td>Cisco-Guid: 1334370502-1348997600-2396699092-3395863316</td>
<td>GUID is the same for the primary call and forked call.</td>
</tr>
<tr>
<td>m=audio 24544 RTP/AVP 0 101 19</td>
<td>First m-line of participant with payload type and codec information.</td>
</tr>
<tr>
<td>m=audio 31166 RTP/AVP 0 101 19</td>
<td>Second m-line of another participant with codec info and payload type.</td>
</tr>
<tr>
<td>a=sendonly</td>
<td>CUBE is always in send only mode towards Recording server.</td>
</tr>
<tr>
<td>a=recvonly</td>
<td>Recording server is in receive mode only.</td>
</tr>
</tbody>
</table>

Step 11  
**debug ccsip messages** (for video calls)

Sent: INVITE sip:575757@9.45.38.39:7686 SIP/2.0

Via: SIP/2.0/UDP 9.41.36.41:5060;branch=z9hG4bK2CC2408
X-Cisco-Recording-Participant: sip:1777@10.104.45.207;media-index="0 2"
X-Cisco-Recording-Participant: sip:1888@10.104.45.207;media-index="1 3"

Cisco-Guid: 0884935168-0000065536-0000000401-3475859466

v=0

m=audio 17232 RTP/AVP 0 19

a=sendonly

m=audio 17234 RTP/AVP 0 19

a=sendonly
m=video 17236 RTP/AVP 126

a=fmtp:126 profile-level-id=42801E;packetization-mode=1
a=sendonly
m=video 17238 RTP/AVP 126

a=fmtp:126 profile-level-id=42801E;packetization-mode=1
a=sendonly

<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent: INVITE sip:575757@9.45.38.39:7686 SIP/2.0</td>
<td>22222 is the destination pattern or the number of recording server and is configured under the recorder dial peer.</td>
</tr>
<tr>
<td>X-Cisco-Recording-Participant: sip:1777@10.104.45.207;media-index=&quot;0 2&quot;</td>
<td>Cisco proprietary header with originating and terminating participant number and IP address used to communicate to the recording server</td>
</tr>
<tr>
<td>X-Cisco-Recording-Participant: sip:1888@10.104.45.207;media-index=&quot;1 3&quot;</td>
<td></td>
</tr>
<tr>
<td>Cisco-Guid: 0884935168-0000065536-000000401-3475859466</td>
<td>GUID is the same for the primary call and forked call.</td>
</tr>
<tr>
<td>m=audio 17232 RTP/AVP 0 19</td>
<td>First m-line of participant with payload type and audio codec.</td>
</tr>
<tr>
<td>m=audio 17234 RTP/AVP 0 19</td>
<td>Second m-line of another participant with payload type and audio codec.</td>
</tr>
<tr>
<td>m=video 17236 RTP/AVP 126</td>
<td>Third m-line of participant with video payload type and codec info .</td>
</tr>
<tr>
<td>m=video 17238 RTP/AVP 126</td>
<td>Fourth m-line of another participant with video payload type and codec info .</td>
</tr>
<tr>
<td>a=sendonly</td>
<td>CUBE is always in send only mode towards Recording server.</td>
</tr>
</tbody>
</table>

Receive:
SIP/2.0 200 OK

v=0

m=audio 1592 RTP/AVP 0

a=recvonly
m=audio 1594 RTP/AVP 0

a=recvonly
m=video 1596 RTP/AVP 126
   a=fmtp:97 profile-level-id=420015
   a=recvonly
m=video 1598 RTP/AVP 126
   a=fmtp:126 profile-level-id=420015
   a=recvonly
Sent:
ACK sip:9.45.38.39:7686;transport=UDP SIP/2.0
Via: SIP/2.0/UDP 9.41.36.41:5060;branch=z9hG4bK2CD7
From: <sip:9.41.36.41>;tag=1ECFD128-24DF
To: <sip:57575709.45.38.39>;tag=16104SIPpTag011
Date: Tue, 19 Mar 2013 11:40:01 GMT
Call-ID: FFFFFFFF91E00FE6-FFFFFFFF8FC011E2-FFFFFFFF824DF469-FFFFFFFFB6661C06@9.41.36.41
Max-Forwards: 70
CSeq: 101 ACK
Allow-Events: telephone-event

Content-Length: 0

<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m=audio 1592 RTP/AVP 0</td>
<td>First m-line of recording server after it started listening.</td>
</tr>
<tr>
<td>m=audio 1594 RTP/AVP 0</td>
<td>Second m-line of recording server after it started listening.</td>
</tr>
<tr>
<td>m=video 1596 RTP/AVP 126</td>
<td>Third m-line of recording server after it started listening.</td>
</tr>
<tr>
<td>m=video 1598 RTP/AVP 126</td>
<td>Fourth m-line of recording server after it started listening.</td>
</tr>
<tr>
<td>a=recvonly</td>
<td>Recording server in receive only mode.</td>
</tr>
</tbody>
</table>

**Step 12**

**debug cc sip messages** (for audio-only recording in a call with both audio and video)

Displays offer sent to MediaSense having only audio m-lines, when the media-type audio command is configured.

Sent:
INVITE sip:54321@9.45.38.39:36212 SIP/2.0
Via: SIP/2.0/UDP 9.41.36.15:5060;branch=z9hG4bK2216B
X-Cisco-Recording-Participant: sip:4321@9.45.38.39;media-index="0"
X-Cisco-Recording-Participant: sip:1111000010@9.45.38.39;media-index="1"
From: <sip:9.41.36.15>;tag=A2C74-5D9
To: <sip:54321@9.45.38.39>.....
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 337

v=0
o=CiscoSystemsSIP-GW-UserAgent 9849 5909 IN IP4 9.41.36.15
s=SIP Call
c=IN IP4 9.41.36.15
t=0 0
m=audio 16392 RTP/AVP 0 19
c=IN IP4 9.41.36.15
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
a=ptime:20
a=sendonly
m=audio 16394 RTP/AVP 0 19
c=IN IP4 9.41.36.15
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
a=ptime:20
a=sendonly

Response from CUBE has inactive video m-lines.

Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.41.36.15:5060;branch=z9hG4bK2216B
....
v=0
...
m=audio 36600 RTP/AVP 0
c=IN IP4 9.45.38.39
a=rtpmap:0 PCMU/8000
a=ptime:20
a=recvonly
m=audio 36602 RTP/AVP 0
c=IN IP4 9.45.38.39
a=rtpmap:0 PCMU/8000
a=ptime:20
a=recvonly
m=video 0 RTP/AVP 98
c=IN IP4 9.45.38.39
b=TIAS:1500000
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=420015
a=inactive

Step 13

Enter one of the following:

- debug ccsip all
- debug voip recmsp all
- debug voip capi all
- debug voip fpi all (for ASR devices only)

Displays detailed debug messages.

For Audio:

Media forking initialized:

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Recording

Verifying the Network-Based Recording Using CUBE

Media Rendering
MF: Data read from TD container.
MSP callid = 105
Overwriting the GUID with the value got from MSP.
*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_iwf_handle_peer_event:
Overwriting the GUID with the value got from MSP.

For Video:

Media Forking Initialized:

*Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Info/notify/32768/ccsip_trigger_media_forking: MF:
Recv Ack & it's Anchor leg. Start MF.
*Mar 19 16:40:01.784 IST:
State = 1 & posting the event E_IPIP_MEDIA_FORKING_CALLSETUP_IND

Media forking started:

*Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Info/info/36864/ccsip_ipip_media_forking: MF:
Current State = 1, event =31
& Event combination is cracked..
*Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Function/sipSPIGetMainStream:
*Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Function/sipSPIGetMainStream:
*Mar 19 16:40:01.787 IST:
//522/34BF0A000000/SIP/Info/info/34816/ccsip_ipip_media_forking_precondition: MF: Can be started
with current config.
*Mar 19 16:40:01.787 IST: //522/34BF0A000000/SIP/Info/info/33792/ccsip_ipip_media_forking_Precondition:
MF: Populate rec parti header from this leg.
*Mar 19 16:40:01.788 IST: //522/34BF0A000000/SIP/Info/info/33792/ccsip_get_recording_participant_header: MF: X-Cisco header
is PAI..

Adding an audio stream:

*Mar 19 16:40:01.788 IST: //522/34BF0A000000/SIP/Function/sipSPIGetFirstStream:
*Mar 19 16:40:01.788 IST:
//522/34BF0A000000/SIP/Info/verbose/32768/ccsip_ipip_media_forking_BuildMediaRecParticipant: MF:
Populate rec parti header from this leg.
*Mar 19 16:40:01.788 IST:
//522/34BF0A000000/SIP/Info/info/33792/ccsip_get_recording_participant_header: MF: X-Cisco header
is PAI..

Recording participant for anchor leg:

//522/34BF0A000000/SIP/Info/verbose/32768/ccsip_ipip_media_forking_BuildMediaRecParticipant: MF: Populate rec parti header from this leg.
*Mar 19 16:40:01.788 IST: //522/34BF0A000000/SIP/Info/info/33792/ccsip_get_recording_participant_header: MF: X-Cisco header
is PAI..

Cisco Unified Border Element Configuration Guide
Verifying the Network-Based Recording Using CUBE

467
Video forking:  
*Mar 19 16:40:01.789 IST: //522/34BF0A000000/SIP/Function/sipSPIGetVideoStream:  

For Video

## Additional References for Network-Based Recording

### Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediaSense Installation and Administration Guide</td>
<td>Cisco MediaSense Installation and Administration Guide</td>
</tr>
</tbody>
</table>

### Standards and RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 3984</td>
<td><em>RTP Payload Format for H.264 Video</em></td>
</tr>
<tr>
<td>RFC 5104</td>
<td><em>Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)</em></td>
</tr>
<tr>
<td>RFC 5168</td>
<td><em>XML Schema for Media Control</em></td>
</tr>
</tbody>
</table>
SIPREC (SIP Recording)

The SIPREC (SIP Recording) feature supports media recording for Real-time Transport Protocol (RTP) streams in compliance with section 3.1.1. of RFC 7245, with CUBE acting as the Session Recording Client. SIP is used as a protocol between CUBE and the recording server. Recording of a media session is done by sending a copy of a media stream to the recording server. Metadata is the information that is passed by the recording client to the recording server in a SIP session. The recording metadata describes the communication session and its media streams, and also identifies the participants of the call. CUBE acts as the recording client and any third party recorder acts as the recording server.

- Feature Information for SIPREC-based Recording, on page 469
- Prerequisites for SIPREC Recording, on page 470
- Restrictions for SIPREC Recording, on page 470
- Information About SIPREC Recording Using CUBE, on page 471
- How to Configure SIPREC-Based Recording, on page 472
- Configuration Examples for SIPREC-based Recording, on page 477
- Example of Metadata Variations with Different Mid-call Flows, on page 477
- Example of Metadata Variations with Different Transfer Flows, on page 490
- Example of Metadata Variations with Caller-ID UPDATE Flow, on page 491
- Example of Metadata Variations with Call Disconnect, on page 492

Feature Information for SIPREC-based Recording

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPREC (SIP Recording)</td>
<td>Cisco IOS 15.6(1)T</td>
<td>The SIPREC Recording feature supports recording of audio and video calls. Only audio and video media lines are forked. The following commands were modified: recorder parameter and recorder profile.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.17S</td>
<td></td>
</tr>
</tbody>
</table>

Cisco Unified Border Element Configuration Guide 469
Prerequisites for SIPREC Recording

- Recorders must be reachable from CUBE.
- SIPREC should be configured; else, CUBE will fall back to the existing Network-Based Recording implementation. For more information on Network-Based Recording, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-ntwk-based.html.
- CUBE supports the SIP Recording Metadata model format requirements specified in draft-ietf-siprec-metadata-17. Recorders must support metadata format of ver17 at a minimum.
- CUBE should be in compliance with the Session Recording Protocols defined in draft-ietf-siprec-protocol-16. CUBE supports only the “siprec Option” Tag and the “src feature” tag among the various other extensions defined in the protocols draft; CUBE does not support the SDP extensions.

Restrictions for SIPREC Recording

SIPREC-based recording is not supported for the following calls:

- Any media service parameter change via Re-INVITE or UPDATE from recording server is not supported. For example, hold.resume or any codec changes.
- IPv6-to-IPv6 call recording.
- IPv6-to-IPv4 call recording if the recording server is configured on the IPv6 call leg.
- Calls that do not use Session Initiation Protocol (SIP). Must be a SIP-to-SIP call flow.
- Flow-around calls.
- Session Description Protocol (SDP) pass-through calls.
- Real-time Transport Protocol (RTP) loopback calls.
- High-density transcoder calls.
- Secure Real-time Transport Protocol (SRTP) passthrough calls.
- SRTP-RTP calls with forking for SRTP leg (forking is supported for the RTP leg).
- Multicast music on hold (MOH).
- Mid-call renegotiation and supplementary services like Hold/Resume, control pause, and so on are not supported on the recorder call leg.
- Recording is not supported if CUBE is running a TCL IVR application with the exception of survivability.tcl, which is supported with SIPREC based recording.
- Media mixing on forked streams is not supported.
- Digital Signal Processing (DSP) resources are not supported on forked legs.

Restrictions for Video Recording
• If the main call has multiple video streams (m-lines), the video streams other than the first video m-line are not forked
• Application media streams of the primary call are not forked to the recording server
• Forking is not supported if the anchor leg or recording server is on IPv6

Information About SIPREC Recording Using CUBE

Deployment

• Participants—SIP UAs involved in the Communication Session. The UA can be any SIP element
• Communication Session (CS)—Session established between the endpoints
• Session Recording Client (SRC)—CUBE acts as the session recording client that triggers the recording session
• Session Recording Server (SRS)—A SIP User Agent (UA) which is a specialized media server and that acts as a sink for the recorded media and metadata
• Recording Session (RS) — SIP dialog established between CUBE (recording client) and the recording server
• Recording Metadata—Information on the CS and the associated media stream data sent from CUBE to RS

The following figure illustrates a third party recorder deployment with CUBE.

Figure 39: Deployment Scenario for SIPREC Recording Solution

Information flow is described below:
1. Incoming call from SIP trunk
2. Outbound call to Contact Center
3. Media between endpoints flow through CUBE
4. CUBE sets up a new SIP session with the recording device (SRS)

5. CUBE forks RTP media to SRS

In the preceding illustration, the Real Time Protocol (RTP) carries voice data and media streams between the user agents and CUBE. The RTP unidirectional stream represent the communication session forked from CUBE to the recording server to indicate forked media. The Session Initiation protocol (SIP) carries call signaling information along with the metadata information. Media streams from CUBE to recording server are unidirectional because only CUBE sends recorded data to recording server; the recording server does not send any media to CUBE.

Metadata has the following functions:

- Carry the communication session data (audio and video calls) that describes the call to the recording server
- Identifies the participants list
- Identifies the session and media association time

If there are any changes in the call sessions, for example, hold-resume, transfer and so on, these sessions are notified to the recording server through metadata.

**SIPREC High Availability Support**

High availability is supported for SIPREC recording using CUBE. All metadata elements will be checkpointed in a forked call when high-availability is configured. In the event of SSO, all the forked calls and media contexts are preserved on failover.

**How to Configure SIPREC-Based Recording**

**Configuring SIPREC-Based Recording (with Media Profile Recorder)**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. media profile recorder profile-tag
4. (Optional) media-type audio
5. media-recording dial-peer-tag [dial-peer-tag2...dial-peer-tag5]
6. exit
7. media class tag
8. recorder profile tag siprec
9. exit
10. dial-peer voice dummy-recorder-dial-peer-tag voip
11. media-class tag
12. destination-pattern [+] string [T]
13. session protocol sipv2
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable&lt;br&gt;Example: Device&gt; enable</td>
<td>Enables privileged EXEC mode.  • Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal&lt;br&gt;Example: Device# configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>media profile recorder <em>profile-tag</em>&lt;br&gt;Example: Device(config)# media profile recorder 100</td>
<td>Configures the media profile recorder and enters media profile configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>(Optional) media-type audio&lt;br&gt;Example: Device(cfg-mediaprofile)# media-type audio</td>
<td>Configures recording of audio only in a call with both audio and video. If this configuration is not done, both audio and video are recorded.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>media-recording <em>dial-peer-tag</em>&lt;br&gt;Example: Device(cfg-mediaprofile)# media-recording 8000 8001 8002</td>
<td>Configures the dial-peers that need to be configured&lt;br&gt;Note: You can specify a maximum of five dial-peer tags.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>exit&lt;br&gt;Example: Device(cfg-mediaprofile)# exit</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>media class <em>tag</em>&lt;br&gt;Example: Device(config)# media class 100</td>
<td>Configures a media class and enters media class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>recorder profile <em>tag</em> siprec&lt;br&gt;Example: Device(cfg-mediaclass)# recorder profile 201 siprec</td>
<td>Configures the media profile SIPREC recorder.</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>9</td>
<td>exit</td>
<td>Exits media class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(cgf-mediaclass)# exit</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>dial-peer voice dummy-recorder-dial-peer-tag voip</td>
<td>Configures a recorder dial peer and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 8000 voip</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>media-class tag</td>
<td>Configures media class on a dial peer.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# media-class 100</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>destination-pattern [+ string [T]]</td>
<td>Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# destination-pattern 595959</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>session protocol sipv2</td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>session target ipv4:[recording-server-destination-address</td>
<td>Specifies a network-specific address for a dial peer. Keyword and argument are as follows:</td>
</tr>
<tr>
<td></td>
<td>recording-server-dns]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session target ipv4:10.42.29.7</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>session transport tcp</td>
<td>Configures a VoIP dial peer to use Transmission Control Protocol (TCP).</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session transport tcp</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>
Configuring SIPREC-Based Recording (without Media Profile Recorder)

**SUMMARY STEPS**

1. enable
2. configure terminal
3. media class tag
4. recorder parameters siprec
5. (Optional) media-type audio
6. media-recording dial-peer-tag
7. exit
8. exit
9. dial-peer voice dummy-recorder-dial-peer-tag voip
10. media-class tag
11. destination-pattern [+ string [T]]
12. session protocol sipv2
13. session target ipv4: [recording-server-destination-address | recording-server-dns]
14. session transport tcp
15. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> media class tag</td>
<td>Configures the media class and enters media class configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# media class 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> recorder parameters siprec</td>
<td>Enables SIPREC recording.</td>
</tr>
<tr>
<td>Example: Device(cfg-mediaclass)# recorder parameter siprec</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> (Optional) media-type audio</td>
<td>Configures recording of audio only in a call with both audio and video.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><code>media-type audio</code></td>
<td>If this configuration is not done, both audio and video are recorded.</td>
</tr>
</tbody>
</table>

**Step 6**

**media-recording dial-peer-tag**  
**Example:**

```
Device(cfg-mediaclass-recorder)# media-recording 8000, 8001, 8002
```

Configures voice-class recording parameters.  
**Note** You can specify a maximum of five dial-peer tags.

**Step 7**

**exit**  
**Example:**

```
Device(cfg-mediaclass-recorder)# exit
```

Exits media class recorder parameter configuration mode.

**Step 8**

**exit**  
**Example:**

```
Device(cfg-mediaclass)# exit
```

Exits media class configuration mode.

**Step 9**

**dial-peer voice dummy-recorder-dial-peer-tag voip**  
**Example:**

```
Device(config)# dial-peer voice 8000 voip
```

 Configures a recorder dial peer and enters dial peer voice configuration mode.

**Step 10**

**media-class tag**  
**Example:**

```
Device(config-dial-peer)# media-class 100
```

Configures media class on a dial peer.

**Step 11**

**destination-pattern [+] string [T]**  
**Example:**

```
Device(config-dial-peer)# destination-pattern 595959
```

Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer. Keywords and arguments are as follows:

**Step 12**

**session protocol sipv2**  
**Example:**

```
Device(config-dial-peer)# session protocol sipv2
```

Configures the VoIP dial peer to use Session Initiation Protocol (SIP).

**Step 13**

**session target ipv4:[recording-server-destination-address | recording-server-dns]**  
**Example:**

```
Device(config-dial-peer)# session target ipv4:10.42.29.7
```

Specifies a network-specific address for a dial peer. Keyword and argument are as follows:

- **ipv4:** *destination address* --IP address of the dial peer, in this format: **xxx.xxx.xxx.xxx**
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 14</strong> session transport tcp</td>
<td>Configures a VoIP dial peer to use Transmission Control Protocol (TCP).</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# session transport tcp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Configuration Examples for SIPREC-based Recording

**Example: Configuring SIPREC-based Recording with Media Profile Recorder**

Router> enable
Router# configure terminal
Router(config)# media class 101
Router(config)# recorder profile 201 siprec

**Example: Configuring SIPREC-based Recording without Media Profile Recorder**

Router> enable
Router# configure terminal
Router(config)# media class 101
Router(config)# recorder parameter siprec
Router(config)# media-recording 403

### Example of Metadata Variations with Different Mid-call Flows

#### Example: Complete SIP Recording Metadata information sent in INVITE or Re-INVITE

The following example provides all the elements involved in Recording Metadata XML body.

```xml
--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required
v=0
o=CiscoSystemsSIP-GW-UserAgent 509 7422 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
t=0 0
m=audio 16552 RTP/AVP 8 101
c=IN IP4 9.42.25.149
a=rtpmap:8 PCMA/8000
```
Example: Complete SIP Recording Metadata information sent in INVITE or Re-INVITE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
  <datamode>complete</datamode>
  <sipSessionID>276ac102a3c05270a4375d99512e1a1;remote=110b0c0f50775078b13d60be0044db11</sipSessionID>
  <start-time>2015-05-19T09:42:06.911Z</start-time>
  <participant participant_id="JaPQeP1CEeSA76sYHx7YVg==">
    <nameID aor="sip:808808@9.0.0.174">
      <name xml:lang="en">808808</name>
    </nameID>
  </participant>
  <participantsessionassoc participant_id="JaPQeP1CEeSA76sYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
    <associate-time>2015-05-19T09:42:06.911Z</associate-time>
  </participantsessionassoc>
  <stream stream_id="JaPQeP1CEeSA66sYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
    <label>1</label>
  </stream>
  <stream stream_id="JaPQeP1CEeSA8asYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
    <label>2</label>
  </stream>
  <stream stream_id="JaPQeP1CEeSA8qsYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
    <label>3</label>
  </stream>
  <stream stream_id="JaPQeP1CEeSA8qsYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
    <label>4</label>
  </stream>
</recording>
```
<label>2</label>
</stream>
<stream stream_id="JaPQeP1CEeSA9ksYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
  <label>4</label>
</stream>

<participantstreamassoc participant_id="JaPQeP1CEeSA76sYHx7YVg==">
  <send>JaPQeP1CEeSA8ksYHx7YVg==</send>
  <recv>JaPQeP1CEeSA86sYHx7YVg==</recv>
  <send>JaPQeP1CEeSA8asYHx7YVg==</send>
  <recv>JaPQeP1CEeSA9ksYHx7YVg==</recv>
</participantstreamassoc>

<participantstreamassoc participant_id="JaPQeP1CEeSA8qsYHx7YVg==">
  <send>JaPQeP1CEeSA86sYHx7YVg==</send>
  <recv>JaPQeP1CEeSA8ksYHx7YVg==</recv>
  <send>JaPQeP1CEeSA9ksYHx7YVg==</send>
  <recv>JaPQeP1CEeSA8asYHx7YVg==</recv>
</participantstreamassoc>

</recording>
—uniqueBoundary—

### Output Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datamode&gt;complete&lt;/datamode</td>
<td>&lt;dataMode&gt; is a recording element that indicates whether the XML document is a complete document or a partial update. If no &lt;dataMode&gt; element is present then the default value is &quot;complete&quot;.</td>
</tr>
<tr>
<td>session session_id=&quot;JaPQeP1CEeSA66sYHx7YVg==&quot;</td>
<td>Session ID which remains constant for the complete call leg.</td>
</tr>
<tr>
<td>sipSessionID 276ac102a3c05270a4375d99512e1a1; remote=110b0c0f50775078b13d60be0044db11</td>
<td>This attribute carries a SIP Session-ID of the original call between the participants.</td>
</tr>
<tr>
<td>participant participant_id=&quot;JaPQeP1CEeSA76sYHx7YVg==&quot;&gt;</td>
<td>Name and participant ID of the first participant. The first participant will always be the anchor leg of the call. Each participant has a unique 'participant_id' attribute. For example, nameID is sip:808808.</td>
</tr>
<tr>
<td>a=label:1;</td>
<td>The &lt;stream&gt; element represents a Media Stream object. Stream element indicates the SDP media lines associated with the session and participants.</td>
</tr>
<tr>
<td>&lt;stream stream_id=&quot;JaPQeP1CEeSA86sYHx7YVg==&quot; session_id=&quot;JaPQeP1CEeSA66sYHx7YVg==&quot;&gt;</td>
<td>The &lt;label&gt; element within the &lt;stream&gt; element references an SDP &quot;a=label&quot; attribute that identifies an m-line within the RS SDP. This m-line carries the media stream from the SRC to the SRS.</td>
</tr>
<tr>
<td>&lt;label&gt;1&lt;/label&gt; &lt;/stream&gt;</td>
<td></td>
</tr>
</tbody>
</table>
ParticipantCSAssociation class describes the association of the first participant to a CS for a period of time. A participant can associate and dissociate from a CS several times.

ParticipantCS Association class has the following attributes:

- Associate-time—Time when the participant is associated to CS.
- Disassociate-time—Time when the participant is disassociated from a CS.

Each CS object is represented by one session element. Each session element has a unique 'session_id' attribute which helps to identify unique CS sessions.

Example: Hold with Send-only / Recv-only Attribute in SDP

When a participant puts the audio call on hold with send-only attribute, the stream is sent only in one direction.

Example: Hold with Send-only / Recv-only Attribute in SDP

Here, in a normal recording session, both participants sent audio and video streams.

```
--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required
v=0
c=IN IP4 9.42.25.149
```

Example: Hold with Send-only / Recv-only Attribute in SDP

```
o=CiscoSystemsSIP-GW-UserAgent 2973 4879 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
```
Example: Hold with Send-only / Recv-only Attribute in SDP

```plaintext
Content-Type: application/rs-metadata+xml
Content-Disposition: recording-session
<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
    ...
    <stream stream_id="jIBTUf1BReSaKsYHx7YVg==" session_id="jH+2kFlBEEsAb6sYHx7YVg==">
        <label>1</label>
    </stream>
    ...
    <stream stream_id="jIBTUf1BReSaKsYHx7YVg==" session_id="jH+2kFlBEEsAb6sYHx7YVg==">
        <label>3</label>
    </stream>
    ...
    <stream stream_id="jIBTUf1BReSaKsYHx7YVg==" session_id="jH+2kFlBEEsAb6sYHx7YVg==">
        <label>2</label>
    </stream>
    ...
    <stream stream_id="jIBTUf1BReSaKsYHx7YVg==" session_id="jH+2kFlBEEsAb6sYHx7YVg==">
        <label>4</label>
    </stream>
</recording>
```
In this scenario, the second participant puts the call on hold using sendonly and the first participant will respond using recvonly. You can see from the participantStream association element that the second participant only sends audio and video streams and the first participant just receives the media streams.

The output after the second participant puts the call on hold with sendonly attribute:

```
Content-Type: application/sdp
v=0
o=CiscoSystemsSIP-GW-UserAgent 2973 4880 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
t=0 0
m=audio 16464 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=inactive
a=label:1
m=audio 16466 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=sendonly
a=label:2
m=video 16468 RTP/AVP 97
c=IN IP4 9.42.25.149
b=TIAS:100000
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42801E;packetization-mode=0
a=inactive
a=label:3
m=video 16470 RTP/AVP 97
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42801E;packetization-mode=0
a=sendonly
a=label:4
```
Example: Hold with Inactive Attribute in SDP

Here, you can see that video call is sent in the initial INVITE to recorder where both the participants send and receive audio and video streams. There are 2 audio and 2 video streams from both the participants each in the participantStream association element.

```sdp
v=0
o=CiscoSystemsSIP-GW-UserAgent 7476 1347 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
t=0 0
m=audio 16496 RTP/AVP 0 101
  c=IN IP4 9.42.25.149
  a=rtpmap:0 PCMU/8000
  a=rtpmap:101 telephone-event/8000
  a=fmtp:101 0-16
  a=ptime:20
  a=sendonly
  a=label:1
m=audio 16498 RTP/AVP 0 101
  c=IN IP4 9.42.25.149
  a=rtpmap:0 PCMU/8000
  a=rtpmap:101 telephone-event/8000
  a=fmtp:101 0-16
  a=ptime:20
  a=sendonly
  a=label:2
m=video 16500 RTP/AVP 97
  c=IN IP4 9.42.25.149
  b=TIAS:100000
  a=rtpmap:97 H264/90000
  a=fmtp:97 profile-level-id=42801E;packetization-mode=0
  a=sendonly
  a=label:3
m=video 16502 RTP/AVP 97
  c=IN IP4 9.42.25.149
  b=TIAS:100000
  a=rtpmap:97 H264/90000
  a=fmtp:97 profile-level-id=42801E;packetization-mode=0
  a=sendonly
  a=label:4
```

Cisco Unified Border Element Configuration Guide

483
When the first participant puts the call on hold with inactive SDP attribute, there will be not any active streams in the metadata.
Example: Escalation

During escalation, video streams will be added to the Re-INVITE meta-data sent to the recorder.

In the below example, you can see the metadata representation of an original audio call sent in the initial INVITE to the recorder where both the participants send and receive audio streams.
After escalation, video streams get added into the **participantStream association** element in metadata for both the participants. There will be 4 streams in total.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
    ...
    <stream stream_id="evyS5/1CEsBOKsYHx7YVg==" session_id="evv2v/1CEsB6sYHx7YVg==">
        <label>1</label>
    </stream>
    ...
    <stream stream_id="evyS5/1CEsBOqsYHx7YVg==" session_id="evv2v/1CEsB6sYHx7YVg==">
        <label>2</label>
    </stream>
    <participantstreamassoc participant_id="evyS5/1CEsB0sYHx7YVg==">
        <send>evyS5/1CEsBOKsYHx7YVg==</send>
        <recv>evyS5/1CEsBOqsYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="evyS5/1CEsB0sYHx7YVg==">
        <send>evyS5/1CEsBOqsYHx7YVg==</send>
        <recv>evyS5/1CEsBOKsYHx7YVg==</recv>
    </participantstreamassoc>
</recording>
```
Example: De-escalation

During de-escalation, video streams will be truncated in the Re-INVITE metadata sent to the recorder.

In the below example, you can see two streams each for the audio and video calls in the metadata.
After de-escalation, video streams are removed from the metadata and only audio calls will be present in the participantStream association element.

---uniqueBoundary---

**Content-Type: application/sdp**

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 7616 8309 IN IP4 9.42.25.149
c=IN IP4 9.42.25.149
t=0 0
m=audio 16648 RTP/AVP 0 101
c=IN IP4 9.42.25.149
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=sendonly
a=label:1
m=audio 16650 RTP/AVP 0 101
c=IN IP4 9.42.25.149
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=sendonly
a=label:2
m=video 0 RTP/AVP 97
c=IN IP4 9.42.25.149
b=TIAS:1000000
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42801E;packetization-mode=0
a=sendonly
a=label:3
m=video 0 RTP/AVP 97
c=IN IP4 9.42.25.149
b=TIAS:1000000
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42801E;packetization-mode=0
a=sendonly
a=label:4
```
Example of Metadata Variations with Different Transfer Flows

Example: Transfer of Re-INVITE/REFER Consume Scenario

In the case of Re-INVITE or REFER Consume transfer scenarios, CUBE receives re-INVITE with caller-id change. This re-INVITE will have the remote-party-ID details.

After transfer, participant A is disassociated from the call and participant C joins the call. This information is provided in the metadata sent to the recording server. Here, 7774442214 associates and 7774442212 disassociates from the call.

INVITE sip:7774442216@10.64.86.102:5060;transport=tcp SIP/2.0
From: <sip:7774442212@10.104.54.52>;tag=498652~97a89a01
To: <sip:7774442216@10.64.86.102>;tag=7C798-1441
...
Remote-Party-ID: <sip:7774442214@10.104.54.52>;party=calling;screen=yes;privacy=off
Contact: <sip:7774442214@10.104.54.52:5060;transport=tcp>
...
<participant participant_id="vm+z2xM6EeWAIN4iOrLrag==">
  <nameID aor="sip:7774442214@10.104.54.52"">
  </nameID>
</participant>
<participantsessionassoc participant_id="vm+z2xM6EeWAIN4iOrLrag==" session_id="vACJ+xM6EeWAGN4iOrLrag==">
  <associate-time>2015-06-16T08:44:32.869Z</associate-time>
</participantsessionassoc>
...
<participant participant_id="vACJ+xM6EeWAGN4iOrLrag==">
  <nameID aor="sip:7774442212@10.104.54.52"">
  </nameID>
</participant>
<participantsessionassoc participant_id="vACJ+xM6EeWAGN4iOrLrag==" session_id="vACJ+xM6EeWAGN4iOrLrag==">
  <disassociate-time>2015-06-16T08:44:32.869Z</disassociate-time>
</participantsessionassoc>
...
Example of Metadata Variations with Caller-ID UPDATE Flow

Example: Caller-ID UPDATE Request and Response Scenario

In case of Re-INVITE based transfer, any UPDATE request will contain caller-id changes. These changes are forwarded to the remote party and once CUBE receives a 200OK message, the remote-party-ID details are transferred.

The response of UPDATE request contains the associated caller-id changes. The CUBE forwards the response UPDATE information to the remote party with caller-id changes after the UPDATE request. From the metadata, you can see that the participants A and C disassociate from the call and participants B and D joins (associates) the call. Here, 7774442212 and 7774442216 disassociates from the call and 7774442214 and 7774442218 joins the call after the caller-id update.

```
UPDATE sip:7774442216@10.64.86.102:5060;transport=tcp SIP/2.0
From: <sip:7774442212@10.104.54.52>;tag=498652~97a89a01
To: <sip:7774442216@10.64.86.102>;tag=7C798-1441
...
Remote-Party-ID: <sip:7774442214@10.104.54.52>;party=calling;screen=yes;privacy=off
Contact: <sip:7774442214@10.104.54.52:5060;transport=tcp>
Response of UPDATE contains caller-id changes
...
SIP/2.0 200 OK
From: <sip:7774442212@10.64.86.102>;tag=7C78C-1E7C
To: <sip:7774442216@10.104.54.52>;tag=498653~97a89a01
...
Remote-Party-ID: <sip:7774442218@10.104.54.52>;party=called;screen=yes;privacy=off
Contact: <sip:7774442218@10.104.54.52:5060>
Content-Length: 0
...
```

```
<participant participant_id="vm+z2xM6eWAIN4iOrLrag==">
  <nameID aor="sip:7774442214@10.104.54.52">
  </nameID>
</participant>
<participantsessionassoc participant_id="vm+z2xM6eWAIN4iOrLrag==" session_id="vACJ+xM6eWAF94iOrLrag==">
  <associate-time>2015-06-16T08:44:32.869Z</associate-time>
</participantsessionassoc>
<participant participant_id="vm+z2xM6eWAIN4iOrLrag==">
  <nameID aor="sip:7774442218@10.104.54.52">
  </nameID>
</participant>
<participantsessionassoc participant_id="vm+z2xM6eWAIN4iOrLrag==" session_id="vACJ+xM6eWAF94iOrLrag==">
  <associate-time>2015-06-16T08:44:32.869Z</associate-time>
</participantsessionassoc>
<participant participant_id="vACJ+xM6eWAGN4iOrLrag==">
  <nameID aor="sip:7774442212@10.104.54.52">
  </nameID>
</participant>
<participantsessionassoc participant_id="vACJ+xM6eWAGN4iOrLrag==" session_id="vACJ+xM6eWAGN4iOrLrag==">
  <disassociate-time>2015-06-16T08:44:32.869Z</disassociate-time>
</participantsessionassoc>
```
Example of Metadata Variations with Call Disconnect

Example: Disconnect while Sending Metadata with BYE

When the original call disconnects without any reason, CUBE initiates a BYE session with the recording server along with the metadata.

In this case, the metadata contains the end time of the session along with the disassociation time of all the active participants from the call.

BYE sip:5555555@8.41.17.71:13961;transport=UDP SIP/2.0
...
Reason: Q.850;cause=16
Content-Type: application/rs-metadata+xml
Content-Disposition: recording-session
Content-Length: 984

<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
  <datamode>complete</datamode>
  <session session_id="t5nW8RM6EeWACt4iOrLrag==">
    <end-time>2015-06-16T08:44:36.661Z</end-time>
  </session>
  <participant participant_id="t5nW8RM6EeWACt4iOrLrag==">
    <nameID aor="sip:7774442212@10.104.54.52"></nameID>
  </participant>
  <participantsessionassoc participant_id="t5nW8RM6EeWACt4iOrLrag==" session_id="t5nW8RM6EeWACt4iOrLrag==">
    <disassociate-time>2015-06-16T08:44:36.657Z</disassociate-time>
  </participantsessionassoc>
  <participant participant_id="t5nW8RM6EeWACd4iOrLrag==">
    <nameID aor="sip:7774442214@10.104.54.52"></nameID>
  </participant>
  <participantsessionassoc participant_id="t5nW8RM6EeWACd4iOrLrag==" session_id="t5nW8RM6EeWACd4iOrLrag==">
    <disassociate-time>2015-06-16T08:44:36.657Z</disassociate-time>
  </participantsessionassoc>
</recording>
Chapter 37

Video Recording - Additional Configurations

This module describes the following additional configurations that can be done for Video Recording:

- Request a Full-Intra Frame using RTCP or SIP INFO methods.
- Configure an H.264 Packetization mode.
- Monitor Intra-Frames and Reference Frames

- Feature Information for Video Recording - Additional Configurations, on page 493
- Information About Additional Configurations for Video Recording, on page 494
- How to Configure Additional Configurations for Video Recording, on page 494
- Verifying Additional Configurations for Video Recording, on page 497

Feature Information for Video Recording - Additional Configurations

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 55: Feature Information for Network-Based Recording of Video Calls Using Cisco Unified Border Element

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network-Based Recording of Video Calls Using Cisco Unified Border Element</td>
<td>15.3(3)M Cisco IOS XE Release 3.10S</td>
<td>The Network-Based Recording of Video Calls Using Cisco Unified Border Element feature supports software-based forking and recording of video calls. The following commands were introduced or modified: media profile video, ref-frame-req rtp, ref-frame-req sip-info, video profile, h264-packetization-mode, monitor-ref-frames.</td>
</tr>
</tbody>
</table>
Information About Additional Configurations for Video Recording

Full Intra-Frame Request

Full Intra-Frame Request is a request sent for an I-frame. An I-frame is an entire key or reference frame that is compressed without considering preceding or succeeding video frames. Succeeding video frames are differences to the original I-frame (what has moved) instead of entire video frame information.

The call between Cisco Unified Border Element and the Cisco MediaSense server is established after the call between the endpoints is established. As a result, the Real-Time Transport Protocol (RTP) channel between the endpoints gets established first and the RTP channel with the recording server gets established later. The impact of this delay is more on video recording because the initial I-frame from the endpoint may not get forked, and frames that follow cannot get decoded. To mitigate the impact of the lost RTP video packets, Cisco Unified Border Element generates Full Intra-Frame Request (FIR) using either Real-Time Transport Control Protocol (RTCP) or SIP INFO, or both, requesting the endpoint to send a fully encoded video frame in the subsequent RTP packet.

The following types of FIR are supported on network-based recording of video calls using Cisco Unified Border Element:

- RTCP FIR (based on RFC 5104).
- SIP INFO FIR (based on RFC 5168).
- Both RTCP FIR and SIP INFO FIR (Cisco Unified Border Element can be configured to send both RTCP FIR and SIP INFO requests at the same time).

How to Configure Additional Configurations for Video Recording

Enabling FIR for Video Calls (Using RTCP of SIP INFO)

Perform this task to enable Full Intra-Frame Request (FIR) during the network-based recording of a video call using Real-Time Transport Control Protocol (RTCP) or using the Session Initiation Protocol (SIP) INFO method.

SUMMARY STEPS

1. enable
2. configure terminal
3. media profile video media-profile-tag
4. Do one of the following:
   - ref-frame-req rtcp retransmit-count retransmit-number
   - ref-frame-req sip-info
5. end
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3 media profile video media-profile-tag</td>
<td>Configures a video media profile and enters media profile configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# media profile video 1</td>
<td></td>
</tr>
<tr>
<td>Step 4 Do one of the following:</td>
<td>Enables FIR using the RTCP or SIP INFO method.</td>
</tr>
<tr>
<td>• ref-frame-req rtcp retransmit-count retransmit-number</td>
<td></td>
</tr>
<tr>
<td>• ref-frame-req sip-info</td>
<td></td>
</tr>
<tr>
<td>Example: Device(cfg-mediaprofile)# ref-frame-req rtcp retransmit-count 4</td>
<td></td>
</tr>
<tr>
<td>Example: Device(cfg-mediaprofile)# ref-frame-req sip-info</td>
<td></td>
</tr>
<tr>
<td>Step 5 end</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td>Example: Device(cfg-mediaprofile)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Configuring H.264 Packetization Mode

When a device configured as CUBE is offered more than one H.264 packetization mode on an inbound video call leg, the device offers all received modes to the outbound call leg, allowing dynamic change of mode during a call. However when a call is forked, the MediaSense recording server is not able to support this dynamic change of the packetization mode.

This feature restricts the device and allows it to offer only the configured packetization mode to the outbound call leg when media forking is configured.

### SUMMARY STEPS

1. enable
2. configure terminal
3. media profile video media-profile-tag
4. h264-packetization-mode packetization mode
5. end
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> media profile video media-profile-tag</td>
<td>Configures a video media profile and enters media profile configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# media profile video 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> h264-packetization-mode packetization mode</td>
<td>Configures the H.264 packetization mode offered by a device on the outbound call leg of a forked call when multiple H.264 packetization modes are present in the offer received by the device on the inbound call leg.</td>
</tr>
<tr>
<td>Example: Device(cfg-mediaprofile)# h264-packetization-mode 2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td>Example: Device(cfg-mediaprofile)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Monitoring Reference files or Intra Frames

Perform this task to configure device to perform deep packet inspection (DPI) of RTP packets received from an endpoint and keep track of how many instantaneous decoder refresh (IDR) frames have been received and the timestamp of the IDRs.

### SUMMARY STEPS

1. enable
2. configure terminal
3. media profile video media-profile-tag
4. monitor-ref-frames
5. end

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> media profile video media-profile-tag</td>
<td>Configures a video media profile and enters media profile configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# media profile video 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> monitor-ref-frames</td>
<td>Monitors reference frames or intra-frames.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(cfg-mediaprofile)# monitor-ref-frames</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(cfg-mediaprofile)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Verifying Additional Configurations for Video Recording

Perform this task to verify the additional configurations of the video recording. The `show` commands can be entered in any order.

#### SUMMARY STEPS

1. `enable`  
   Enables privileged EXEC mode.

   **Example:**  
   Device> enable

2. `show call active video called-number number | include VideoRtcpIntraFrameRequestCount`  
   3. `show call active video called-number number | include VideoSipInfoIntraFrameRequestCount`  
   4. `show call active video | include VideoTimeOfLastReferenceFrame`  
   5. `show call active video | include VideoReferenceFrameCount`

#### DETAILED STEPS

**Step 1**  

`enable`  

Enables privileged EXEC mode.

**Example:**  
Device> enable

**Step 2**  

`show call active video called-number number | include VideoRtcpIntraFrameRequestCount`  

Displays the number of RTCP FIR requests sent on each leg.

**Example:**  
Device# show call active video called-number 990057 | include VideoRtcpIntraFrameRequestCount  

! Main call legs
Verifying Additional Configurations for Video Recording

VideoRtcpIntraFrameRequestCount=1
VideoRtcpIntraFrameRequestCount=1

!CUBE does not generate FIR request on forked leg
VideoRtcpIntraFrameRequestCount=0

**Step 3**  
```bash
show call active video called-number number | include VideoSipInfoIntraFrameRequestCount
```

Displays the number of SIP INFO FIR requests sent on each leg.

**Example:**
```
Device# show call active video called-number 990062 | include VideoSipInfoIntraFrameRequestCount
```

! Main call legs
VideoSipInfoIntraFrameRequestCount=1
VideoSipInfoIntraFrameRequestCount=1

!CUBE does not generate FIR request on forked leg
VideoSipInfoIntraFrameRequestCount=0

**Step 4**  
```bash
show call active video | include VideoTimeOfLastReferenceFrame
```

Displays the timestamp of latest IDR frame.

**Step 5**  
```bash
show call active video | include VideoReferenceFrameCount
```

Displays the number of IDR frames received on that call leg.
CHAPTER 38

Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

The Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording feature provides support for the transmission of globally unique identifiers (GUIDs) received from a third-party private branch exchange (PBX) to the recording server using an established Session Initiation Protocol (SIP) session, making CUBE recording more interoperable with third-party vendors.

- Feature Information for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 499
- Restrictions for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 500
- Information About Third-Party GUID Capture for Correlation Between Calls and SIP-based recording, on page 500
- How to Capture Third-Party GUID for Correlation Between Calls and SIP-based Recording, on page 500
- Verifying Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 503
- Configuration Examples for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 504

Feature Information for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
### Restrictions for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

- The third-party GUID must be received through an INVITE message or a 200 OK message (depending on whether the third-party PBX is initiating the call [caller] or receiving the call [callee]). No other request type, including re-invites, is supported.
- The third-party GUID can be received only through the primary inbound call leg or the primary outbound call leg.

### Information About Third-Party GUID Capture for Correlation Between Calls and SIP-based recording

Enterprise call control systems such as the Cisco Unified Communications Manager (CUCM) use globally unique identifiers (GUIDs) to correlate the multiple call legs of a single call. The call can then be forwarded or transferred, creating additional call legs associated with the same GUID. When recording is configured, CUBE initiates a SIP session with a recorder server and forks the media packets it receives or transmits, along with participant information like called number, calling number, Remote Party ID (RPID), and P-Asserted-Identity (PAI).

While the Cisco-Guid header (used by CUCM) is transmitted to the recording server, third-party GUIDs are not. Third-party GUIDs can be received through an INVITE message or a 200 OK message, depending on whether the third-party PBX is initiating the call [caller] or receiving the call [callee].

Forwarding the GUID to the recording server enables correlation between call records of the PBX and the recording server.

### How to Capture Third-Party GUID for Correlation Between Calls and SIP-based Recording

To capture the third-party GUID and forward it to the recording server, you need to copy a third-party GUID header that CUBE receives, configure a SIP copylist for that header, and apply it to the primary inbound and...
outbound call leg dial peers. A SIP profile is configured to copy this incoming header to a user-defined variable and apply it to an outgoing header on the recording leg dial peer.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice class sip-copylist tag
4. sip-header ThirdParty-GUID-headername
5. exit
6. dial-peer voice inbound-dialpeer-tag voip
7. voice class sip-copylist tag
8. exit
9. dial-peer voice outbound-dialpeer-tag voip
10. voice class sip-copylist tag
11. exit
12. voice class sip-profiles profile-id
13. request INVITE peer-header sip GUID-header-to-copy copy header-value-to-match copy-variable
14. request INVITE sip-header header-to-add add header-value-to-add
15. request INVITE sip-header GUID-header-to-modify modify header-value-to-match header-value-to-replace
16. exit
17. dial-peer voice recorder-dial-peer-tag voip
18. voice-class sip profiles profile-tag
19. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>voice class sip-copylist tag</td>
<td>Configures a list of entities to be sent to a peer call leg and enters voice class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# voice class sip-copylist 100</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>sip-header ThirdParty-GUID-headername</td>
<td>Specifies that the third-party GUID header must be copied from the inbound dial-peer leg to the outbound dial-peer call leg.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# sip-header Third-Party-GUID</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
</tbody>
</table>
| **Step 5** | **exit**  
**Example:**  
Device(config-class)# exit | Exits voice class configuration mode. |
| **Step 6** | **dial-peer voice inbound-dialpeer-tag voip**  
**Example:**  
Device(config)# dial-peer voice 2 voip | Enters inbound dial-peer configuration mode. |
| **Step 7** | **voice class sip-copylist tag**  
**Example:**  
Device(config-dial-peer)# voice class sip-copylist 100 | Applies the copy list to the dial peer. |
| **Step 8** | **exit**  
**Example:**  
Device(config-dial-peer)# exit | Exits to global configuration mode. |
| **Step 9** | **dial-peer voice outbound-dialpeer-tag voip**  
**Example:**  
Device(config)# dial-peer voice 3 voip | Enters outbound dial-peer configuration mode. |
| **Step 10** | **voice class sip-copylist tag**  
**Example:**  
Device(config-dial-peer)# voice class sip-copylist 100 | Applies the copy list to the dial peer. |
| **Step 11** | **exit**  
**Example:**  
Device(config-dial-peer)# exit | Exits to global configuration mode. |
| **Step 12** | **voice class sip-profiles profile-id**  
**Example:**  
Device(config)# voice class sip-profiles 10 | Creates a SIP profile and enters voice class configuration mode. |
| **Step 13** | **request INVITE peer-header sip GUID-header-to-copy copy header-value-to-match copy-variable**  
**Example:**  
Device(config-class)# request INVITE peer-header sip Third-Party-GUID copy "(.*)" u01 | Copies headers from the INVITE message of the incoming dial peer into a copy variable. |
<p>| <strong>Step 14</strong> | <strong>request INVITE sip-header header-to-add add header-value-to-add</strong> | Adds a SIP header to a SIP request. |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# request INVITE sip-header Unsupported add &quot;Unsupported: Dummy Header&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td></td>
</tr>
<tr>
<td>request INVITE sip-header <strong>GUID-header-to-modify</strong> modify <strong>header-value-to-match</strong> <strong>header-value-to-replace</strong></td>
<td>Modifies the outgoing header using the copy variable defined in the previous step.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# request INVITE sip-header Unsupported modify &quot;.*&quot; &quot;Third-Party-GUID: \u01&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>Exits to global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 17</strong></td>
<td></td>
</tr>
<tr>
<td>dial-peer voice <strong>recorder-dial-peer-tag</strong> voip</td>
<td>Enters the dial peer configuration mode for the specified outbound recorder dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 2 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 18</strong></td>
<td></td>
</tr>
<tr>
<td>voice-class sip profiles <strong>profile-tag</strong></td>
<td>Applies the SIP profile to the recording dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip profiles 30</td>
<td></td>
</tr>
<tr>
<td><strong>Step 19</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Verifying Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

**SUMMARY STEPS**

1. **debug ccsip messages** for an INVITE message
2. **debug ccsip messages** for a 200 OK message

**DETAILED STEPS**

**Step 1**  **debug ccsip messages** for an INVITE message

Displays all Session Initiation Protocol (SIP) Service Provider Interface (SPI) messages for an INVITE message.

**Example:**
Configuration Examples for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

Step 2  debug ccsip messages for a 200 OK message

Displays all Session Initiation Protocol (SIP) Service Provider Interface (SPI) messages for a 200 OK message.

Example:

```
Step 2 debug ccsip messages for a 200 OK message

Displays all Session Initiation Protocol (SIP) Service Provider Interface (SPI) messages for a 200 OK message.

Example:

```

Configuration Examples for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

! Create a copylist
Device(config)# voice class sip-copylist 100
! GUID for third party PBX
Device(config-class)# sip-header Third-Party-GUID
! GUID for CUCM
Device(config-class)# sip-header Cisco-Guid
Device(config-class)# exit

! Apply copylist to inbound dial peer so that headers specified in copylist are copied
Device(config)# dialpeer voice 2 voip
Device(config-dial-peer)# voice class sip-copylist 100
Device(config-dial-peer)# exit

! SIP profile copies incoming third-party GUID to a variable from a peer header. This variable
! is then used modify outgoing headers
Device(config)# voice class sip-profiles 10
Device(config-class)# request INVITE peer-header sip Third-Party-GUID copy "(.*)" u01
Device(config-class)# request INVITE sip-header Unsupported add "Unsupported: Dummy Header"
Device(config-class)# request INVITE sip-header Unsupported modify ".*" "Third-Party-GUID: \u01"

```
Device(config-class)# exit

! Apply SIP profile to outbound dial peer
Device(config)# dial-peer voice 2 voip
Device(config-dial-peer)# voice-class sip profiles 30
CHAPTER 39

Cisco Unified Communications Gateway Services--Extended Media Forking

The Cisco Unified Communications (UC) Services API provides a unified web service interface for the different services in IOS gateway thereby facilitating rapid service development at application servers and managed application service providers.

This chapter explains the Extended Media Forking (XMF) provider that allows applications to monitor calls and trigger media forking on Real-time Transport Protocol (RTP) and Secure RTP calls.

- Feature Information for Cisco Unified Communications Gateway Services—Extended Media Forking, on page 507
- Restrictions for Unified Communications Gateway Services—Extended Media Forking, on page 508
- Information About Cisco Unified Communications Gateway Services, on page 508
- How to Configure UC Gateway Services, on page 515
- Configuration Examples for UC Gateway Services, on page 522

Feature Information for Cisco Unified Communications Gateway Services—Extended Media Forking

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco Unified Communications Gateway Services</td>
<td>Cisco IOS 15.3(3)M, Cisco IOS XE 3.10S</td>
<td>The Cisco Unified Communications (UC) Services API provides a unified web service interface for the different services in IOS gateway thereby facilitating rapid service development at application servers and managed application service providers.</td>
</tr>
</tbody>
</table>
## Restrictions for Unified Communications Gateway Services—Extended Media Forking

- Media renegotiation is not supported.
- Media mixing on forked media streams is not supported.
- recordTone insertion is not supported with SRTP calls.
- mediaForkingReason tag is only to notify midcall stream events; notification for events such as codec change is not supported.
- Only voice media stream is supported.
- Supplementary services are not supported.
- High Availability is not supported.
- Virtual Routing and Forwarding (VRF) or Multi-VRF is not supported.

## Information About Cisco Unified Communications Gateway Services

### Extended Media Forking (XMF) Provider and XMF Connection

The XMF provider allows applications to monitor calls and trigger media forking on the calls and has the capability to service up to 32 applications. The XMF provider can invoke a call-based or a connection-based media forking using the Unified Communications (UC) API. After the media forking is invoked, it can preserve the media forking initiated by the web application if the WAN connection to the application is lost. The XMF provider also provides the recording tone to the parties involved in the call.

The XMF connection describes the relationship between an XMF call and the endpoint (or trunk) involved in the call. A connection abstraction maintained in the gateway has the following connection states:

- **IDLE**: This state is the initial state for all new connections. Such connections are not actively part of a telephone call, yet their references to the Call and Address objects are valid. Connections typically do not stay in the IDLE state for long and quickly transition to other states. The application may choose to be notified at this state using the event filters and if done, call/connection at the gateway provider will...
use the NotifyXmfConnectionData(CREATED) message to notify the application listener that a new connection is created.

- ADDRESS_COLLECT: In this state the initial information package is collected from the originating party and is examined according to the “dialing plan” to determine the end of collection of addressing information. In this state, the call in the gateway collects digits from the endpoint. No notification is provided.

- CALL_DELIVERY: On the originating side, this state involves selecting of the route as well as sending an indication of the desire to set up a call to the specified called party. On the terminating side, this state involves checking the busy/idle status of the terminating access and also informing the terminating message of an incoming call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (CALL_DELIVERY) message to notify the application listener.

- ALERTING: This state implies that the Address is being notified of an incoming call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (ALERTING) message to notify the application listener.

- CONNECTED: This state implies that a connection and its Address is actively part of a telephone call. In common terms, two parties talking to one another are represented by two connections in the CONNECTED state. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (CONNECTED) message to notify the application listener.

- DISCONNECTED: This state implies it is no longer part of the telephone call. A Connection in this state is interpreted as once previously belonging to this telephone call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (DISCONNECTED) message to notify the application listener.

**XMF Call-Based Media Forking**

In call-based media forking of the gateway, the stream from the calling party is termed as near-end stream and the stream from the called party is termed as far-end stream. The XMF provider actively handles single media forking request per session. Any new media forking request from the external application will override or stop the current forking instance and would start a new forking instance (to the appropriate target IP address or ports). After the media forking request is accepted, the XMF provider returns a response message and starts to fork media streams of a connection to the target forked streams. A NotifyXmfCallData message will be notified to the application for the updated media forking status, that is, FORK-FAILED, FORK_STARTED, or FORK_DONE.

**XMF Connection-Based Media Forking**

In connection-based media forking of the gateway, the incoming stream to the connection is termed as near-end stream and the outgoing stream of the connection is termed as far-end stream. The XMF provider actively handles single media forking request per session. Any new media forking request from the external application will override or stop the current forking instance and would start a new forking instance (to the appropriate target IP address or ports). After the media forking request is accepted, the XMF provider returns a response message and starts to fork media streams of a connection to the target forked streams.
A NotifyXmfConnectionData message will be notified to the application for the updated media forking status:

- FORK_FAILED—Media forking is setup failure. No forked RTP connections can be established to target RTP addresses.
- FORK_STARTED—Media forking is set up successfully. Both Tx (transmit) and Rx (receive) forked RTP connections are established and connected to target (farEnd and nearEnd) RTP addresses.
- FORK_DONE—Media forking is completed. Both Tx and Rx forked RTP connections are released.

**Cisco UC Gateway Services Media Forking API with Survivability TCL**

Cisco Unified Border Element (CUBE) supports Survivability TCL Script to co-exist with Cisco Unified Communication (UC) Services API.

Cisco UC Services API XMF interface supports media forking for all the calls controlled by survivability TCL script including the survivability re-attempted calls. Thus, all the calls controlled by survivability TCL script can be recorded when requested by Cisco UC Services XMF API.

Cisco Unified Communications Manager controlled Gateway recording utilizes XMF to trigger media forking on CUBE or SIP based PSTN gateways in the supported call flows.

**Note**

Media forking is allowed only for survivability TCL script supported by Cisco Unified Customer Voice Portal (CVP). CVP survivability TCL script is not supported in High Availability mode.

The following call scenarios are supported:

- Basic comprehensive call
• Calls with Refer Consume
• Calls with Mid-call failure
• Calls with alternative route with initial call failure

There are no configuration changes required for enabling CVP survivability TCL support with Cisco UC Gateway Services API.

**Media Forking for SRTP Calls**

- SRTP forking is supported in XCC and XMF application service providers and the supported APIs are RequestCallMediaForking, RequestCallMediaSetAttributes, and RequestConnectionMediaForking.

- SRTP forking is supported for SRTP-to-SRTP, SRTP-to-RTP, and RTP-to-SRTP calls.
  - For SRTP-to-SRTP calls, media forking on either leg would result in SRTP streams being forked.
  - For SRTP fallback calls, after the initial offer, CUBE will fall back to RTP. Media forking either call legs would result in RTP streams being forked.
  - For SRTP-to-RTP interworking calls, a digital signal processor (DSP) is required and involves transcoding. In this case, one leg would be SRTP and the other leg RTP.

- SRTP Crypto keys are notified over the API.
- Supports automatic stopping of media forking when stream changes from SRTP or to SRTP.
  - The optional mediaForkingReason tag in XMF or XCC Notify messages indicates that the forking has been stopped internally.
  - mediaForkingReason tag is only present when the connection changes state, such as mid-call re-INVITE. SRTP stream can change to RTP or SRTP stream can change keys mid-call.
  - mediaForkingReason tag is always accompanied by FORK_DONE.

**Crypto Tag**

For SRTP forking, the optional Crypto tag in NotifyXmfConnectionData or NotifyXmfCallData message indicates the context of an actively forked SRTP connection.

---

**Note**

The Crypto tag is only present in the notification message where FORK_STARTED tag is present.

The optional Crypto tag specifies the following:

- The Crypto suite used for encryption and authentication algorithm.
- The base64 encoded mastery key and salt used for encryption.

Crypto suite can be one of the two suites supported in IOS:

- AES_CM_128_HMAC_SHA1_32
- AES_CM_128_HMAC_SHA1_80
Example of SDP Data sent in an SRTP Call

<table>
<thead>
<tr>
<th>Original SIP SDP Crypto Offer</th>
<th>SIP SDP Crypto Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=0</td>
<td>v=0</td>
</tr>
<tr>
<td>o=CiscoSystemsSIP-GW-UserAgent 7826 3751 IN IP4 172.18.193.98</td>
<td>o=CiscoSystemsSIP-GW-UserAgent 7826 3751 IN IP4 172.18.193.98</td>
</tr>
<tr>
<td>s=SIP Call</td>
<td>s=SIP Call</td>
</tr>
<tr>
<td>c=IN IP4 172.18.193.98</td>
<td>c=IN IP4 172.18.193.98</td>
</tr>
<tr>
<td>t=0 0</td>
<td>t=0 0</td>
</tr>
<tr>
<td>m=audio 51372 RTP/SAVP 0</td>
<td>m=audio 49170 RTP/SAVP 0</td>
</tr>
<tr>
<td>a=rtpmap:0 PCMU/8000</td>
<td>a=rtpmap:0 PCMU/8000</td>
</tr>
<tr>
<td>a=crypto:1 AES_CM_128_HMAC_SHA1_32 inline:0RmdmcmVCPxSc3QGZINWpVLbHqXlfH4w8Soj</td>
<td>a=crypto:1 AES_CM_128_HMAC_SHA1_32 inline:NzB4d1BINUvLEw6UzF3WSJ+PsdFeGdUJShpXIZj</td>
</tr>
</tbody>
</table>

**Note**

The application is notified of the content in Crypto and inline SDP lines.

**Multiple XMF Applications and Recording Tone**

Multiple XMF allows multiple (maximum 32) web applications to register with the XMF provider as separate XMF applications and provide redundancy for the voice calls recording. Recording tone provides recording tone capability to the recording sessions. Recording tone is supported for IP to IP, IP to TDM, and TDM to TDM trunks.

An example topology is as shown below where 4 CUCM applications are deployed. CUCM triggers media forking request to Cisco UBE. Recording tone is played to the parties involved in the call based on the recordTone parameter set in the media forking request.
Media forking can be invoked using any of the following APIs:

- RequestXmfConnectionMediaForking
- RequestXmfCallMediaForking
- RequestXmfCallMediaSetAttributes

The “recordTone” parameter can be enabled in any of the above requests and recording tone will be played for the parties involved in the call. The “recordTone” parameter in the API request can have the following values:

- COUNTRY_US
- COUNTRY_AUSTRALIA
- COUNTRY_GERMANY
- COUNTRY_RUSSIA
- COUNTRY_SPAIN
- COUNTRY_SWITZERLAND
There is no difference in the recording tone beep when any country value is chosen. Recording tone beep is played at an interval of every 15 seconds. Digital signal processors and other resources are not utilized for playing recording tone even for transcoded calls. No specific configuration is required to enable or disable recording tone. By default, no recording tone is enabled.

If “recordTone” parameter is enabled only on the farEndAddr, then this tone is played only on the outgoing leg. Likewise, if enabled only on the nearEndAddr, then the tone is played only on the incoming leg. When enabled in both the far and near end, then recording tone is played on both the legs.

The RequestXmfConnectionMediaForking API allows insertion of recording tone on a per connection basis. There could be scenarios where one leg receives two recordTone insertion requests. When a leg receives recordTone insertion request, the nearEnd request always takes precedence over the farEnd request.

**Forking Preservation**

After media forking is initiated by the web application, the forking can be preserved to continue the recording, even if the WAN connection to the application is lost or if the application is unregistered.

*Figure 42: Forking Preservation*

The “preserve” parameter value can be set to TRUE or FALSE in any of the 3 forking requests (RequestXmfConnectionMediaForking, RequestXmfCallMediaForking, or RequestXmfCallMediaSetAttributes) from the application to Cisco UBE.

- If the “preserve” parameter received is TRUE, then forking will continue the recording, even if the WAN connection to application is lost or application is unregistered.
- If the “preserve” parameter received is FALSE, then forking will not continue the recording.
- If the “preserve” parameter is not received in the media forking request, then forking will not continue the recording.
How to Configure UC Gateway Services

Configuring Cisco Unified Communication IOS Services on the Device

SUMMARY STEPS

1. enable
2. configure terminal
3. ip http server
4. ip http max-connections value
5. ip http timeout-policy idle seconds life seconds requests value
6. http client connection idle timeout seconds
7. uc wsapi
8. message-exchange max-failures number
9. probing max-failures number
10. probing interval keepalive seconds
11. probing interval negative seconds
12. source-address ip-address
13. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3 ip http server</td>
<td>Enables the HTTP server (web server) on the system.</td>
</tr>
<tr>
<td>Example: Device(config)# ip http server</td>
<td></td>
</tr>
<tr>
<td>Step 4 ip http max-connections value</td>
<td>Sets the maximum number of concurrent connections to the HTTP server that will be allowed. The default value is 5.</td>
</tr>
<tr>
<td>Example: Device(config)# ip http max-connection 100</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>ip http timeout-policy idle seconds life seconds requests value</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>http client connection idle timeout seconds</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td>uc wsapi</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

| Step 8 | message-exchange max-failures *number*  
Example:  
Device(config-uc-wsapi)# message-exchange max-failures 2 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Configures the maximum number of failed message exchanges between the application and the provider before the provider stops sending messages to the application. Range is 1 to 3. Default is 1.</td>
</tr>
</tbody>
</table>

| Step 9 | probing max-failures *number*  
Example:  
Device(config-uc-wsapi)# probing max-failures 5 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Configures the maximum number of failed probing messages before the router unregisters the application. Range is 1 to 5. Default is 3.</td>
</tr>
</tbody>
</table>

| Step 10 | probing interval keepalive *seconds*  
Example:  
Device(config-uc-wsapi)# probing interval keepalive 255 |
| --- | --- |
| Purpose | Configures the time interval between probing messages when the session is in a keepalive state. Range is from 1 to 255 seconds. Default is 5 seconds.  
**Note**  
The keepalive timer restarts when a valid HTTP message is received from the UC services API. The following are valid HTTP messages that can restart the timer:  
• RESPONSE_XMF_REGISTER  
• RESPONSE_XMF_CONN_MEDIA_FORKING  
• SOLICIT_XMF_PROBING  
• NOTIFY_XMF_CONNECTION_DATA |

| Step 11 | probing interval negative *seconds*  
Example:  
Device(config-uc-wsapi)# probing interval negative 10 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Configures the interval between negative probing messages, in seconds.</td>
</tr>
</tbody>
</table>

| Step 12 | source-address *ip-address*  
Example:  
Device(config-uc-wsapi)# source-address 192.1.12.14 |
| --- | --- |
| Purpose | Configures the IP address (hostname) as the source IP address for the UC IOS service.  
**Note**  
The source IP address is used by the provider in the NotifyProviderStatus messages. |

| Step 13 | end  
Example:  
Device(config-uc-wsapi)# end |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
Configuring the XMF Provider

SUMMARY STEPS

1. enable
2. configure terminal
3. uc wsapi
4. source-address ip address
5. provider xmf
6. no shutdown
7. remote-url index url
8. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> uc wsapi</td>
<td>Enters Cisco Unified Communication IOS Service configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# uc wsapi</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> source-address ip address</td>
<td>Configures the source ip address.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# source-address 172.156.19.38</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> provider xmf</td>
<td>Enters XMF provider configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-uc-wsapi)# provider xmf</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> no shutdown</td>
<td>Activates XMF provider.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-uc-wsapi)# no shutdown</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Step 7</td>
<td></td>
</tr>
<tr>
<td>remote-url 1</td>
<td>Specifies the URL (IP address and port number) that the application uses to communicate with XMF provider. The XMF provider uses the IP address and port to authenticate incoming requests.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-uc-wsapi)# remote-url 1 <a href="http://test.com:8090/ucm_xmf">http://test.com:8090/ucm_xmf</a></td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-uc-wsapi)# end</td>
<td></td>
</tr>
</tbody>
</table>

**Verifying the UC Gateway Services**

The `show` commands can be entered in any order.

**SUMMARY STEPS**

1. `enable`
2. `show wsapi registration all`
3. `show wsapi registration xmf remote-url-index`
4. `show call media-forking`

**DETAILED STEPS**

**Step 1**

`enable`

Enables privileged EXEC mode.

**Example:**

Device> enable

**Step 2**

`show wsapi registration all`

Displays the details of applications registered. Each registered application is identified by a different ID.

**Example:**

Device# show wsapi registration all

```
Provider XMF
---------------------------------------------------------
registration index: 11
id: 2E7C3034:XMF:myapp:26
appUrl: http://pascal-lnx.cisco.com:8094/xmf
appName: myapp
provUrl: http://9.45.46.16:8090/cisco_xmf
prober state: STEADY
connEventsFilter: CREATED|REDIRECTED|ALERTING|CONNECTED|TRANSFERRED|CALL_DELIVERY|DISCONNECTED|HANDOFF_JOIN|HANDOFF_LEAVE
mediaEventsFilter: DTMF|MEDIA_ACTIVITY|MODE_CHANGE|TONE_DIAL|TONE_OUT_OF_SERVICE|TONE_SECOND_DIAL
```
Step 3  

**show wsapi registration xmf**  

Displays the details of only a particular XMF registered application with any ID ranging from 1 to 32.

**Example:**

```
Device# show wsapi registration xmf 1
```

Provider XMF
```
registration index: 1
id: 2E7C6423:XMF:myapp:28
appUrl: http://pascal-lnx.cisco.com:8096/xmf
appName: myapp
provUrl: http://9.45.46.16:8090/cisco_xmf
prober state: STEADY
connEventsFilter:
 CREATED|REDIRECTED|ALERTING|CONNECTED|TRANSFERRED|CALL_DELIVERY|DISCONNECTED|HANDOFF_JOIN|HANDOFF_LEAVE
mediaEventsFilter: DTMF|MEDIA_ACTIVITY|MODE_CHANGE|TONE_DIAL|TONE_OUT_OF_SERVICE|TONE_SECOND_DIAL
```

Step 4  

**show call media-forking**

Displays the forked stream information.

**Example:**

```
Device# show call media-forking
```

Warning: Output may be truncated if sessions are added/removed concurrently!
Session Call n/f Destination (port address)
187 BA near 45864 10.104.105.232
188 BA far 54922 10.104.105.232
189 B9 near 45864 10.104.105.232
190 B9 far 54922 10.104.105.232

FORK DONE Notifications

//WSAPI/INFRA/wsapi_send_outbound_message_by_provider_info:
*Dec 21 10:31:21.016 IST: //WSAPI/INFRA/0/9/546CF8:25:tx_contextp 15898C1C tx_id 19 context1 (0 0)
context2 (9 9):
wsapi_send_outbound_message_by_provider_info:
<?xml version="1.0" encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"><SOAP:Body>
<NotifyXmfConnectionData xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0"><msgHeader><transactionID>
546CF8:25</transactionID><registrationID>4CA5E4:XMF:myapp:4</registrationID></msgHeader><callData><callID>25</callID><state>
ACTIVE</state></callData><connData><connID>132</connID><state>ALERTING</state></connData><event><mediaForking>
<FORK DONE</mediaForking></event></NotifyXmfConnectionData></SOAP:Body></SOAP:Envelope>

FORK_FAILED Notification

//WSAPI/INFRA/wsapi_send_outbound_message_by_provider_info:
*Dec 21 10:31:21.016 IST: //WSAPI/INFRA/0/9/546CF8:25:tx_contextp 15898C1C tx_id 19 context1 (0 0)
context2 (9 9):
wsapi_send_outbound_message_by_provider_info:
<?xml version="1.0" encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"><SOAP:Body>
<NotifyXmfConnectionData xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0"><msgHeader><transactionID>
546CF8:25</transactionID><registrationID>4CA5E4:XMF:myapp:4</registrationID></msgHeader><callData><callID>25</callID><state>
ACTIVE</state></callData><connData><connID>132</connID><state>ALERTING</state></connData><event><mediaForking>
<FORK_FAILED</mediaForking></event></NotifyXmfConnectionData></SOAP:Body></SOAP:Envelope>

Troubleshooting Tips

You can use the following debug commands to troubleshoot the UC Gateway Services configurations.

- debug wsapi infrastructure all
- debug wsapi xcc all
- debug wsapi xmf all
- debug wsapi xmf messages
- debug wsapi infrastructure detail
- debug voip application
- debug voip application media forking
Configuration Examples for UC Gateway Services

Example: Configuring Cisco Unified Communication IOS Services

The following example shows how to configure the device for Cisco Unified Communication IOS Services and enable the HTTP server:

```
Device> enable
Device# configure terminal
Device(config)# ip http server
Device(config)# ip http max-connection 100
Device(config)# ip http timeout-policy idle 600 life 86400 requests 86400
Device(config)# http client connection idle timeout 600
Device(config)# uc wsapi
Device(config-uc-wsapi)# message-exchange max-failures 2
Device(config-uc-wsapi)# probing max-failures 5
Device(config-uc-wsapi)# probing interval keepalive 255
Device(config-uc-wsapi)# probing interval negative 10
Device(config-uc-wsapi)# source-address 192.1.12.14
Device(config-uc-wsapi)# end
```

Example: Configuring the XMF Provider

The following example shows how to enable the XMF providers. The configuration specifies the address and port that the application uses to communicate with the XMF provider:

```
Device> enable
Device# configure terminal
Device(config)# uc wsapi
Device(config-uc-wsapi)# provider xmf
Device(config-uc-wsapi)# no shutdown
Device(config-uc-wsapi)# remote-url 1 http://test.com:8090/ucm_xmf
Device(config-uc-wsapi)# end
```

Example: Configuring UC Gateway Services

```
uc wsapi
    message-exchange max-failures 5
    response-timeout 10
    source-address 192.1.12.14
    probing interval negative 20
    probing interval keepalive 250
    !
    provider xmf
    remote-url 1 http://pascal-lnx.cisco.com:8050/ucm_xmf
```
PART IX

CUBE Media Proxy

• CUBE Media Proxy, on page 525
Chapter 40

CUBE Media Proxy

The CUBE Media Proxy is a solution that provides multiple forking function and it is built on the CUBE architecture. Multiple forks are required for recorder redundancy and advanced media processing needs. The CUBE Media Proxy supports mandatory and optional recorders. CUSP supports the CUBE Media Proxy with functions such as recorder redundancy and load balancing.

- Feature Information for CUBE Media Proxy, on page 525
- Supported Platforms, on page 526
- Restrictions for CUBE Media Proxy, on page 526
- Information About Multiple Media Forking Using CUBE Media Proxy, on page 526
- How to Configure CUBE Media Proxy for Media Forking, on page 532
- Recording State Notification, on page 545
- Support for Mid-Call Features, on page 548
- Secure Recording of Secure Calls, on page 551
- Support for High Availability (HA), on page 552
- Media Latch, on page 553

Feature Information for CUBE Media Proxy

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 57: Feature Information for Recording Proxy

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBE Media Proxy</td>
<td>IOS XE Gibraltar Release 16.10.1a</td>
<td>The CUBE Media Proxy solution provides multiple forking function for redundancy and advanced media processing.</td>
</tr>
</tbody>
</table>
Supported Platforms

CUBE Media Proxy is supported on the following Cisco Router Platforms running on Cisco IOS XE Software Releases:

- Cisco 4000 Series-Integrated Services Routers (ISR G3 - ISR4331, ISR4351, ISR4431, ISR4451)
- Cisco Aggregated Services Routers (ASR - ASR1001-X, ASR1002-X, ASR1004 with RP2, ASR1006 with RP2)
- Cisco Cloud Services Routers (CSR1000V series)

Restrictions for CUBE Media Proxy

The CUBE Media Proxy does not support the following:

- Video recording
- Recording of calls from the endpoints that are registered with the Cloud. Example, Cisco Webex Teams Hybrid calls.
- Secure media (SRTP) forking of non secure calls
- SRTP fall back
- Midcall block
- Collocation with CUBE
- Midcall updates from the recorders such as pause or resume recording, RE-INVITE with SDP changes, INVITE that replaces header that is sent by recorders when they switch from active to standby CUBE Media Proxy.

Note

Midcall update "BYE" from the recorders is supported.

- If the primary recorder sends a=inactive in the response SDP, the same is forwarded to CUCM. Forking is not triggered to any of the recorders.

Information About Multiple Media Forking Using CUBE Media Proxy

Deployment Scenarios for CUBE Media Proxy Based Media Forking

The CUBE Media Proxy as a media forking solution supports the following functions:

- Recording forking to multiple recorders
Given below are the deployment scenarios of CUBE Media Proxy solution. You can deploy the CUBE Media Proxy for both external and internal calls. The information flow is described below:

1. External or internal call is set up between the endpoints.
2. The CUBE Media Proxy receives the media forking request.
3. The CUBE Media Proxy sets up sessions with recorders based on the proxy policy.
   - Mandatory recorder: Proxy policy is configured to set a recorder as mandatory. First, the CUBE Media Proxy tries to establish a connection with the mandatory recorder. Forking to the remaining recorders happen only if the connection with the mandatory recorder is successful.
   - Optional recorders: When the proxy policy is not configured, all the recorders are set as optional. CUBE Media Proxy continues to try to establish a connection with the remaining recorders even if any of the recorders fail.
Media Forking Topologies for CUBE Media Proxy

The following topologies support media forking using CUBE Media Proxy:

**Gateway Forking to CUBE Media Proxy**

The figure below illustrates the forking to CUBE Media Proxy via Gateway. This topology supports non secure forking.

*Figure 44: Gateway Forking to CUBE Media Proxy*

**Phone BiB Forking to CUBE Media Proxy**

The figure below illustrates the forking to CUBE Media Proxy via phone BiB. This topology supports secure forking of all calls and non secure forking of non secure calls.

---

**Note**

If the proxy server receives '486' response from the initial recording server, CUBE does not fork the INVITE to other recorders. To perform alternative routing, configure the **voice hunt user-busy** command in global configuration mode.

Example: `Router(config)# voice hunt user-busy`

4. The CUBE Media Proxy sends the invite to the recorders via CUSP.
5. CUSP performs load balancing and redundancy among recorders.

**Note**

The minimum supported Release of CUCM and CUSP required for the CUBE Media Proxy solution are CUCM Release 12.5.1 and CUSP Release 9.1.8 respectively.
Recording Metadata

Metadata is the information that a Recording Server receives from a recording client in a SIP session. The SIP header in the recording request that the CUBE Media Proxy receives from CUCM contains metadata that provides information on the communication session. The CUBE Media Proxy sends this SIP INVITE header with metadata to all the configured recorders.

A sample SIP header of a recording request is given below. A SIP header of size up to 583 bytes is supported.

```
From: "abcd" <sip:198101@10.200.25.137;
x-nearend;x-refci=27298698;x-nearendclusterid=NY-NJ-Labcluster;
x-nearenddevice=SEP2834A28318CE;
x-nearendaddr=198101;x-farendrefci=27298699;
x-farendclusterid=NY-NJ-Labcluster;x-farenddevice=AFIFIM-VI1;x-farendaddr=172001;
x-sessionid=696dd5d3f7755c6abdc01febf>
tag=14087-b35a5915-3167-4d6a-871d-c12121602bf-27298703
```

The following metadata from the SIP header is sent in all the requests and responses of a recording session. A maximum of 16 parameters are supported in the metadata of a SIP header.

```
;x-nearend;x-refci=27298698;x-nearendclusterid=NY-NJ-Labcluster;
x-nearenddevice=SEP2834A28318CE;
x-nearendaddr=198101;x-farendrefci=27298699;
x-farendclusterid=NY-NJ-Labcluster;x-farenddevice=AFIFIM-VI1;x-farendaddr=172001;
x-sessionid=696dd5d3f7755c6abdc01febf
```

For more information on recording metadata, see, Recording Metadata in CUCM Features and Services Guide.

Session Identifier

CUBE Media Proxy supports Session Identifier for tracking of a recording session. "Session-ID" header in the SIP request and response message provides the details of the identifier. The Session Identifier refers to the value of the identifier.

The Session-ID comprises of two Universally Unique Identifiers (UUIDs) corresponding to the initiator and recipient of the recording request respectively:
• Local UUID corresponds to UUID of the User Agent that sends a recording request to the participants of a recording session.
• Remote UUID corresponds to UUID of the User Agent that receives the recording request in a recording session.

Configuring Support for Session Identifier

Session Identifier support is enabled on CUBE Media Proxy by default and no additional configuration is required.

Session-ID Handling

The CUBE Media Proxy generates a unique UUID locally, and this UUID is passed as local UUID value in the Session-ID header of the following SIP request and response:
• Request to primary and optional recorders from CUBE Media Proxy.
• Response to CUCM from CUBE Media Proxy.

The following events are involved in the Session-ID handling by CUBE Media Proxy:

1. The Session-ID header in the recording request from CUCM to CUBE Media Proxy contains UUID of CUCM as local UUID and a "NULL" for remote UUID as shown in the following example.

   Session-ID: db248b6cbdc547bbc6c6fd6916eeb;remote=00000000000000000000000000000000

2. The CUBE Media Proxy locally generates a local UUID. The Session-ID header with this local UUID value and the remote UUID with a "NULL" is sent in the request to the primary recorder.

   Session-ID: 8dfb2f2e1d4c518db6122080fb8b1d83;remote=00000000000000000000000000000000

3. The CUBE Media Proxy receives a 200 OK response from the primary recorder. The Session-ID header of the response message contains UUID of the primary recorder as Local UUID, and the locally generated UUID by CUBE Media Proxy as the remote UUID.

   Session-ID: 4fd24d9121935531a7f8d750ad16e19;remote=8dfb2f2e1d4c518db6122080fb8b1d83

4. The CUBE Media Proxy sends a 200 OK response to CUCM. The Session-ID header of the response message contains locally generated UUID by CUBE Media Proxy as local UUID, and UUID of CUCM as remote UUID.

   Session-ID: 8dfb2f2e1d4c518db6122080fb8b1d83;remote=db248b6cbdc547bbc6c6fd6916eeb

5. CUBE Media Proxy sends a forking request to the remaining four recorders with Session-ID header containing locally generated UUID as the local UUID and a "NULL" value for the remote UUID.
6. CUBE Media Proxy receives 200 OK response from the remaining four recorders. The Session-ID header of the response message from each recorder contains UUID of the recorder as the local UUID and the locally generated UUID by the CUBE Media Proxy as the remote UUID.

7. CUBE Media Proxy sends a SIP Info Message to CUCM with status of the recorders. For more information on SIP Info Message, see SIP Info Messages from CUBE Media Proxy to CUCM, on page 545. The Session-ID header of the SIP Info Message contains locally generated UUID by CUBE Media Proxy as local UUID and the UUID of CUCM as remote UUID.

CUBE Media Proxy also handles recording request that does not contain a Session-ID header. Session-ID of the incoming recording requests does not have any negative impact on the CUBE Media Proxy. However, Session-ID is useful for troubleshooting the call flow.

Troubleshooting Tips

You can use the following show command to troubleshoot any issues with the Session Identifier.

- **show call active voice session-id** *WORD*

  *WORD* can be complete session identifier (local, remote, or both), or wildcard pattern of local or remote UUID. The valid wildcard patterns for search are *, 0-9, a-f, A-F.

```
Device# show call active voice session-id 8dfb2f2e1d4c518db6122080fb8b1d83
Telephony call-legs: 0
SIP call-legs: 6
H323 call-legs: 0
Call agent controlled call-legs: 0
  .
SessionIDLocaluuid=8dfb2f2e1d4c518db6122080fb8b1d83
SessionIDRemoteuuid=db248b6cbdc547bbc6c6fd6b6916eeb
  .
SessionIDLocaluuid=4fd24d912193531a7f8d750ad16e19
SessionIDRemoteuuid=8dfb2f2e1d4c518db6122080fb8b1d83
  .
SessionIDLocaluuid=4fd24d912193531a7f8d750ad17f20
SessionIDRemoteuuid=8dfb2f2e1d4c518db6122080fb8b1d83
  .
SessionIDLocaluuid=4fd24d912193531a7f8d750ad18g21
SessionIDRemoteuuid=8dfb2f2e1d4c518db6122080fb8b1d83
```
How to Configure CUBE Media Proxy for Media Forking

Following are the steps to configure CUBE Media Proxy for Media Forking:

1. Configuring Outbound Dial-Peers to the Recorders, on page 532
2. Configuring CUBE Media Proxy, on page 533
3. Configuring Inbound Dial-Peer from CUCM, on page 535

Configuring Outbound Dial-Peers to the Recorders

SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice dummy-recorder-dial-peer-tag voip
4. destination-pattern [+] string
5. session protocol sipv2
6. session target ipv4: [recording-server-destination-address | recording-server-mdns]
7. session transport tcp
8. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td>Configures a recorder dial peer and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>dial-peer voice dummy-recorder-dial-peer-tag voip</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# dial-peer voice 8000 voip</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>destination-pattern [+] string</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# destination-pattern 595995</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>session protocol sipv2</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>**session target ipv4:[recording-server-destination-address</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session target ipv4:198.51.100.1</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>session transport tcp</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session transport tcp</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# end</td>
</tr>
</tbody>
</table>

**Configuring CUBE Media Proxy**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. media profile recorder `profile-tag`
4. media-recording proxy `[dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5]`
5. proxy policy mandatory `dial-peer-tag`
6. exit
7. media class `tag`
8. recorder profile `tag`
9. exit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  Example:  
  ```  
  Device> enable  
  ``` |
| **Step 2** configure terminal | Enters global configuration mode.  
  Example:  
  ```  
  Device# configure terminal  
  ``` |
| **Step 3** media profile recorder `profile-tag` | Configures the media profile recorder and enters media profile configuration mode.  
  Example:  
  ```  
  Device(config)# media profile recorder 100  
  ``` |
| **Step 4** media-recording proxy `[dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5]` | Configures the dial-peers for recording. The proxy configures the first dial-peer of the sequence for establishing a B2B call and the remaining dial-peers for media forking.  
  Note: You can specify a maximum of five dial-peer tags.  
  Example:  
  ```  
  Device(cfg-mediaprofile)# media-recording proxy 8000 8001 8002  
  ``` |
| **Step 5** proxy policy mandatory `dial-peer-tag` | Configures the specified dial-peer for establishing a B2B call.  
  Example:  
  ```  
  ``` |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(cfg-mediaprofile)# proxy policy mandatory 8001</td>
<td><strong>Note</strong> You can specify only one dial-peer tag for establishing a B2B call. The dial-peer tag that is specified, must be one of the dial-peer tags configured using the <code>media-recording proxy</code> command. Each Recorder Group receives one fork. When the recording session policy control is set to mandatory, the recording session failure for Recorder Group 1, fails the recording sessions for remaining Recorder Groups that are configured. The command <code>proxy policy mandatory</code> will initiate recording only when it is configured along with the command <code>media recording proxy</code> that is mentioned in Step 4.</td>
</tr>
</tbody>
</table>

**Step 6**

**Example:**

```
Device(cfg-mediaprofile)# exit
```

Exits media profile configuration mode.

**Step 7**

**media class tag**

**Example:**

```
Device(config)# media class 100
```

Configures a media class and enters media class configuration mode.

**Step 8**

**recorder profile tag**

**Example:**

```
Device(cfg-mediaclass)# recorder profile 100
```

Configures the media profile recorder.

**Step 9**

**exit**

**Example:**

```
Device(cfg-mediaclass)# exit
```

Exits media class configuration mode.

---

**Configuring Inbound Dial-Peer from CUCM**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice call-manager-dial-peer-tag voip
4. incoming uri {from | request | to | via } tag
5. media-class tag
6. exit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | enable  
Example: Device> enable | Enables privileged EXEC mode.  
- Enter your password if prompted. |
| Step 2 | configure terminal  
Example: Device# configure terminal | Enters global configuration mode. |
| Step 3 | dial-peer voice call-manager-dial-peer-tag voip  
Example: Device(config)# dial-peer voice 1000 voip | Configures an inbound dial peer and enters dial peer voice configuration mode. |
| Step 4 | incoming uri {from | request | to | via} tag  
Example: Device(config-dial-peer)# incoming uri via 101 | Configures the voice class that is used to match the VoIP dial-peer to the URI of an incoming call from CUCM via the header in an incoming SIP Invite message.  
**Note** For more information on incoming uri command, see incoming uri. |
| Step 5 | media-class tag  
Example: Device(config-dial-peer)# media-class 100 | Configures media class on the inbound dial peer from CUCM. |
| Step 6 | exit  
Example: Device(cfg-mediaclass)# exit | Exits media class configuration mode. |

### Verifying Media Forking Using CUBE Media Proxy Configuration

Perform this task to verify the configuration of the CUBE Media Proxy for multiple media forking. The `show` and `debug` commands can be entered in any order.

### SUMMARY STEPS

1. enable
2. show voip rtp connections
3. show voip recmsp session
4. show voip recmsp session detail call-id call-id
5. show voip rtp forking
6. show call active voice compact
7. show sip-ua calls
8. show media-proxy sessions
9. show media-proxy sessions summary
10. show media-proxy sessions summary history
11. show media-proxy sessions call-id call-id
12. show media-proxy sessions session-id WORD
13. show media-proxy sessions metadata-session-id x-session-id
14. debug ccsip messages (for audio calls)
15. Enter one of the following:
   • debug ccsip all
   • debug voip recmsp all
   • debug voip ccapi all
   • debug voip fpi all (for ASR devices only)

DETAILED STEPS

Step 1  enable
Enables privileged EXEC mode.
Example:
Device> enable

Step 2  show voip rtp connections
Displays Real-Time Transport Protocol (RTP) connections. Every recording session has one rtp connection each towards
the recorders and one rtp connection towards the recording source that is on the inbound side.
Example:
Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 3
Port range not configured
Min   Max   Ports   Ports
Media-Address Range  Port   Port   Available   Reserved   In-use
Global Media Pool   8000    48198  19999      101        3
VoIP RTP active connections :
No.   CallId   dstCallId   LocalRTP   RmtRTP   LocalIP   RemoteIP   MPSS   VRF
1     248      249        8218      8372    198.51.100.1  192.0.2.1  NO   NA
2     249      248        8220      9000    198.51.100.1  192.0.2.1  NO   NA
3     252      251        8222      9238    198.51.100.1  192.0.2.1  NO   NA
Found 3 active RTP connections

Step 3  show voip recmsp session
Displays active recording Media Service Provider (MSP) session information internal to CUBE Media Proxy.
Example:
Device# show voip recmsp session
RECMSP active sessions:
MSP Call-ID  AnchorLeg Call-ID  ForkedLeg Call-ID
143        141         145
Found 1 active sessions

**Step 4**

**show voip recmsp session detail call-id call-id**

Displays detailed information about the recording MSP Call ID.

**Example:**

```
Device# show voip recmsp session detail call-id 145
RECMSP active sessions:
Detailed Information
-------------------------
Recording MSP Leg Details:
Call ID: 143
GUID : 7C5946D38ECD

AnchorLeg Details:
Call ID: 141
Forking Stream type: voice-nearend
Participant: 708090

Non-anchor Leg Details:
Call ID: 140
Forking Stream type: voice-farend
Participant: 10000

Forked Leg Details:
Call ID: 145
Voice Near End Stream CallID 145
Stream State ACTIVE
Found 1 active sessions
```

**Step 5**

**show voip rtp forking**

Displays RTP media-forking connections.

**Example:**

```
Device# show voip rtp forking
VoIP RTP active forks :
Fork 1
  stream type voice-only (0): count 0
  stream type voice+dtmf (1): count 0
  stream type dtmf-only (2): count 0
  stream type voice-nearend (3): count 1
    remote ip 192.0.2.1, remote port 38526, local port 18648
    codec g711ulaw, logical ssrc 0x53
    packets sent 29687, packets received 0
  stream type voice+dtmf-nearend (4): count 0
  stream type voice+dtmf-farend (6): count 0
  stream type video (7): count 0
  stream type video-nearend (8): count 0
  stream type video-farend (9): count 0
  stream type application (10): count 0
```

**Step 6**

**show call active voice compact**

Displays a compact version of voice calls in progress. An additional call leg is displayed for media forking.

**Example:**
Step 7

**show sip-ua calls**

Displays active user agent client (UAC) and user agent server (UAS) information on SIP calls.

**Example:**

```
Device# show sip-ua calls
Total SIP call legs: 3, User Agent Client: 2, User Agent Server: 1
```

```
SIP UAC CALL INFO
Call 1
SIP Call ID : 4091A49B-308911E8-8008EC4C-8D01D66C@192.0.2.1
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 808808
Called Number : 8453
Called URI : 
Bit Flags : 0xC04018 0x80000100 0x80
CC Call ID : 2
Local UUID : c7351800dd135daba19758eac6b1dd70
Remote UUID : ab9f482302156aaa8d6e2e04aaa2b96
Source IP Address (Sig ) : 192.0.2.1
Destn SIP Req Addr:Port : [192.0.2.2]:9312
Destn SIP Resp Addr:Port: [192.0.2.2]:9312
Destination Name : 
Number of Media Streams : 1
Number of Active Streams: 1
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 2
Stream Type : voice-only (0)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [192.0.2.1]:8002
Media Dest IP Addr:Port : [192.0.2.2]:9000
Mid-Call Re-Assocation Count: 0
SRTP-RTP Re-Assocation DSP Query Count: 0
Options-Ping ENABLED:NO ACTIVE:NO
Call 2
SIP Call ID : 4097E619-308911E8-8008EC4C-8D01D66C@8.43.33.213
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 
Called Number : 8453
Called URI : 
Bit Flags : 0xC04018 0x80000100 0x80
```
CC Call ID: 5
Local UUID:
Remote UUID:
Source IP Address (Sig): 192.0.2.1
Destn SIP Req Addr:Port: [192.0.2.2]:9322
Destn SIP Resp Addr:Port: [192.0.2.2]:9322
Destination Name:
Number of Media Streams: 1
Number of Active Streams: 1
RTP Fork Object: 0x0
Media Mode: flow-through
Media Stream 1
State of the stream: STREAM_ACTIVE
Stream Call ID: 5
Stream Type: voice-nearend (3)
Stream Media Addr Type: 1
Negotiated Codec: g711ulaw (160 bytes)
Codec Payload Type: 0
Negotiated Dtmf-relay: inband-voice
Dtmf-relay Payload Type: 0
QoS ID: -1
Local QoS Strength: BestEffort
Negotiated QoS Strength: BestEffort
Negotiated QoS Direction: None
Local QoS Status: None
Media Source IP Addr:Port: [192.0.2.1]:8004
Media Dest IP Addr:Port: [192.0.2.2]:9238
Mid-Call Re-Assocation Count: 0
SRTP-RTP Re-Assocation DSP Query Count: 0
Options-Ping: ENABLED:NO  ACTIVE:NO
Number of SIP User Agent Client (UAC) calls: 2
SIP UAS CALL INFO
Call 1
SIP Call ID: 1-25986@192.0.2.2
State of the call: STATE_ACTIVE (7)
Substate of the call: SUBSTATE_NONE (0)
Calling Number: 808808
Called Number: 8453
Called URI: sip:8453@192.0.2.1:5060
Bit Flags: 0xC0401C 0x10000100 0x4
CC Call ID: 1
Local UUID: ab9f4823802156aaa8d62e04aaa2b96
Remote UUID: c7351800dd135daba19758eac6b1dd70
Source IP Address (Sig): 192.0.2.1
Destn SIP Req Addr:Port: [192.0.2.2]:10344
Destn SIP Resp Addr:Port: [192.0.2.2]:10344
Destination Name: 8.0.0.200
Number of Media Streams: 1
Number of Active Streams: 1
RTP Fork Object: 0x0
Media Mode: flow-through
Media Stream 1
State of the stream: STREAM_ACTIVE
Stream Call ID: 1
Stream Type: voice-only (0)
Stream Media Addr Type: 1
Negotiated Codec: g711ulaw (160 bytes)
Codec Payload Type: 0
Negotiated Dtmf-relay: inband-voice
Dtmf-relay Payload Type: 0
QoS ID: -1
Local QoS Strength: BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [192.0.2.1]:8000
Media Dest IP Addr:Port : [192.0.2.2]:8372
Mid-Call Re-Assocation Count: 0
SRTP-RTP Re-Assocation DSP Query Count: 0
Options-Ping ENABLED: NO ACTIVE: NO
Number of SIP User Agent Server(UAS) calls: 1

Step 8 show media-proxy sessions

Displays the inbound and forked Call-ID, Session-ID, and dial peer tag details of the active recording sessions. The "Secure" field in the command output is tagged Y if the recording session is secure and N if the recording session is non-secure.

Example:

Device# show media-proxy sessions

<table>
<thead>
<tr>
<th>No.</th>
<th>Call-ID</th>
<th>Session-ID</th>
<th>Dialpeer Tag</th>
<th>Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7253/-</td>
<td>a234a20672ce596d969c59ee9767f127; aaaaaaaaaaaaaaaaaaaaaaaaaaaa</td>
<td>10</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>7253/7254</td>
<td>bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb; a234a20672ce596d969c59ee9767f127</td>
<td>11</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>7253/7260</td>
<td>bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb; a234a20672ce596d969c59ee9767f127</td>
<td>12</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>7253/7262</td>
<td>bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb; a234a20672ce596d969c59ee9767f127</td>
<td>13</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>7253/7264</td>
<td>bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb; a234a20672ce596d969c59ee9767f127</td>
<td>14</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>7253/7266</td>
<td>bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb; a234a20672ce596d969c59ee9767f127</td>
<td>15</td>
<td>N</td>
</tr>
</tbody>
</table>

Step 9 show media-proxy sessions summary

Displays the active recording session details such as the dial peer tag, IP address, port number, number of failed recording sessions, and total number of recording sessions.

Example:

Device# show media-proxy sessions summary

<table>
<thead>
<tr>
<th>No.</th>
<th>Inbound/Forked</th>
<th>Dialpeer-Tag</th>
<th>IP:Port</th>
<th>Total/Failed Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Forked</td>
<td>100</td>
<td>ipv4:192.0.2.2:6680</td>
<td>2/0</td>
</tr>
<tr>
<td>2</td>
<td>Forked</td>
<td>200</td>
<td>ipv4:192.0.2.2:6220</td>
<td>2/0</td>
</tr>
<tr>
<td>3</td>
<td>Inbound</td>
<td>5678</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Step 10 show media-proxy sessions summary history

Displays the details of all the completed recording sessions. Once the recording session is completed, the session details moves to the call history table. Total Sessions refers to the cumulative value of all the completed recording sessions.

Example:

Device# show media-proxy sessions summary history

<table>
<thead>
<tr>
<th>No.</th>
<th>Inbound/Forked</th>
<th>Dialpeer Tag</th>
<th>IP:Port</th>
<th>Total/Failed Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Verifying Media Forking Using CUBE Media Proxy Configuration

1 Inbound 5678 2/0
2 Forked 100 ipv4:192.0.2.2:6680 2/0
3 Forked 200 ipv4:192.0.2.2:6220 2/0

Note
To clear the history data for the CUBE Media Proxy recording sessions, use the command `clear media-proxy sessions summary history`, in privileged EXEC mode.

Device# clear media-proxy sessions summary history

Following is the sample output of the command `show media-proxy sessions summary history` after clearing the CUBE Media Proxy recording session history data.

Device# show media-proxy sessions summary history

<table>
<thead>
<tr>
<th>No.</th>
<th>Inbound/Forked</th>
<th>Dialpeer Tag</th>
<th>IP:Port</th>
<th>Total/Failed Sessions</th>
</tr>
</thead>
</table>

Step 11
`show media-proxy sessions call-id call-id`

Displays the details of the inbound leg and all the forked legs that are associated with the specified SIP leg call-ID. MSP call-ID is not a valid call-ID for this command. Specify the CCAPI call identifier of the SIP leg.

Example:

Device# show media-proxy sessions call-id 2
CC Call-ID: 1 Inbound-leg
Dur: 00:00:15 tx: 0/0 rx: 1484/296800 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:6009 Local-Addr: 192.0.2.1:8000 rtt:0ms pl:0/0/0ms
Dialpeer-Tag: 100 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

CC Call-ID: 2 Forked-leg (Primary)
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:6000 Local-Addr: 192.0.2.1:8002 rtt:0ms pl:0/0/0ms
Dialpeer-Tag: 200 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 7 Forked-leg
Dur: 00:00:15 tx: 1480/296000 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:6001 Local-Addr: 192.0.2.1:8004 rtt:0ms pl:0/0/0ms
Dialpeer-Tag: 300 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: cccccccccccccccccccccccccccccccccccc RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 9 Forked-leg
Dur: 00:00:15 tx: 1479/295800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:6004 Local-Addr: 192.0.2.1:8006 rtt:0ms pl:0/0/0ms
Dialpeer-Tag: 400 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: cccccccccccccccccccccccccccccccccccc RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 11 Forked-leg
Dur: 00:00:15 tx: 1479/295800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:6005 Local-Addr: 192.0.2.1:8008 rtt:0ms pl:0/0/0ms
Dialpeer-Tag: 500 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: cccccccccccccccccccccccccccccccccccc RemoteUUID: 6bde661e9767590b930f3427ad6e94e9
Step 12

**show media-proxy sessions session-id WORD**

Displays the details of the Media Proxy recording sessions that are associated with the specified session-ID. To display the details of a specific call-leg, specify the complete session ID string as, `local-uuid;remote=remote-uuid`. Tokens that are allowed for `WORD` are '*', `[0-9]`, `[a-f]`, and `[A-F]`.

**Example:**

```
Device# show media-proxy sessions session-id 6bde661e9767590b930f3427ad6e94e9
CC Call-ID: 1 Inbound-leg
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0 delay: 0/0ms
Remote-Addr: 198.51.100.1:6009 Local-Addr: 192.0.2.3:8002 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 200 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 2 Forked-leg (Primary)
Dur: 00:00:15 tx: 1480/296000 rx: 0/0 lost: 0/0 delay: 0/0ms
Remote-Addr: 198.51.100.1:6001 Local-Addr: 192.0.2.3:8004 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 300 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ccccccccccccccddcdccccccccccccccccccc RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 7 Forked-leg
Dur: 00:00:15 tx: 1480/296000 rx: 0/0 lost: 0/0 delay: 0/0ms
Remote-Addr: 198.51.100.1:6004 Local-Addr: 192.0.2.3:8006 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 400 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ccccccccccccccddcdccccccccccccccccccc RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 9 Forked-leg
Dur: 00:00:15 tx: 1479/295800 rx: 0/0 lost: 0/0 delay: 0/0ms
Remote-Addr: 198.51.100.1:6005 Local-Addr: 192.0.2.3:8008 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 500 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ccccccccccccccddcdccccccccccccccccccc RemoteUUID: 6bde661e9767590b930f3427ad6e94e9
```

Step 13

**show media-proxy sessions metadata-session-id x-session-id**

Displays the details of the Media Proxy recording sessions based on the x-session-id present in the "From" header of the INVITE from CUCM.
Example:

Device# show media-proxy sessions metadata-session-id 696dd5d3f7755c6abdc438e93d01febf
CC Call-ID: 77 Inbound-leg
Dur: 00:00:46 tx: 0/0 rx: 3105/578880 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:8010 Local-Addr: 198.51.100.1:8048 rtt: Oms pl: 0/0/0ms
Dialpeer-Tag: 1 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 528b282b804c5fd098eaba3696c00de2 RemoteUUID: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

CC Call-ID: 78 Forked-leg (Primary)
Dur: 00:00:46 tx: 3105/578880 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:8014 Local-Addr: 198.51.100.1:8050 rtt: Oms pl: 0/0/0ms
Dialpeer-Tag: 2 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb RemoteUUID: 528b282b804c5fd098eaba3696c00de2

CC Call-ID: 84 Forked-leg
Dur: 00:00:46 tx: 3100/577880 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:8018 Local-Addr: 198.51.100.1:8052 rtt: Oms pl: 0/0/0ms
Dialpeer-Tag: 3 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb RemoteUUID: 528b282b804c5fd098eaba3696c00de2

CC Call-ID: 86 Forked-leg
Dur: 00:00:46 tx: 3101/578080 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:8022 Local-Addr: 198.51.100.1:8054 rtt: Oms pl: 0/0/0ms
Dialpeer-Tag: 4 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb RemoteUUID: 528b282b804c5fd098eaba3696c00de2

CC Call-ID: 88 Forked-leg
Dur: 00:00:46 tx: 3101/578080 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:8026 Local-Addr: 198.51.100.1:8056 rtt: Oms pl: 0/0/0ms
Dialpeer-Tag: 5 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb RemoteUUID: 528b282b804c5fd098eaba3696c00de2

CC Call-ID: 91 Forked-leg
Dur: 00:00:46 tx: 3101/578080 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 198.51.100.1:8030 Local-Addr: 198.51.100.1:8058 rtt: Oms pl: 0/0/0ms
Dialpeer-Tag: 6 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb RemoteUUID: 528b282b804c5fd098eaba3696c00de2

Step 14  debug csip messages(for audio calls)

The CUBE Media Proxy sends INVITEs to the recorders with a single stream, which successfully forks the primary call to the recorders. INVITEs to recorders have a single m-line with a send-only attribute.

Step 15  Enter one of the following:

• debug csip all
• debug voip recmsp all
• debug voip ccap all
• debug voip fpi all (for ASR devices only)

Displays detailed debug messages.
Recording State Notification

SIP Info Messages from CUBE Media Proxy to CUCM

The CUBE Media Proxy sends SIP Info message to CUCM for providing the consolidated status of all the recorders, after the recording session is established or tried with all the recorders.

A SIP Info message is sent during the following stages of a recording session:

1. Initial Call: After receiving response from all the configured recorders during the initial call, a SIP Info message with status of each recorder is sent to the initiator of the recording session.

2. Mid-Call: When status of any of the recorders changes during the midcall, another SIP Info message with status of each recorder is sent to the initiator of the recording session. The scenario corresponds to any of the recorders sending a "BYE" or rejecting a midcall RE-INVITE.

Note

All the scenarios in the upcoming sections explains CUBE Media Proxy supporting two recorders. However, CUBE Media Proxy supports five recorders.

XML Format of a SIP Info Message

The Content-Type header present in the SIP Info message is

Content-Type: application/x-cisco-proxy-recording-status+xml

The XML format of a SIP info message is given below.

```xml
<recorderList>
  <recorder>
    <uri>recorder1</uri>
    <recordertype>Mandatory</recordertype>
    <status>Success</status>
    <errormessage>null</errormessage>
  </recorder>
  <recorder>
    <uri>recorder2</uri>
    <recordertype>Mandatory</recordertype>
    <status>Failed</status>
    <errormessage>SIP error code received from Recorder</errormessage>
  </recorder>
</recorderList>
```
### Table 58: Details of XML Tag and Data Type

<table>
<thead>
<tr>
<th>XML Tag</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri (Mandatory)</td>
<td>String</td>
</tr>
<tr>
<td>recordertype (Mandatory)</td>
<td>Enum (Mandatory, Optional)</td>
</tr>
<tr>
<td>status (Mandatory)</td>
<td>Enum (Success, Failed)</td>
</tr>
<tr>
<td>errormessage (Optional)</td>
<td>String</td>
</tr>
</tbody>
</table>

### SIP Info Message Sent During the Initial Call

**SIP Info Message Sent During the Initial Call (All the Recorders as Optional)**

For information on how to configure the recorders as Optional, see Step 3 and Step 4 of Configuring CUBE Media Proxy, on page 533.

The SIP Info Message sent during a recording session depends on the scenarios that are given in the following table.

### Table 59: Call scenarios and recorder status during the initial call with all recorders as optional

<table>
<thead>
<tr>
<th>Scenario</th>
<th>&lt;status&gt; of recorder-1 in a SIP Info Message</th>
<th>&lt;status&gt; of recorder-2 in a SIP Info Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call to the primary recorder recorder-1 is established and forking to recorder-2 is triggered successfully.</td>
<td>&lt;success&gt;</td>
<td>&lt;success&gt;</td>
</tr>
<tr>
<td>Call to the primary recorder recorder-1 is established and forking to recorder-2 is rejected with 503 Service Unavailable.</td>
<td>&lt;success&gt;</td>
<td>&lt;failure&gt;</td>
</tr>
<tr>
<td>Call to the primary recorder recorder-1 is established and there is no response from recorder-2 to the forking request.</td>
<td>&lt;success&gt;</td>
<td>&lt;failure&gt;</td>
</tr>
<tr>
<td>Call to the recorder recorder-1 and recorder-2 is rejected with 503 Service Unavailable.</td>
<td>&lt;failure&gt;</td>
<td>&lt;failure&gt;</td>
</tr>
<tr>
<td>recorder-1 and recorder-2 are down.</td>
<td>&lt;failure&gt;</td>
<td>&lt;failure&gt;</td>
</tr>
</tbody>
</table>
Scenario | <status> of recorder-1 in a SIP Info Message | <status> of recorder-2 in a SIP Info Message
---|---|---
recorder-1 and recorder-2 responds to the call with a 488 Not Acceptable Here response. | <failure> | <failure>

recorder-1 and recorder-2 responds to the call with a 600 Busy Everywhere response. | <failure> | <failure>

**Note**

After a SIP Info Message is sent, a 200 OK response is received from the initiator of the recording session.

SIP Info Message Sent During the Initial Call (One Recorder as Mandatory and Remaining as Optional)

For information on how to configure the recorders as Mandatory, see Step 3, Step 4 and, Step 5 of Configuring CUBE Media Proxy, on page 533.

The SIP Info Message sent during a recording session depends on the scenarios that are given in the following table.

**Table 60: Call scenarios and recorder status during the initial call with a mandatory recorder**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>&lt;status&gt; of recorder-1 in a SIP Info Message</th>
<th>&lt;status&gt; of recorder-2 in a SIP Info Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call to the mandatory recorder recorder-1 is established and forking to the optional recorder recorder-2 is triggered successfully.</td>
<td>&lt;success&gt;</td>
<td>&lt;success&gt;</td>
</tr>
</tbody>
</table>

Call to the mandatory recorder recorder-1 is rejected with a failure message and hence the optional recorder recorder-2 is not tried.

<failure>  
**Note** Error code is sent in the <errormessage>.

Call to the mandatory recorder recorder-1 is established and when the optional recorder recorder-2 is tried, the mandatory recorder disconnects with a BYE.

<failure>  
**Note** BYE is sent in the <errormessage>.  
**Note** The connection to the optional recorder is cancelled as the primary recorder disconnects.
After the call is established with the mandatory recorder recorder-1 and the optional recorder recorder-2, the mandatory recorder disconnects with a *BYE*.

### Note

After a SIP Info Message is sent, a 200 *OK* response is received from the initiator of the recording session. CUCM sends a 415 *Unsupported Media Type* message if the INFO sent from CUBE Media Proxy has a malformed XML body.

### Support for Mid-Call Features

CUBE Media Proxy supports mid-call signalling events that involve RE-INVITEs from CUCM to the recorders. The CUBE Media Proxy handles the RE-INVITEs that request for an SDP change, codec change, session refresh and, direction change in SDP. Based on the change in the status of the recorders during the mid-call, SIP Info messages are sent from the recorders to the CUCM.

### Note

CUBE Media Proxy also supports mid-call RE-INVITE with no SDP change.

The CUBE Media Proxy supports the mid-call features given in the table below, when all the recorders are configured as optional.

### Table 61: Details of Mid-Call Scenarios and the events (optional recorders)

<table>
<thead>
<tr>
<th>Mid-Call Scenario (optional recorders)</th>
<th>Events in the Mid-Call Scenario</th>
</tr>
</thead>
</table>
| Mid-call session refresh              | 1. CUCM sends a mid-call RE-INVITE with no SDP change.  
                                        | 2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder only. |
| Mid-call directory change             | 1. CUCM sends a mid-call RE-INVITE with direction change in SDP.  
                                        | 2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.  
                                        | 3. The primary recorder sends a 200K response to CUBE Media Proxy, which sends this response to CUCM.  
<pre><code>                                    | 4. The CUBE Media Proxy sends the RE-INVITE to the remaining recorders. |
</code></pre>
<table>
<thead>
<tr>
<th>Mid-Call Scenario (optional recorders)</th>
<th>Events in the Mid-Call Scenario</th>
</tr>
</thead>
</table>
| Mid-call codec change with a 2000K from the primary recorder | 1. CUCM sends a mid-call RE-INVITE with a codec change in SDP.  
2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.  
3. The primary recorder sends a 2000K response for the codec change to CUBE Media Proxy, which sends this response to CUCM.  
4. The CUBE Media Proxy sends the RE-INVITE containing new codec to the remaining recorders.  
5. The remaining recorders send a 2000K response to CUBE Media Proxy. |
| Mid-call codec change with failure status of the primary recorder | 1. CUCM sends a mid-call RE-INVITE with a codec change in SDP.  
2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.  
3. The primary recorder sends a 488 error response for the codec change to CUBE Media Proxy, which sends this response to CUCM.  
4. REINVITE is not sent to the remaining recorders. |
| Mid-call codec change with failure status of the recorders | 1. CUCM sends a mid-call RE-INVITE with a codec change in SDP.  
2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.  
3. The primary recorder sends a 2000K response for the codec change to CUBE Media Proxy, which sends this response to CUCM.  
4. The CUBE Media Proxy sends the RE-INVITE containing new codec to the remaining recorders.  
5. Two of the recorders send a 488 error response for the codec change to CUBE Media Proxy and the remaining recorders respond with a 2000K.  
6. CUBE Media Proxy sends a BYE to the failed recorders and disconnects them. |
| Mid-call SRTP suite or SRTP key change with a 2000K from the primary recorder | 1. CUCM sends a mid-call RE-INVITE with an SRTP key change.  
2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.  
3. The primary recorder sends a 2000K response for the codec change to CUBE Media Proxy, which sends this response to CUCM.  
4. The CUBE Media Proxy sends the RE-INVITE containing new SRTP key to the remaining recorders.  
5. The remaining recorders send a 2000K response to CUBE Media Proxy. |
### Mid-Call Scenario (optional recorder)

<table>
<thead>
<tr>
<th>Mid-call SRTP suite or SRTP key change with failure status of the primary recorder</th>
<th>Events in the Mid-Call Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. CUCM sends a mid-call RE-INVITE with an SRTP key change.</td>
<td></td>
</tr>
<tr>
<td>2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.</td>
<td></td>
</tr>
<tr>
<td>3. The primary recorder sends a 488 error response for the SRTP key change to CUBE Media Proxy, which sends this response to CUCM.</td>
<td></td>
</tr>
<tr>
<td>4. REINVITE is not sent to the remaining recorders.</td>
<td></td>
</tr>
</tbody>
</table>

### Mid-call SRTP suite or SRTP key change with failure status of the recorders

<table>
<thead>
<tr>
<th>Mid-call SRTP suite or SRTP key change with failure status of the recorders</th>
<th>Events in the Mid-Call Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. CUCM sends a mid-call RE-INVITE with an SRTP change.</td>
<td></td>
</tr>
<tr>
<td>2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.</td>
<td></td>
</tr>
<tr>
<td>3. The primary recorder sends a 200OK response for the SRTP key change to CUBE Media Proxy, which sends this response to CUCM.</td>
<td></td>
</tr>
<tr>
<td>4. The CUBE Media Proxy sends the RE-INVITE containing new SRTP key to the remaining recorders.</td>
<td></td>
</tr>
<tr>
<td>5. Two of the recorders send a 488 error response for the SRTP key change to CUBE Media Proxy and the remaining recorders respond with a 200OK.</td>
<td></td>
</tr>
<tr>
<td>6. CUBE Media Proxy sends a BYE to the failed recorders and disconnects them.</td>
<td></td>
</tr>
</tbody>
</table>

The CUBE Media Proxy supports the mid-call features given in the table below, when a mandatory recorder is configured.

**Table 62: Details of Mid-Call Scenarios and the events (mandatory recorder)**

<table>
<thead>
<tr>
<th>Mid-Call Scenario (mandatory recorder)</th>
<th>Events in the Mid-Call Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-call SDP change with failure status of the mandatory recorder</td>
<td>1. CUCM sends a mid-call RE-INVITE with any of the SDP changes.</td>
</tr>
<tr>
<td>2. The CUBE Media Proxy sends the RE-INVITE to the primary recorder.</td>
<td></td>
</tr>
<tr>
<td>3. The primary recorder sends a 488 error response or 503 error response for the SDP change to CUBE Media Proxy, which sends this response to CUCM.</td>
<td></td>
</tr>
<tr>
<td>4. REINVITE is not sent to the remaining recorders.</td>
<td></td>
</tr>
</tbody>
</table>
A mid-call SIP Info message containing status of the recorders is sent to the CUCM when the mid-call RE-INVITEs to the recorders fail.

**Note**

During the mid-call, the following events happen when the primary recorder sends a **BYE** to the CUBE Media Proxy:

- **Primary recorder as Optional**: CUBE Media Proxy sends a **200OK** response to the primary recorder. The CUBE Media Proxy does not send **BYE** to CUCM. The call-leg between CUCM and the CUBE Media Proxy enters midcall-signaling block mode to reduce interoperability issues that arise to consuming mid-call RE-INVITES. The call-leg between the CUBE Media Proxy and the primary recorder remains in a dormant state. Hence the primary recorder stops recording. However, the remaining recorders continue with the recording.

- **Primary recorder as Mandatory**: CUBE Media Proxy sends **BYE** from the mandatory recorder to CUCM. All the recorders disconnect and the recording stops.

**Secure Recording of Secure Calls**

The CUBE Media Proxy solution supports secure recording of secure calls. Following are the events that are involved in forking of a secure recording request.

1. The CUBE Media Proxy receives a secure recording request from CUCM over TLS transport through **SIP INVITE** with SDP body containing SRTP crypto attributes.

2. The CUBE Media Proxy enters the SRTP-SRTP passthrough mode and sends the secure recording request to the primary recorder through **SIP INVITE** with SDP containing crypto attributes.

3. It is not required to explicitly configure the SRTP on the dial-peers toward the recorders. TLS configuration on the dial-peers from CUBE Media Proxy to the recorders must be done. For information on how to configure TLS on dial-peers to the recorders, see **Configuring SIP TLS**.
TLS Trustpoints must be configured. For information on how to configure TLS Trustpoints, see Configuring TLS Trustpoints.

4. The CUBE Media Proxy offers all the crypto suites that are received from CUCM to the primary recorder. The crypto suite that is negotiated with the primary recorder is offered to the remaining recorders.

5. SRTP media packets are successfully forked to all the recorders.

CUBE Media Proxy supports SRTP media pass-through for SRTP. Hence the CUBE Media Proxy never rejects the SRTP media packets. TCP TLS is required for signaling.

Support for High Availability (HA)

High Availability (HA) feature ensures that the active recording sessions are preserved whenever the CUBE Media Proxy router experiences an outage. Stateful Switchover (SSO) preserves the recording sessions and post-switchover teardown of recording sessions. The CUBE Media Proxy recording sessions continue to be active even after the Active CUBE Media Proxy box goes down (provided a redundant standby box is present). The Standby CUBE Media Proxy box becomes active and service new recording requests.

CUBE Media Proxy does not support ISSU (upgrade) from legacy CUBE versions.

In a recording session, the call-leg with primary recorder and forked call-leg with the remaining recorders is checkpointed in the Standby CUBE Media Proxy router. If the Active device goes down, the Standby device ensures that the forking call is active and is able to exchange further transactions between the recorders and the CUCM. There is no change in the number of call-legs on the Active box and Standby box. The Active and Standby devices must have the same configurations for checkpointing to happen correctly.

CUBE Media Proxy supports Box-to-Box High Availability and Inbox High Availability with up to five recorders. The support extends to both secure and non secure calls.

The following conditions and restrictions apply to the current implementation of SSO.

- Checkpointing is not done on the call-leg that is in the middle of a transaction.
- HA is supported for all mid-call transactions that include REINVITE, UPDATE, or BYE from CUCM. Checkpointing is done after the completion of mid-call transactions. Checkpointing is not done if the SSO happens when the mid-call transactions are in progress.
- Checkpointing is done only for the recorders that were successfully connected before the SSO.
- When one of the optional recorders sends BYE, CUBE Media Proxy sends an Info message to CUCM with the updated status of all the recorders. Checkpointing for the recorder status happens only after receiving a 200OK response from CUCM for the Info message. Checkpointing is not done if the SSO
happens after receiving the **BYE** from the recorder and before receiving a **200 OK** response for the Info Message from CUCM.

- **BYE** received from any of the recorders, is checkpointed.

- Info messages that indicate the status change of the recorders are checkpointed after a response is received for the corresponding Info message.

- Metadata, SRTP context, and common Session ID used across all the five recorders are checkpointed.

You can use the following show commands to monitor the recording sessions on the Active and the Standby instances of CUBE Media Proxy.

- **show call active voice compact**
- **show voip rtp connections**
- **show voip recmsp session**

## Media Latch

By default, the CUBE Media Proxy does source address validation to check if the IP address and port details that are received in the UDP header of the RTP or SRTP packets, matches with the details in the SDP sent by the SIP User Agent. The packets without matching IP address and port are dropped.

In a typical SCCP-based BiB recording using CUBE Media Proxy recording solution, CUCM first sends an SDP with the IP address and a dummy port to the CUBE Media Proxy to get the capabilities of CUBE Media Proxy. CUCM then sends this SDP to the SCCP phone. The CUBE Media Proxy does not know the vBiB IP address and port details of the SCCP phone. In these call flows, the IP address and port details in the media packets that are sent from vBiB of the SCCP phone to SCCP phone, are different from the IP address and port details in the packets that are sent from CUCM to the CUBE Media Proxy.

Media Latching is enabled in the CUBE Media Proxy by default so that the CUBE Media Proxy learns the remote IP address and port details from the UDP transport header of the first RTP or SRTP packet. Media latching is turned on for every call flowing through the CUBE Media Proxy, and works for initial and mid-call scenarios. Media Latching will be always turned on in the inbound leg (CUCM leg), such that the media packets are accepted even if they are sent from a source IP address and port that is different from the IP address that is advertised in the SDP.
PART X

SIP Header Manipulation

• Passing Headers Unsupported by CUBE, on page 557
• Copying SIP Headers, on page 559
• Manipulating SIP Status-Line Header of SIP Responses, on page 567
CHAPTER 41

Passing Headers Unsupported by CUBE

This feature is used to pass parameters that are unsupported by CUBE, but mandatory to the service provider from one leg to another. When a SIP message is received, a check is done for the header, and if it is available, it is copied into a copy list and passed on to the outbound dial peer leg.

- Feature Information for Copying with SIP Profiles, on page 557
- Example: Passing a Header Not Supported by CUBE, on page 557

Feature Information for Copying with SIP Profiles

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 63: Feature Information for Copying with SIP Profiles

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for conditional header manipulation of SIP headers</td>
<td>15.1(3)T</td>
<td>This feature allows users to copy content from one header to the another. This is done by copying the content of messages into variables which can then be used to modify other SIP headers.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE Release 3.6S</td>
<td>This feature modifies the following commands: voice class sip-profiles, response, request, voice-class sip copy-list, sip-header</td>
</tr>
</tbody>
</table>

Example: Passing a Header Not Supported by CUBE

CUBE does not pass “x-cisco-tip”. However, certain TelePresence equipments require “TIP”. The SIP profile below will look for "x-cisco-tip" in the inbound contact header then pass it in the outbound contact header.
Inbound Contact Header
Contact: <sip:89016442998@161.44.77.193;transport=udp>;x-cisco-tip

Outbound Contact Header
Contact: <sip:89016442998@10.86.176.19:5060>;x-cisco-tip

Create a copylist to pass the Contact Header from the incoming message to the outgoing message. The “x-cisco-tip” is not copied in this step as it is unsupported by CUBE.

!Create a copyList
Device(config)# voice class sip-copylist 1
Device(config-class)# sip-header Contact
Device(config-class)# exit

!Apply the copylist to incoming dial peer.
Device(config)# dial-peer voice 1 voip
Device(config-dial-peer)# description incoming SIP Trunk
Device(config-dial-peer)# incoming called-number
Device(config-dial-peer)# voice-class sip copy-list 1

Create a SIP profile that copies “x-cisco-tip” into a variable, and use that variable to modify the outgoing Contact header. Apply the SIP profile to an outbound dial peer.

Device# voice class sip-profiles 3001

!Copy the Contact header from the incoming dial peer into variable u01
Device(config-class)# request INVITE peer-header sip Contact copy "(;x-cisco-tip)" u01

!Modify the outgoing SIP Invite with this variable.
Device(config-class)# request INVITE sip-header Contact modify "$" \u01""

!Apply the SIP Profile to the outgoing dial peer.
Device(config)# dial-peer voice 5000 voip
Device(config-dial-peer)# description outbound SIP
Device(config-dial-peer)# destination-pattern 5...$
Device(config-dial-peer)# voice-class sip profiles 3001
CHAPTER 42

Copying SIP Headers

This feature shows you how outgoing SIP headers can be manipulated using information from incoming and other outgoing SIP headers.

- Feature Information for Copying with SIP Profiles, on page 559
- How to Copy SIP Header Fields to Another, on page 560
- Example: Copying the To Header into the SIP-Req-URI, on page 563

Feature Information for Copying with SIP Profiles

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 64: Feature Information for Copying with SIP Profiles

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for conditional header manipulation of SIP headers</td>
<td>15.1(3)T Cisco IOS XE Release 3.6S</td>
<td>This feature allows users to copy content from one header to the another. This is done by copying the content of messages into variables which can then be used to modify other SIP headers. This feature modifies the following commands: voice class sip-profiles, response, request, voice-class sip copy-list, sip-header</td>
</tr>
</tbody>
</table>
How to Copy SIP Header Fields to Another

Copying From an Incoming Header and Modifying an Outgoing Header

To copy content from an incoming header that a device receives to an outgoing header, configure a SIP copylist for that header and apply it to an incoming dial peer. A SIP profile is configured to copy this incoming header to a user-defined variable and apply it to an outgoing header.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice class sip-copylist tag
4. Do one of the following:
   - sip-header header-name
   - sip-header SIP-Req-URI
5. exit
6. dial-peer voice inbound-dial-peer-tag voip
7. voice class sip-copylist tag
8. exit
9. voice class sip-profiles profile-id
10. {request | response} message peer-header sip header-to-copy copy header-value-to-match copy-variable
11. {request | response} message {sip-header | sdp-header} header-to-modify modify header-value-to-match header-value-to-replace
12. exit
13. dial-peer voice inbound-dial-peer-tag voip
14. voice class sip-copylist tag
15. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong> voice class sip-copylist tag</td>
<td>Configures a list of entities to be sent to a peer call leg and enters voice class configuration mode.</td>
</tr>
</tbody>
</table>

Example:

Device(config)# voice class sip-copylist 100
### SIP Header Manipulation

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 4</strong>&lt;br&gt;Do one of the following:&lt;br&gt;• sip-header header-name&lt;br&gt;• sip-header SIP-Req-URI&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-class)# sip-header To</td>
<td>Specifies the SIP header to be copied to the peer call leg. <strong>sip-req-uri</strong>—Configures Cisco Unified Border Element (UBE) to send a SIP request Uniform Resource Identifier (URI) to the peer call leg.&lt;br&gt;<strong>header-name</strong>—Configures Cisco Unified Border Element (UBE) to send the header name specified to the peer call leg.</td>
</tr>
<tr>
<td><strong>Step 5</strong>&lt;br&gt;<strong>exit</strong></td>
<td>Exits voice class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 6</strong>&lt;br&gt;dial-peer voice inbound-dial-peer-tag voip&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config)# dial-peer voice 2 voip</td>
<td>Enters the dial peer configuration mode for the specified inbound dial peer.</td>
</tr>
<tr>
<td><strong>Step 7</strong>&lt;br&gt;voice class sip-copylist <strong>tag</strong> &lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-dial-peer)# voice class sip-copylist 100</td>
<td>Applies the copy list to the dial-peer.</td>
</tr>
<tr>
<td><strong>Step 8</strong>&lt;br&gt;<strong>exit</strong></td>
<td>Exits to global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 9</strong>&lt;br&gt;voice class sip-profiles <strong>profile-id</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config)# voice class sip-profiles 10</td>
<td>Creates a SIP Profiles and enters voice class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 10</strong>&lt;br&gt;{request</td>
<td>response} message peer-header sip&lt;br&gt;header-to-copy copy header-value-to-match copy-variable&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-class)# request INVITE peer-header sip TO copy &quot;sip:(.*)@&quot; u01</td>
</tr>
<tr>
<td><strong>Step 11</strong>&lt;br&gt;{request</td>
<td>response} message {sip-header</td>
</tr>
<tr>
<td><strong>Step 12</strong>&lt;br&gt;<strong>exit</strong></td>
<td>Exits to global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 13</strong>&lt;br&gt;dial-peer voice inbound-dial-peer-tag voip&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config)# dial-peer voice 2 voip</td>
<td>Enters the dial peer configuration mode for the specified inbound dial peer.</td>
</tr>
</tbody>
</table>
Copying From One Outgoing Header to Another

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice class sip-profiles *profile-id*
4. `{request | response} message {sip-header | sdp-header} header-to-copy copy header-value-to-match copy-variable`
5. `{request | response} message {sip-header | sdp-header} header-to-modify modify header-value-to-match header-value-to-replace`
6. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  • Enter your password if prompted.  
| **Step 2** configure terminal | Enters global configuration mode. |
| **Step 3** voice class sip-profiles *profile-id* | Creates a SIP profile and enters voice class configuration mode. |
| Example: Device(config)# voice class sip-profiles 10 | |
| **Step 4** `{request | response} message {sip-header | sdp-header} header-to-copy copy header-value-to-match copy-variable` | Copies the contents of the specified header from an outbound message into a copy variable. |
| Example: Device(config-class)# request INVITE sip-header TO copy "sip:(.*)@" u01 | |
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 5 `{request</td>
<td>response} message {sip-header</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# request INVITE sip-header SIP-Req-URI modify &quot;.<em>@(.</em>)*&quot; &quot;INVITE sip:&lt;u01@\1&quot;</td>
<td></td>
</tr>
<tr>
<td>Step 6 end</td>
<td>Exits voice class configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# end</td>
<td></td>
</tr>
</tbody>
</table>

### What to do next

Apply the SIP Profile to an outbound dial peer.

### Example: Copying the To Header into the SIP-Req-URI

#### Copying Contents from One Header to Another

Given below is a scenario in an organization, where the provider has sent only a global reference number in the SIP-Req-URI header of the INVITE message, and has placed the actual phone destination number only in the To: SIP header. The CUCM typically routes on the SIP-Req-URI.

Given below is the original SIP message, where the INVITE has a non-routable value of 43565432A5. The actual phone destination number is 25555552 and is present in the To: SIP header.

*Figure 46: Incoming SIP Message*

Given below is the SIP message that is required. Note that 43565432A5 has changed to 25555552 in the SIP INVITE.
Because CUBE is a back-to-back user agent, the incoming dial peer is matched to the outgoing dial peer. The SIP Profile configured below copies the value from the incoming dial peer

```
Device# voice class sip-profiles 1

!Copy the To header from the incoming dial peer into variable u01
Device(config-class)# request INVITE peer-header sip TO copy "sip:(.*)@" u01

!Modify the outgoing SIP Invite with this variable.
Device(config-class)# request INVITE sip-header SIP-Req-URI modify ".*@(.*)" "INVITE sip:\u01@\1"

Apply the SIP profile to the incoming dial peer.

Device(config)# dial-peer voice 99 voip
Device(config-dial-peer)# outgoing to CUCM
Device(config-dial-peer)# destination-pattern 02555555.
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.1.2.3

!Applying SIP profile to the dial peer
Device(config-dial-peer)# voice-class sip profiles 1
Device(config-dial-peer)# voice-class code 1
Device(config-dial-peer)# dtmf-relay rtp-nte
Device(config-dial-peer)# no vad

Additionally, if you would like to copy the To: Header from the inbound dial peer to the outbound dial peer, use a copy list.

!Create a copy List
Device(config)# voice class sip-copylist 1
Device(config-class)# sip-header TO
Device(config-class)# exit

!Apply the copy list to incoming dial peer.
Device(config)# dial-peer voice 1 voip
Device(config-dial-peer)# description incoming SIP Trunk
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target sip-server
Device(config-dial-peer)# incoming uri to TRUNK
Device(config-dial-peer)# voice-class code 1
Device(config-dial-peer)# voice-class sip copy-list 1

Device(config)# voice class uri TRUNK sip
Device(config-class)# user-id 2555555.
Device(config-class)# end
```
Example: Copying the To Header into the SIP-Req-URI
Example: Copying the To Header into the SIP-Req-URI
CHAPTER 43

Manipulating SIP Status-Line Header of SIP Responses

The SIP status line is a SIP response header, and it can be modified like any other SIP headers of a message. It can either be modified with a user-defined value, or the status line from an incoming response can be copied to an outgoing SIP response. The SIP header keyword used for the response status line is *SIP-StatusLine*.

- Feature Information for Manipulating SIP Responses, on page 567
- Copying Incoming SIP Response Status Line to Outgoing SIP Response, on page 568
- Modifying Status-Line Header of Outgoing SIP Response with User Defined Values, on page 571

Feature Information for Manipulating SIP Responses

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

**Table 65: Feature Information for Manipulating SIP Responses**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| SIP Profile Enhancements for SIP responses and error codes | 15.4(1)T Cisco IOS XE Release 3.12S | This feature extends SIP profiles to allow the following:  
- Modification of the outgoing SIP response status line. Previously, only modification of outgoing SIP requests and responses was possible.  
- Copying of the incoming SIP response status-line. The information from the peer-leg status-line can then be copied to user-variables and applied to the outbound response status-line. This option can be used to pass-thru the error-code and error phrase from peer-leg. Previously, only copying of SIP headers were possible.  
- Before applying a SIP profile to a response from CUBE, the response can be mapped to its corresponding request. |
Copy Incomming SIP Response Status Line to Outgoing SIP Response

To copy content from the status line of an incoming SIP response that a device receives to an outgoing response, configure a SIP copylist for SIP status line and apply it to an incoming dial peer. A SIP profile must be configured to copy the status line of an incoming SIP response to a user-defined variable and apply it to an outgoing SIP response.

**Figure 48: Call Flow for Copying the Status Line from the Incoming SIP Response to the Outgoing SIP Response**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice class sip-copylist *tag*
4. sip-header SIP-StatusLine
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>voice class sip-copylist tag</td>
<td>Configures a list of entities to be sent to the peer call leg and enters voice class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# voice class sip-copylist 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>sip-header SIP-StatusLine</td>
<td>Specifies that the Session Initiation Protocol (SIP) status line header must be sent to the peer call leg.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-class)# sip-header SIP-StatusLine</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>exit</td>
<td>Exits voice class configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-class)# exit</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>dial-peer voice inbound-dial-peer-id voip</td>
<td>Specifies an inbound dial peer and enters dial peer configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# dial-peer voice 99 voip</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>voice-class sip copy-list list-id</td>
<td>Associates the SIP copy list with the inbound dial peer.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-dial-peer)# voice-class sip copy-list 1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>exit</td>
<td>Exits dial peer configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 9</strong></td>
<td>Enables dial peer-based VoIP SIP profile configurations and enters voice class configuration mode.</td>
</tr>
<tr>
<td><strong>voice class sip-profiles</strong> tag</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# voice class sip-profiles 10</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>Copies responses from the corresponding incoming call leg into a copy variable.</td>
</tr>
<tr>
<td><strong>response</strong> response-code peer-header sip SIP-StatusLine copy match-pattern copy-variable</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-class)# response ANY peer-header sip SIP-StatusLine copy &quot;(.*)&quot; u01</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>Modifies an outgoing response using the copy variable defined in the previous step.</td>
</tr>
<tr>
<td><strong>response</strong> response-code sip-header SIP-StatusLine modify match-pattern copy-variable</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-class)# response ANY sip-header SIP-StatusLine modify &quot;.*&quot; &quot;\u01&quot;</td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td>Exits voice class configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td><strong>exit</strong></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-class)# exit</td>
</tr>
</tbody>
</table>

### What to do next

Apply the SIP profile to the outbound dial peer to copy the SIP response to the outbound leg.
Modifying Status-Line Header of Outgoing SIP Response with User Defined Values

Figure 49: Call Flow Configuring a New Status Line for an Outgoing SIP Response Based on an Incoming SIP Request

**SUMMARY STEPS**

1. **enable**
2. **configure terminal**
3. **voice class sip-profiles tag**
5. **exit**

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: enable</td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: configure terminal</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enables dial peer-based VoIP SIP profile configurations and enters voice class configuration mode.</td>
</tr>
<tr>
<td>voice class sip-profiles <em>tag</em></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice class sip-profiles 10</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Modifies SIP status line of a SIP response with user-defined values.</td>
</tr>
<tr>
<td>response <em>response-code</em> [@method method-type] sip-header SIP-StatusLine modify <em>match-pattern</em> <em>replacement-pattern</em></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Modifying status line of a SIP header to a user-defined response type:</td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# response 404 sip-header SIP-StatusLine modify &quot;404 Not Found&quot; &quot;404 MyError&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Exits voice class configuration mode.</td>
</tr>
<tr>
<td>exit</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# exit</td>
<td></td>
</tr>
</tbody>
</table>

**What to do next**

Associate the SIP profile with an outbound dial peer.
PART XI

Payload Type Interoperability

• Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls, on page 575
Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls

The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature provides dynamic payload type interworking for dual tone multifrequency (DTMF) and codec packets for Session Initiation Protocol (SIP) to SIP calls.

Based on this feature, the Cisco Unified Border Element (Cisco UBE) interworks between different dynamic payload type values across the call legs for the same codec. Also, Cisco UBE supports any payload type value for audio, video, named signaling events (NSEs), and named telephone events (NTEs) in the dynamic payload type range 96 to 127.

- Feature Information for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls, on page 575
- Restrictions for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls, on page 576
- Symmetric and Asymmetric Calls, on page 576
- High Availability Checkpointing Support for Asymmetric Payload, on page 577
- How to Configure Dynamic Payload Type Passthrough for DTMF and Codec Packets for SIP-to-SIP Calls, on page 578
- Configuration Examples for Assymetric Payload Interworking, on page 581

Feature Information for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 66: Feature Information for Dynamic Payload Interworking for DTMF and Codec Packets Support

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls</td>
<td>15.0(1)XA 15.1(1)T</td>
<td>The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature provides dynamic payload type interworking for DTMF and codec packets for SIP-to-SIP calls. The following commands were introduced or modified: asymmetric payload and voice-class sip asymmetric payload.</td>
</tr>
<tr>
<td>Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls</td>
<td>Cisco IOS Release XE 3.1S</td>
<td>The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature provides dynamic payload type interworking for DTMF and codec packets for SIP-to-SIP calls. The following commands were introduced or modified: asymmetric payload and voice-class sip asymmetric payload.</td>
</tr>
</tbody>
</table>

Restrictions for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls

The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature is not supported for the following:

- H323-to-H323 and H323-to-SIP calls.
- All transcoded calls.
- Secure Real-Time Protocol (SRTP) pass-through calls.
- Flow-around calls.
- Asymmetric payload types are not supported on early-offer (EO) call legs in a delayed-offer to early-offer (DO-EO) scenario.
- Cisco fax relay.
- Multiple \( m \) lines with the same dynamic payload types, where \( m \) is:

\[
m = \text{audio} <\text{media-port1}> \text{ RTP/AVP } XXX \ m = \text{video} <\text{media-port2}> \text{ RTP/AVP } XXX
\]

Symmetric and Asymmetric Calls

Cisco UBE supports dynamic payload type negotiation and interworking for all symmetric and asymmetric payload type combinations. A call leg on Cisco UBE is considered as symmetric or asymmetric based on the payload type value exchanged during the offer and answer with the endpoint:
• A symmetric endpoint accepts and sends the same payload type.
• An asymmetric endpoint can accept and send different payload types.

The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature is enabled by default for a symmetric call. An offer is sent with a payload type based on the dial-peer configuration. The answer is sent with the same payload type as was received in the incoming offer. When the payload type values negotiated during the signaling are different, the Cisco UBE changes the Real-Time Transport Protocol (RTP) payload value in the VoIP to RTP media path.

To support asymmetric call legs, you must enable The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature. The dynamic payload type value is passed across the call legs, and the RTP payload type interworking is not required. The RTP payload type handling is dependent on the endpoint receiving them.

**High Availability Checkpointing Support for Asymmetric Payload**

High availability for a call involving asymmetric payloads is supported. In case of fail-over from active to stand-by, the asymmetric payload interworking will be continued as new active CUBE passes across the payload type values according to the negotiation and call establishment.

*Figure 50: Sample High-Availability Topology*
How to Configure Dynamic Payload Type Passthrough for DTMF and Codec Packets for SIP-to-SIP Calls

Configuring Dynamic Payload Type Passthrough at the Global Level

Perform this task to configure the pass through of DTMF or codec payload to the other call leg (instead of performing dynamic payload type interworking) feature at the global level.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. sip
5. asymmetric payload {dtmf | dynamic-codecs | full | system}
6. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service configuration mode.</td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td>Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> asymmetric payload {dtmf</td>
<td>dynamic-codecs</td>
</tr>
</tbody>
</table>
Configuring Dynamic Payload Type Passthrough for a Dial Peer

Perform this task to configure the pass through of DTMF or codec payload to the other call leg (instead of performing dynamic payload type interworking) feature at the dial-peer level.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice  tag  voip
4. voice-class sip asymmetric payload  {dtmf | dynamic-codecs | full | system}
5. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice  tag  voip</td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# dial-peer voice 77 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class sip asymmetric payload  {dtmf</td>
<td>dynamic-codecs</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>

**Note** The dtmf and dynamic-codecs keywords are internally mapped to the full keyword to provide asymmetric payload type support for audio and video codecs, DTMF, and NSEs.
Verifying Dynamic Payload Interworking for DTMF and Codec Packets Support

This task shows how to display information to verify Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls configuration feature. These `show` commands need not be entered in any specific order.

**SUMMARY STEPS**

1. `enable`
2. `show call active voice compact`
3. `show call active voice`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config-dial-peer)# voice-class sip asymmetric payload full</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>(Optional) Exits dial peer voice configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

**Troubleshooting Tips**

Use the following commands to debug any errors that you may encounter when you configure the Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature:

- `debug ccsip all`
- `debug voip ccapi inout`
Configuration Examples for Assymetric Payload Interworking

Example: Asymmetric Payload Interworking—Passthrough Configuration

```plaintext
! voice service voip
   allow-connections sip to sip
   sip
      rel1xx disable
      asymmetric payload full
      midcall-signaling passthru
!
! dial-peer voice 1 voip
   voice-class sip asymmetric payload full
   session protocol sipv2
   rtp payload-type cisco-codec-fax-ind 110
   rtp payload-type cisco-codec-video-h264 112
   session target ipv4:9.13.8.23
!
```
In the above example, it is assumed that 110 and 112 are not used for any other payload.

Example: Asymmetric Payload Interworking—Interworking Configuration

! voice service voip
 allow-connections sip to sip
!
dial-peer voice 1 voip
 session protocol sipv2
 rtp payload-type cisco-codec-fax-ind 110
 rtp payload-type cisco-codec-video-h264 112
 session target ipv4:9.13.8.23
!

In the above example, it is assumed that 110 and 112 are not used for any other payload.
PART XII

Protocol Interworking

• Delayed-Offer to Early-Offer, on page 585
• H323-to-SIP Interworking on CUBE, on page 595
• H.323-to-H.323 Interworking on CUBE, on page 601
• SIP RFC 2782 Compliance with DNS SRV Queries, on page 615
CHAPTER 45

Delayed-Offer to Early-Offer

The Delayed-Offer to Early-Offer (DO-EO) feature allows CUBE to convert a delayed offer that it receives into an early offer. This feature is supported in the Media Flow-Around mode.

This feature also supports high-density transcoding calls, where transcoding IP addresses and port numbers are exchanged between the sender and receiver. This feature also supports midcall renegotiation of codecs required if an exchange of parameters that is not end-to-end causes an inefficient media flow.

- Feature Information for Delayed-Offer to Early-Offer, on page 585
- Prerequisites for Delayed-Offer to Early-Offer, on page 586
- Restrictions for Delayed-Offer to Early-Offer Media Flow-Around, on page 586
- Delayed-Offer to Early-Offer in Media Flow-Around Calls, on page 586
- MidCall Renegotiation Support for Delayed-Offer to Early-Offer Calls, on page 590
- High-Density Transcoding Calls in Delayed-Offer to Early-Offer, on page 592

Feature Information for Delayed-Offer to Early-Offer

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delayed-Offer to Early-Offer</td>
<td>Cisco IOS 12.4(3)</td>
<td>The Delayed-Offer to Early-Offer feature allows CUBE to convert a delayed offer it receives into an early offer. The following commands were introduced by this feature: voice-class sip early-offer forced, early-offer forced and media transcoder high-density.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 12.4(24)T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 15.0(1)M</td>
<td></td>
</tr>
<tr>
<td>Delayed-Offer to Early-Offer Support for Video Calls</td>
<td>Cisco IOS 12.4(22)T</td>
<td>The Delayed-Offer to Early-Offer support was extended for video calls. The following command was introduced: codec-profile</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Media Flow-Around with SIP Signaling control on CUBE</td>
<td>Cisco IOS 15.1(3)T</td>
<td>Support for Media Flow-Around for Delayed-Offer to Early-Offer audio calls on CUBE was introduced. No new commands were introduced or modified.</td>
</tr>
<tr>
<td>Midcall Renegotiation Support for DO-EO Calls</td>
<td>Cisco IOS 15.4(2)T, Cisco IOS XE 3.12S</td>
<td>The Midcall renegotiation of codecs feature configures the midcall renegotiation of codecs, if an exchange of parameters that is not end-to-end causes an inefficient media flow. The following commands were modified by this feature: voice-class sip early-offer forced renegotiate [always], early-offer forced renegotiate [always].</td>
</tr>
</tbody>
</table>

### Prerequisites for Delayed-Offer to Early-Offer

Configure delayed-offer to early-offer in media flow-around mode.

### Restrictions for Delayed-OffertoEarly-OffermMedia Flow-Around

- CUBE does not support change in IP address or port number in the locally triggered RE-INVITE response.
- CUBE does not support DE-EO Media Flow-Around for video calls.

### Delayed-Offer to Early-Offer in Media Flow-Around Calls

Delayed-Offer to Early-Offer (DO-EO) allows CUBE to convert a delayed offer (DO) into an early offer (EO) in the media flow-around mode.

CUBE sends its local IP address in the initial EO INVITE Session Description Protocol (SDP) message. In the image, this is illustrated by INVITE (SDP1, CUIP1). Later, an additional RE-INVITE is locally generated by CUBE to communicate the SDP message details from the sender. This is illustrated by RE-INVITE (SDP5, IP2) in the below image. The RE-INVITE response is consumed by CUBE and not communicated to the sender.

*Figure 51: Delayed Offer to Early Offer in Media Flow-Around Calls*

CUBE supports delayed offer to early offer for SIP-to-SIP video calls. CUBE generates an outgoing Early Offer INVITE with the configured codec list, for a incoming Delayed Offer INVITE.
DO-EO video call is supported if both audio and video codecs are configured under a dial peer. codec profile command defines the codec attributes for Video (H263, H264) and Audio (AACLD) codecs. The codec attributes configured under codec-profile is used to generate the a=fmtp attribute line in the Early Offer SDP.

**Configuring Delayed Offer to Early Offer**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. Configure conversion of a delayed offer to an early offer:
   - In dial-peer configuration mode
     ```
     voice-class sip early-offer forced
     ```
   - In global VoIP SIP configuration mode
     ```
     early-offer forced
     ```
4. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>Configure conversion of a delayed offer to an early offer:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>voice-class sip early-offer forced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• In global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>early-offer forced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In dial-peer configuration mode:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device (config) dial-peer voice 10 voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device (config-dial-peer) voice-class sip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>early-offer forced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device (config-dial-peer) end</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring Delayed Offer to Early Offer for Video Calls

#### SUMMARY STEPS

1. enable
2. configure terminal
3. codec profile <tag profile>
4. dial-peer voice number <number> voip
5. codec codec profile
6. video codec codec profile
7. voice-class sip early-offer forced
8. end

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> codec profile &lt;tag profile&gt;</td>
<td>Configures the audio and video codec profiles.</td>
</tr>
<tr>
<td>Example: codec profile 1 aacld codec profile 2 H264</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> dial-peer voice number &lt;number&gt; voip</td>
<td>Enters dial peer configuration mode for the specified VoIP dial peer.</td>
</tr>
<tr>
<td>Example: Device(config)# dial-peer voice 1 voip</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>5</td>
<td>codec codec profile</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# profile 1 aacld</td>
</tr>
<tr>
<td>6</td>
<td>video codec codec profile</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# video codec h264 profile 2</td>
</tr>
<tr>
<td>7</td>
<td>voice-class sip early-offer forced</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device (config-dial-peer)# voice-class sip early-offer forced</td>
</tr>
<tr>
<td>8</td>
<td>end</td>
</tr>
</tbody>
</table>

### Configuring Delayed Offer to Early Offer Medial Flow-Around

#### SUMMARY STEPS

1. enable
2. configure terminal
3. medial flow-around
4. Configure conversion of a delayed offer to an early offer:
   - In dial-peer configuration mode
     ```
     voice-class sip early-offer forced
     ```
   - In global VoIP SIP configuration mode
     ```
     early-offer forced
     ```
5. end

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>enable</td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td>Enables media flow-around.</td>
<td></td>
</tr>
</tbody>
</table>

**Step 3**

**medial flow-around**

**Example:**

```
Device(config-voi-serv)# media flow-around
```

**Step 4**

Configure conversion of a delayed offer to an early offer:
- In dial-peer configuration mode
  ```
  voice-class sip early-offer forced
  ```
- In global VoIP SIP configuration mode
  ```
  early-offer forced
  ```

**Example:**

In dial-peer configuration mode:

```
Device (config) dial-peer voice 10 voip
Device (config-dial-peer) voice-class sip
early-offer forced
Device (config-dial-peer) end
```

**Example:**

In global VoIP SIP mode:

```
Device(config)# voice service voip
Device (config-voi-serv) sip
Device (config-voi-sip) early-offer forced
Device (config-voi-sip) end
```

**Step 5**

end

Exits to privileged EXEC mode.

---

**MidCall Renegotiation Support for Delayed-Offer to Early-Offer Calls**

When CUBE converts a delayed offer into an early offer, an incomplete exchange of Format specific parameters (FMTP) occurs during call establishment, resulting in either the noninitiation of media transmission or media transmission in a quality that may not be the best. This is especially a problem in video calls.

To overcome this situation, midcall renegotiation of capabilities can be configured.

The **early-offer forced renegotiate [always]** command is used to configure this in global VoIP configuration mode (config-voi-serv) and the **voice-class sip early-offer forced renegotiate** command is dial-peer configuration mode (config-dial-peer) and voice-class configuration mode (config-class).
The *early-offer forced renegotiate* command triggers a delayed-offer RE-INVITE if the negotiated codecs are one of the following:

- `aaclld`—Audio codec AACLD 90000 bps
- `h263`—Video codec H263
- `h263+`—Video codec H263+
- `h264`—Video codec H264
- `mp4a`—Wideband audio codec

The *early-offer forced renegotiate always* command always triggers a delayed-offer RE-INVITE. This option can be used to support all other codecs.

### Restrictions for MidCall Renegotiation Support for DO-EO Calls

- If `midcall-signaling block` or `midcall-signaling passthru media-change` commands have been configured, the feature does not work because a midcall RE-INVITE is not triggered by CUBE.
- If initial call is transcoded, then midcall re-invite is not triggered by CUBE.

#### Note

For EO to EO calls, the Delayed-Offer midcall RE-INVITE is not triggered by the CUBE, if either `midcall-signaling block` or `midcall-signaling passthru media-change` command is configured.

### Configuring Mid Call Renegotiation Support for Delayed-Offer to Early-Offer Calls

#### SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice *id* voip
4. media transcoder high-density
5. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice id voip</td>
<td>Enters dial-peer configuration mode and configures the selected dial peer.</td>
</tr>
<tr>
<td><strong>Step 4</strong> media transcoder high-density</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config) dial-peer voice 10 voip Device (config-dial-peer) media transcoder high-density Device (config-dial-peer) end</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

High-Density Transcoding Calls in Delayed-Offer to Early-Offer

High-Density Transcoding Calls in the media flow-around DO-to-EO mode is a feature where the transcoding IP address and port number are exchanged between the originating and terminating user agents. For high-density transcoding calls, CUBE is in the media flow-through mode even if media flow-around is configured.

In the figure below, XIP1 is passed to CUCM1 when a 200 OK is received from SBC1. ACK from CUCM1 triggers new RE-INVITE with transcoding IP address and port number (XIP2) and this RE-INVITE has to be locally handled in CUBE.

*Figure 53: High-Density Transcoding Calls in DO-to-EO*
The **media transcoder high-density** command is used to configure this feature in dial-peer configuration mode (config-dial-peer). Refer to “Modes for Configuring Dial Peers” section to enter these modes and configure this feature.

For high-density transcoding calls with a common codec, CUBE should be in Media Flow-Through mode even though media flow-around is configured.

**Figure 54: High-Density Transcoding Calls for Common Codecs in DO-to-EO**

---

**Restrictions for High-Density Transcoding DO-EO Calls**

For high-density transcoding calls with a common codec, CUBE will be in Media Flow-through mode even though Media Flow-Around is configured.

**Configuring High-Density Transcoding**

To configure High-Density Transcoding delayed offer to early offer calls in media flow-around mode, perform the following steps:

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. media transcoder high-density
5. sip
6. early offer-forced
7. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode. Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Configuring High-Density Transcoding

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>voice service voip</td>
<td>Enters voice service configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>media transcoder high-density</td>
<td>Enables media transcoder high-density for transcoding high-density media calls.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# media transcoder high-density</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>sip</td>
<td>Enters SIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>early offer-forced</td>
<td>Forcefully sends SIP EO invites on the Out-Leg.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-sip)# early offer-forced</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Exits the present configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 46

H323-to-SIP Interworking on CUBE

This chapter describes how to configure H.323-to-SIP interworking in CUBE and lists the various features supported in this interworking model.

- Prerequisites, on page 595
- Restrictions, on page 595
- H323-to-SIP Basic Call Interworking, on page 596
- H323-to-SIP Supplementary Features Interworking, on page 598
- H.323-to-SIP Codec Progress Indicator Interworking for Media Cut-Through, on page 599
- Configuring H323-to-SIP Interworking, on page 599

Prerequisites

- Enable CUBE on the device
- Perform basic H.323 gateway configuration. See Configuring H.323 Gateway (Optional)
- Perform basic H.323 gatekeeper configuration. See Configuring H.323 Gatekeeper (Optional)

Restrictions

- Changing codecs during rotary dial peer selection is not supported.
- Voice class codec is not supported.
- Configure extended capabilities on dial peers for fast start-to-early media scenarios.
- Delayed Offer to Slow-Start is not supported for SRTP-to-SRTP H.323-to-SIP calls.
- During a triggered INVITE scenario the Cisco UBE always generates a delayed offer INVITE.
- Fast-start to delayed-media signal interworking is not supported.
- Fast Start to Early Offer Supplementary Service will not work without extended capabilities configured under dial-peer.
- GSMFR and GSMEFR codecs are not supported.
- Media flow-around is not supported.
- Passing multiple diversion headers or multiple contact header in 302 to the H.323 leg is not supported.
- RSVP for supplementary scenarios is not supported.
- Session refresh is not supported.
- SIP-to-H.323 Supplementary Services based on H.450 is not supported.
- Slow-start to early media signal interworking is not supported.
Supplementary services are Empty Capability Set (ECS) based supplementary services from the H.323 perspective, not H.450 supplementary services.

- LTI based transcoding is not supported.
- Transcoding for supplementary calls is not supported.
- SCCP based codec transcoding is not support with an exception of Delayed-Offer to Slow-Start with static codec.
- DTMF interworking rtp-nte to inband is supported only with non-high-density transcoding in a delayed-offer to slow-start call.

**H323-to-SIP Basic Call Interworking**

This feature enables the IP-to-IP gateway to bridge calls between networks that support different VoIP call-signaling protocols (SIP and H.323). The SIP-to-H.323 protocol interworking capabilities of the CUBE support the following:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported Release</th>
<th>Additional Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic voice calls (G.711 and G.729 codecs)</td>
<td>12.4(11)T</td>
<td></td>
</tr>
<tr>
<td>UDP and TCP transport</td>
<td>12.3(11)T</td>
<td>SIP (UDP)&lt;—&gt;H.323 (TCP) SIP (TCP)&lt;—&gt;H.323 (TCP) SIP (UDP)&lt;—&gt;H.323 (UDP) SIP (TCP)&lt;—&gt;H.323 (UDP) Default SIP protocol is UDP. Default H.323 protocol is TCP.</td>
</tr>
<tr>
<td>Interworking between</td>
<td>12.3(11)T</td>
<td>• H.323 Fast Start&lt;—&gt;SIP Early Media</td>
</tr>
<tr>
<td>• H.323 Fast-Start and SIP early-media signaling</td>
<td></td>
<td>• H.323 Slow Start&lt;—&gt;SIP Delay Media</td>
</tr>
<tr>
<td>• H.323 Slow-Start and SIP delayed-media signaling</td>
<td></td>
<td><strong>Note</strong> No other combinations are supported. For example, H.323 Slow Start&lt;—&gt;SIP Early Media is not supported and results in call failure.</td>
</tr>
</tbody>
</table>
### Protocol Interworking

#### H.323-to-SIP Basic Call Interworking

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported Release</th>
<th>Additional Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.323-to-SIP RSVP Support</td>
<td>12.3(11)T</td>
<td>The following cases are supported (acc-qos and reg_qos):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H.323 to H.323 with only one leg having RSVP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H.323 to H.323 with both legs having RSVP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H.323 to SIP with only one leg having RSVP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H.323 to SIP with both legs having RSVP</td>
</tr>
<tr>
<td>DTMF relay interworking:</td>
<td>12.3(11)T</td>
<td>- H.245 alpha/signal&lt;—&gt;SIP Notify</td>
</tr>
<tr>
<td></td>
<td>12.4(6)XE</td>
<td>- H.245 alpha/signal&lt;—&gt;SIP RFC 2833</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H.245 alpha/signal&lt;—&gt;SIP KPML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- G.711 Inband DTMF&lt;—&gt;RFC 2833</td>
</tr>
<tr>
<td>Voice call transcoding support</td>
<td>12.3(11)T</td>
<td>- Only voice and DTMF are supported. (G.711-G.729)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Codec transparent and codec filtering is not supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Cisco Fax Relay and T.38 Fax are not supported</td>
</tr>
<tr>
<td>Calling/called name and number</td>
<td>12.3(11)T</td>
<td>- H.323 IOS FXS/SCCP – IPIPGW – SIP IOS FXS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H.323 IOS FXS/SCCP – IPIPGW – SIP CCME Skinny Phone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H.323 IOS FXS/SCCP – IPIPGW – SIP IP Phone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CCM Phone – IPIPGW – SIP CCME Skinny Phone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CCM Phone – IPIPGW – SIP IP Phone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SIP IOS FXS – IPIPGW – H.323 IOS FXS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SIP IOS FXS – IPIPGW – H.323 CCME Skinny Phone</td>
</tr>
</tbody>
</table>
## H323-to-SIP Supplementary Features Interworking

This interworking provides enhanced termination and re-origination of signaling and media between VoIP and Video Networks in conformance with RFC3261.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported Release</th>
<th>Additional Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADIUS call-accounting records</td>
<td>12.3(11)T</td>
<td>H.323&lt;-&gt;SIP Radius call accounting</td>
</tr>
<tr>
<td>TCL IVR 2.0 for SIP, including media playout and digit collection</td>
<td>12.3(11)T</td>
<td>12.3(11)T — TCL IVR 2.0 for SIP, including media playout and digit collection (RFC 2833 DTMF relay)</td>
</tr>
<tr>
<td></td>
<td>12.4(11)T</td>
<td>12.4(11)T — TCL IVR support with SIP NOTIFY DTMF</td>
</tr>
<tr>
<td>SRTP Passthrough</td>
<td>12.4(15)XY</td>
<td></td>
</tr>
<tr>
<td>Supplementary Services (ECS based)</td>
<td>12.4(11)XJ2</td>
<td></td>
</tr>
<tr>
<td>Codec Transparent</td>
<td>12.4(11)T</td>
<td></td>
</tr>
<tr>
<td>Extended codec support and codec filtering</td>
<td>12.4(11)T</td>
<td></td>
</tr>
</tbody>
</table>

### H.323-to-SIP Supplementary Services Interworking

<table>
<thead>
<tr>
<th>Feature</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support H.323-to-SIP Supplementary services for CUCM with MTP on the H.323 Trunk.</td>
<td>12.3(11)T</td>
</tr>
<tr>
<td>ILBC Codec Support</td>
<td>12.3(11)T</td>
</tr>
<tr>
<td>Interworking between G.711 inband DTMF to RFC2833</td>
<td>12.3(11)T</td>
</tr>
<tr>
<td>VXML 3.x support</td>
<td>12.3(11)T</td>
</tr>
<tr>
<td>SIP CDRs and H.323 CDRs Mapping</td>
<td>12.3(11)T</td>
</tr>
<tr>
<td>Conference ID can be used to correlate H.323 and SIP Radius records. Conference ID is unique on both H.323 and SIP legs</td>
<td>12.3(11)T</td>
</tr>
<tr>
<td>VXML support with SIP Notify</td>
<td>12.4(11)T</td>
</tr>
<tr>
<td>Mapping ECS to ReINVITE and ECS to REFER on the Cisco CUBE</td>
<td>12.4(20)T</td>
</tr>
</tbody>
</table>
H.323-to-SIP Codec Progress Indicator Interworking for Media Cut-Through

OGW is the originating gateway and TGW is the terminating gateway.

Table 68: SIP(OGW) → IP/IPGW → H323(TGW) calls

<table>
<thead>
<tr>
<th>SIP at In Leg</th>
<th>H.323 at Out Leg</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>183 Session Progress</td>
<td>Progress/Alert PI = 8</td>
<td>Analog phone at TGW</td>
</tr>
<tr>
<td>180 Ring</td>
<td>Alert with PI = 0</td>
<td>SCCP phone at TGW</td>
</tr>
</tbody>
</table>

Table 69: H323(OGW) → IP/IPGW → SIP(TGW) calls

<table>
<thead>
<tr>
<th>H.323 at In Leg</th>
<th>SIP at Out Leg</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progress/Alert PI = 8</td>
<td>183 Session Progress</td>
<td>Analog phone at TGW</td>
</tr>
<tr>
<td>Alert with PI = 0</td>
<td>180 Ring</td>
<td>SIP/SCCP phone at TGW</td>
</tr>
</tbody>
</table>

Configuring H323-to-SIP Interworking

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. allow-connections h323 to sip
5. allow-connections sip to h323
6. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable Example:</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

Example:

`Router> enable`
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>voice service voip</td>
<td>Enters Global VoIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>allow-connections h323 to sip</td>
<td>Allows connections from a h323 endpoint to a SIP endpoint.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>allow-connections sip to h323</td>
<td>Allows connections from a SIP endpoint to a H.323 endpoint.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Exits to privilged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-voi-serv)# end</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 47

H.323-to-H.323 Interworking on CUBE

This chapter describes how to configure and enable features for H.323-to-H.323 connections on CUBE.

Configuring H.323-to-H.323 connections on a CUBE opens all ports by default. If CUBE has a public IP address and a PSTN connection, CUBE becomes vulnerable to malicious attackers who can execute toll fraud across the gateway. To eliminate this threat, you can bind an interface to a private IP address that is inaccessible to untrusted hosts. In addition, you can protect any public or untrusted interface by configuring a firewall or an access control list (ACL) to prevent unwanted traffic from traversing the router.

- Feature Information for H.323-to-H.323 Interworking, on page 601
- Prerequisites, on page 602
- Restrictions, on page 602
- Slow Start to Fast-Start Interworking, on page 602
- Call Failure Recovery (Rotary), on page 604
- Managing H.323 IP Group Call Capacities, on page 605
- Overlap Signaling, on page 610
- Verifying H.323-to-H.323 Interworking, on page 611
- Troubleshooting H.323-to-H.323 Interworking, on page 613

Feature Information for H.323-to-H.323 Interworking

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing H.323 IP Group Call Capacities</td>
<td>12.2(13)T</td>
<td>Creates a maximum capacity for the IP group providing extra control for load and resource balancing.</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Overlap Signaling for H.323-to-H.323 Connections on a Cisco Unified Border</td>
<td>12.3(11)T</td>
<td>The terminating gateway is responsible for collecting all the called number digits. Overlap signaling is implemented by matching destination patterns on the dial peers.</td>
</tr>
<tr>
<td>Element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotary Support</td>
<td>12.3(11)T</td>
<td>12.3(11)T—H.323-to-H.323 Call Failure Recovery (Rotary) on a Cisco Unified Border Element. Eliminates codec restrictions and enables the Cisco UBE to restart codec negotiation with the originating endpoint based on the codec capabilities of the next dial peer in the rotary group for H.323-to-H.323 interconnections.</td>
</tr>
<tr>
<td>Signal Interworking</td>
<td>12.3(11)T</td>
<td>H.323-to-H.323 Interworking Between Fast Start and Slow Start. This feature enables the Cisco UBE to bridge calls between VoIP endpoints that support only H.323 FastStart procedures and endpoints that support only normal H.245 signaling (SlowStart).</td>
</tr>
</tbody>
</table>

**Prerequisites**

- Enable CUBE application on a device
- Perform basic H.323 gateway configuration. See Configuring H.323 Gateway
- Perform basic H.323 gatekeeper configuration. See Configuring H.323 Gatekeeper

**Restrictions**

- Voice class codec is not supported.
- LTI-based transcoding is not supported.
- Supplementary services with transcoding is not supported.
- DTMF Interworking rtp-nte to out of band is not supported when high density transcoder is enabled. Use normal transcoding for rtp-nte to out of band DTMF interworking.
- SCCP based codec transcoding is not supported. An exception to this restriction is slow start to slow start with a static codec.

**Slow Start to Fast-Start Interworking**

The slow-start to fast-start interworking feature allows two endpoints configured for slow start and fast start respectively to connect with each other through CUBE without dropping the call.
Restrictions for Slow-Start and Fast-Start Interworking

- Slow-start to fast-start interworking is supported only for H.323-to-H.323 calls.
- Transcoding in slow-start to fast-start interworking is not supported.

Enabling Interworking between Slow Start and Fast Start

Configure interworking between slow start and fast start on both inbound and outbound call legs.

Note

This task should not be used in situations where fast-start to fast-start or slow-start to slow-start calls are possible.

Before you begin

Ensure that a codec is configured on incoming and outgoing call legs.

SUMMARY STEPS

1. enable
2. configure terminal
3. Use one of the following commands to configure interworking between slow start and fast start.
   - call start interwork in global VoIP configuration mode
   - call start interwork in voice class configuration and applied to inbound and outbound dial peers.
4. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Router&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3 Use one of the following commands to configure interworking between slow start and fast start.</td>
<td>Enables interworking between slow start and fast start.</td>
</tr>
<tr>
<td>• call start interwork in global VoIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>• call start interwork in voice class configuration and applied to inbound and outbound dial peers.</td>
<td></td>
</tr>
</tbody>
</table>
### Call Failure Recovery (Rotary)

Call failure recovery (Rotary) is a feature that provides the flexibility to route a call to a destination with multiple paths based on the policy of a service provider. If one path disconnects the call for any reason (like unreachableDestination, destinationReject, noPermission etc), the call can be routed by choosing another dial peer to the same destination based on configured preference.

Rotary is implemented using the dial peer hunt feature (see Configuring Hunt Groups), and the search for a successful dial peer continues until a huntstop command is encountered.

The feature described in this chapter is an enhancement that removes a restriction on codec configuration, that requires for identical codec capabilities configured on all dial peers in a rotary group. This is done by supporting an Empty Capability set (TCS=0) when rotary is configured.

The feature allows the CUBE to restart the codec negotiation process with the originating endpoint based on the codec capabilities of the next dial peer in the rotary group.

### Enabling Call Failure Recovery (Rotary) without Identical Codec Configuration

**Before you begin**

Configure Call Failure Recovery (Rotary) using dial-peer hunt groups. See Configuring Dial-Peer Hunt Groups.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. h323
5. emptycapability
6. exit

### Detailed Steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>voice service voip</td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>h323</td>
<td>Enters H.323 voice-service configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(conf-voi-serv)# h323</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>emptycapability</td>
<td>Enables call failure recovery (TCS=0) without the need for identical codec configuration.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(conf-serv-h323)# emptycapability</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(conf-serv-h323)# exit</td>
<td></td>
</tr>
</tbody>
</table>

### Managing H.323 IP Group Call Capacities

Managing maximum capacity for an IP group is done with carrier IDs created on an IP trunk group. If you do not configure specific carrier IDs, you can use the `ip circuit default only` command to create a single carrier. However, if you want to use carrier ID-based routing, or if you need extra control for load and resource balancing, you must configure carrier IDs in conjunction with the `voice source-group` command.

CUBE works with the `voice source-group` command to provide matching criteria for incoming calls. The `voice source-group` command assigns a name to a set of source IP group characteristics. The terminating gateway uses these characteristics to identify and translate the incoming VoIP call. If there is no voice source group match, the default carrier ID is used, any source carrier ID on the incoming message is transmitted
without change, and no destination carrier is available. Call-capacity information is reported to the gatekeeper, but carrier routing information is not.

If the voice source group matches, the matched source carrier ID is used and the target carrier ID defined in the voice source group is used for the destination carrier ID.

---

**Note**
You can use this task only when there are no active calls are active.

---

**SUMMARY STEPS**

1. **enable**
2. **configure terminal**
3. **voice service voip**
4. **h323**
5. **ip circuit max-calls maximum-calls**
6. **ip circuit carrier-id carrier-name [reserved-calls reserved ]**
7. **ip circuit default only**
8. **exit**

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> h323</td>
<td>Enters H.323 voice-service configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(conf-voi-serv)# h323</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>5</td>
<td>ip circuit max-calls <em>maximum-calls</em></td>
</tr>
<tr>
<td>6</td>
<td>ip circuit carrier-id <em>carrier-name</em> [reserved-calls <em>reserved</em> ]</td>
</tr>
<tr>
<td>7</td>
<td>ip circuit default only</td>
</tr>
<tr>
<td>8</td>
<td>exit</td>
</tr>
</tbody>
</table>

**Configuration Examples for Managing H.323 IP Group Call Capacities**

The following examples show a default carrier with no voice source group configured:

**Example: Default Carrier with No Voice Source Group**

```
voice service voip
```
allow-connections h323 to h323
h323
ip circuit max-calls 1000
ip circuit default only

If there is no incoming source carrier ID:

• Capacity only is reported to the gatekeeper using the default circuit (two call legs).
• No source or destination carrier information is reported.

If there is an incoming source carrier ID:

• Two call legs are counted against the default circuit and reported to the GK.
• The source carrier ID is passed through the gateway to the terminating leg.

The following examples show a configuration with more reserved calls than the default value for the max-calls argument (1000):

**Example: Configuration with Default Calls in Excess of 1000**

This example assigns 1100 calls to other carriers, leaving 400 calls available to the default carrier:

voice service voip
allow-connections h323 to h323
h323
ip circuit max-calls 1000
ip circuit carrier-id AA reserved-calls 500
ip circuit carrier-id bb reserved-calls 500
ip circuit carrier-id cc reserved-calls 100

The following examples show the default carrier configured with an incoming source carrier but no voice source group configured.

**Note**

In this example, 800 call legs are implicitly reserved for the default circuit.

**Example: Default Carrier and Incoming Source Carrier with No Voice Source Group**

**Note**

A gatekeeper is required with carrier-id routing.

voice service voip
allow-connections h323 to h323
h323
ip circuit max-calls 1000
ip circuit carrier-id AA reserved-calls 200

If there is no incoming source carrier ID:

• Capacity only is reported to the GK using the default circuit (two call legs).
• No source or destination carrier information is reported.

If there is an incoming source carrier ID called “AA”:

• One call leg is counted against circuit “AA”.
• One call leg (outbound) is counted against the default circuit.
• The source carrier ID is passed through the gateway to the terminating leg.

If there is an incoming source carrier ID called “BB” (for example) or anything other than “AA”:
  • Two call legs are counted against the default circuit.
  • The source carrier ID “BB” is passed through the gateway to the terminating leg.

The following examples show the first voice source-group match case:

**Example: Voice Source-Group Match Case 1**

```plaintext
voice service voip
  allow-connections h323 to h323
  h323
    ip circuit max-calls 1000
    ip circuit carrier-id AA reserved-calls 200
  !
  voice source-group 1
  carrier-id source AA
  carrier-id target AA
```

If there is no incoming source carrier ID, the default circuit is used because there is no match in the voice source group.

If there is an incoming source carrier ID called “AA,” the following are in effect:
  • The voice source group matches.
  • Both call legs are counted against circuit “AA”.
  • The source carrier ID is passed through the gateway to the terminating leg.
  • The destination carrier ID is “AA”.

The following examples show the second voice source group match case:

**Example: Voice Source-Group Match Case 2**

```plaintext
voice service voip
  allow-connections h323 to h323
  h323
    ip circuit max-calls 1000
    ip circuit carrier-id AA reserved-calls 200
    ip circuit carrier-id BB reserved-calls 200
  !
  voice source-group 1
  carrier-id source AA
  carrier-id target BB
```

If there is no incoming source carrier ID, the default circuit is used because there is no match in the voice source group.

If there is an incoming source carrier ID called “AA”:
  • The voice source-group matches.
  • One leg is counted against circuit “AA”.
  • One leg is counted against circuit “BB”.
  • The source carrier ID is passed through the gateway to the terminating leg.
  • The destination carrier ID is “BB”.

The following examples show the third voice source-group match case:
Example: Voice Source-Group Match Case 3

```
voice service voip
  allow-connections h323 to h323
h323
  ip circuit max-calls 1000
  ip circuit carrier-id AA reserved-calls 200
  ip circuit carrier-id BB reserved-calls 200
!
voice source-group 1
  access-list 1
  carrier-id source BB

If the access-list matches, the following apply:
  • One leg is counted against circuit “BB”.
  • One leg is counted against the default circuit (for the destination circuit).
  • The source carrier ID is synthesized to “BB” and used to report to the gatekeeper. It is also used on the
    outgoing setup.

If a source carrier ID is received on the incoming setup, it is overridden with the synthesized carrier ID
```

**Overlap Signaling**

Overlap signaling requires that called digits be sent one-by-one as they are received from the calling device. The first digit is sent in a call setup message and subsequent digits are sent in information messages. This technique is used when a receiving gateway is able to recognize variable-length phone numbers, and requires that the originating gateway signal the end of the call setup process.

Overlap signaling is implemented by matching destination patterns on the dial peers. When H.225 signal overlap is configured on the originating gateway, it sends the SETUP to the terminating gateway once a dial-peer match is found. The originating gateway sends all further digits received from the user to the terminating gateway using INFO messages until it receives a sending complete message from the user. The terminating gateway receives the digits in SETUP and subsequent INFO messages and does a dial-peer match. If a match is found, it sends a SETUP with the collected digits to the PSTN. All subsequent digits are sent to the PSTN using INFO messages to complete the call.

**Configuring Overlap Signaling**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. h323
5. h225 signal overlap
6. h225 timeout t302 seconds
7. exit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>voice service voip</td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>h323</td>
<td>Enters H.323 voice-service configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-voi-serv)# h323</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>h225 signal overlap</td>
<td>Activates overlap signaling to the destination gateway.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-serv-h323)# h225 signal overlap</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>h225 timeout t302 seconds</td>
<td>Sets the t302 timer timeout value. The argument is as follows:</td>
</tr>
<tr>
<td>Example:</td>
<td>• seconds—Number of seconds for timeouts. Range: 1 to 30.</td>
</tr>
<tr>
<td>Router(conf-serv-h323)# h225 timeout t302 15</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-serv-h323)# exit</td>
<td></td>
</tr>
</tbody>
</table>

### Verifying H.323-to-H.323 Interworking

To verify Cisco Unified Border Element feature configuration and operation, perform the following steps (listed alphabetically) as appropriate.

---

**Note**

The word “calls” refers to call legs in some commands and output.
SUMMARY STEPS

1. show call active video
2. show call active voice
3. show call active fax
4. show call history video
5. show call history voice
6. show call history fax
7. show crm
8. show dial-peer voice
9. show running-config
10. show voip rtp connections

DETAILED STEPS

Step 1  show call active video
Use this command to display the active video H.323 call legs.

Step 2  show call active voice
Use this command to display call information for voice calls that are in progress.

Step 3  show call active fax
Use this command to display the fax transmissions that are in progress.

Step 4  show call history video
Use this command to display the history of video H.323 call legs.

Step 5  show call history voice
Use this command to display the history of voice call legs.

Step 6  show call history fax
Use this command to display the call history table for fax transmissions that are in progress.

Step 7  show crm
Use this command to display the carrier ID list or IP circuit utilization.

Step 8  show dial-peer voice
Use this command to display information about voice dial peers.

Step 9  show running-config
Use this command to verify which H.323-to-H.323, H.323-to-SIP, or SIP-to-SIP connection types are supported.

Step 10 show voip rtp connections
Use this command to display active Real-Time Transport Protocol (RTP) connections.
Troubleshooting H.323-to-H.323 Interworking

⚠️ Caution

Under moderate traffic loads, these `debug` commands produce a high volume of output.

- `debug cch323 all`
- `debug h225 asn1`
- `debug h225 events`
- `debug h225 q931`
- `debug h245 asn1`
- `debug h245 events`
- `debug voip ipipgw`
- `debug voip ccapi inout`
CHAPTER 48

SIP RFC 2782 Compliance with DNS SRV Queries

Effective with Cisco IOS XE Release 2.5, the Domain Name System Server (DNS SRV) query used to determine the IP address of the user endpoint is modified in compliance with RFC 2782 (which supersedes RFC 2052). The DNS SRV query prepends the protocol label with an underscore "_" character to reduce the risk of duplicate names being used for unrelated purposes. The form compliant with RFC 2782 is the default style.

- Finding Feature Information, on page 615
- Prerequisites SIP RFC 2782 Compliance with DNS SRV Queries, on page 615
- Information SIP RFC 2782 Compliance with DNS SRV Queries, on page 616
- How to Configure SIP-RFC 2782 Compliance with DNS SRV Queries, on page 616
- Configuring DNS Server Lookups, on page 617
- Verifying, on page 619
- Feature Information for SIP RFC 2782 Compliance with DNS SRV Queries, on page 619

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites SIP RFC 2782 Compliance with DNS SRV Queries

Cisco Unified Border Element

- Cisco IOS Release 12.2(8)T or a later release must be installed and running on your Cisco Unified Border Element.

Cisco Unified Border Element (Enterprise)

- Cisco IOS XE Release 2.5 or a later release must be installed and running on your Cisco ASR 1000 Series Router.
Information SIP RFC 2782 Compliance with DNS SRV Queries

Session Initiation Protocol (SIP) on Cisco VoIP gateways uses the DNS SRV query to determine the IP address of the user endpoint. The query string has a prefix in the form of "protocol.transport." and is attached to the fully qualified domain name (FQDN) of the next hop SIP server. This prefix style originated in RFC 2052. Beginning with Cisco IOS XE Release 2.5, a second style, in compliance with RFC 2782, prepends the protocol label with an underscore "_"; for example, "_protocol._transport." The addition of the underscore reduces the risk of the same name being used for unrelated purposes. The form compliant with RFC 2782 is the default style.

Note

The DNS SRV lookup is always attempted first for a Fully Qualified Domain Name (FQDN). If the DNS SRV lookup fails CUBE falls back to A-AAAA lookup. If you manually add a port number to a FQDN, the CUBE performs an A-AAAA lookup instead of SRV lookup.

Example:

'session target dns:cisco.com' would perform an SRV lookup and 'session target dns:cisco.com:5060' would perform an A-AAAA lookup.

How to Configure SIP-RFC 2782 Compliance with DNS SRV Queries

Configuring DNS Server Query Format RFC 2782 Compliance with DNS SRV Queries

Compliance with RFC 2782 changes the DNS SRV protocol label style. RFC 2782 updates RFC 2052 by prepending the protocol label with an underscore character. The prefix format compliant with RFC 2782 is the default format. However, backward compatibility is available, allowing newer versions of Cisco IOS software to work with older networks that support only RFC 2052 DNS SRV prefix style.

To configure the format of DNS SRV queries to comply with RFC 2782, complete this task.

Note

You do not have to perform this task if you want to use the default RFC 2782 format.

SUMMARY STEPS

1. enable
2. configure terminal
3. interface type number
4. sip-ua
5. srv version {1 | 2}
6. exit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
Example:  
Router> enable |
| **Step 2** configure terminal | Enters global configuration mode.  
Example: |
| **Step 3** interface type number | Configures an interface type and enters interface configuration mode  
Example:  
Router(config)# interface gigabitethernet 0/0/0 |
| **Step 4** sip-ua | Enters SIP UA configuration mode.  
Example:  
Router(config-if)# sip-ua |
| **Step 5** srv version {1 | 2} | Generates DNS SRV queries in either RFC 2782 or RFC 2052 format.  
Example:  
Router(config-sip-ua)# srv version 2  
- 1 --The query is set to the domain name prefix of protocol.transport. (RFC 2052 style).  
- 2 --The query is set to the domain name prefix of _protocol._transport. (RFC 2782 style). This is the default. |
| **Step 6** exit | Exits the current configuration mode.  
Example:  
Router(config-sip-ua)# exit |

### Configuring DNS Server Lookups

Following is the example to configure '_sip._udp.'

```
!  
dial-peer voice 1 voip  
  session protocol sipv2  
  session transport udp  
  session target dns:cisco.com  
!  
```

Following are the examples to configure '_sip._tcp.'
dial-peer voice 1 voip
    session protocol sipv2
    session transport tcp
    session target dns:cisco.com

Following is the example to configure '_sips._tcp.'.

dial-peer voice 1 voip
    session protocol sipv2
    session transport tcp tls
    session target dns:cisco.com
    voice-class sip url sips

Following is the sample configuration for a local DNS SRV.

ip name-server 172.18.110.64
ip domain lookup
ip host 1.cisco.com 10.10.10.1
ip host 2.cisco.com 10.10.10.2
ip host 3.cisco.com 10.10.10.3
ip host _sip._tcp.cisco.com srv 1 50 5061 1.cisco.com
ip host _sip._tcp.cisco.com srv 1 50 5061 2.cisco.com
ip host _sip._tcp.cisco.com srv 1 50 5061 3.cisco.com
ip host _sips._tcp.cisco.com srv 1 50 5061 1.cisco.com
ip host _sips._tcp.cisco.com srv 1 50 5061 2.cisco.com
ip host _sips._tcp.cisco.com srv 1 50 5061 3.cisco.com
ip host _sip._udp.cisco.com srv 1 50 5060 1.cisco.com
ip host _sip._udp.cisco.com srv 1 50 5060 2.cisco.com
ip host _sip._udp.cisco.com srv 1 50 5060 3.cisco.com
ip host _sip._tcp.cisco.com srv 1 50 5060 1.cisco.com
ip host _sip._tcp.cisco.com srv 1 50 5060 2.cisco.com
ip host _sip._tcp.cisco.com srv 1 50 5060 3.cisco.com

Locally hosted DNS SRV entries are not supported until IOS-XE release 3.17S.

Troubleshooting Tips
You can use the following commands to troubleshoot the DNS SRV issues.
debug ip dns view
debut ip domain
debug csip info
debug csip messages

Verifying

The following example shows sample output from the `show sip-ua status` command used to verify the style of DNS server queries:

Router# show sip-ua status
SIP User Agent Status
SIP User Agent for UDP : ENABLED
SIP User Agent for TCP : ENABLED
SIP User Agent bind status(signaling): DISABLED
SIP User Agent bind status(media): DISABLED
SIP max-forwards : 6
SIP DNS SRV version: 1 (rfc 2052)

Feature Information for SIP RFC 2782 Compliance with DNS SRV Queries

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

ISR feature history table entry

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| SIP: RFC 2782 Compliance of DNS SRV Queries | 12.2(8)T, 12.2(11)T, 12.2(15)T | Effective with Cisco IOS XE Release 2.5, the DNS SRV query used to determine the IP address of the user endpoint is modified in compliance with RFC 2782 (which supersedes RFC 2052). The DNS SRV query prepends the protocol label with an underscore "_" character to reduce the risk of duplicate names being used for unrelated purposes. The form compliant with RFC 2782 is the default style.

The following command was introduced or modified: `srv version`.

ASR feature history table entry
Table 72: Feature Information for SIP: RFC 2782 Compliance with DNS SRV Queries

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP: RFC 2782 Compliance of DNS SRV Queries</td>
<td>Cisco IOS XE Release 2.5</td>
<td>Effective with Cisco IOS XE Release 2.5, the DNS SRV query used to determine the IP address of the user endpoint is modified in compliance with RFC 2782 (which supersedes RFC 2052). The DNS SRV query prepends the protocol label with an underscore &quot;_&quot; character to reduce the risk of duplicate names being used for unrelated purposes. The form compliant with RFC 2782 is the default style. The following command was introduced or modified: <code>srv version</code>.</td>
</tr>
</tbody>
</table>
PART XIII

Support for SRTP

- SRTP-SRTP Interworking, on page 623
- SRTP-RTP Interworking, on page 639
- SRTP-SRTP Pass-Through, on page 653
SRTP-SRTP Interworking

Cisco Unified Border Element (CUBE) supports secure calls between two networks having different cipher suites. SRTP-SRTP interworking is supported for audio and video calls.

- Feature Information for SRTP-SRTP Interworking, on page 623
- Prerequisites for SRTP-SRTP Interworking, on page 624
- Restrictions for SRTP-SRTP Interworking, on page 624
- Information About SRTP-SRTP Interworking, on page 624
- How to Configure SRTP-SRTP Interworking, on page 626
- Configuration Examples, on page 634

Feature Information for SRTP-SRTP Interworking

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Readiness Criteria (SRC)—Modified the command show sip-ua calls.</td>
<td>Cisco IOS XE Gibraltar Release 16.11.1a</td>
<td>Command <strong>show sip-ua calls</strong> is modified to display local crypto key and remote cryto key.</td>
</tr>
</tbody>
</table>
| Support for SRTP-SRTP interworking | Cisco IOS XE Everest 16.5.1b     | This feature allows secure calls between two enterprises using different cipher suites. Supported cipher suites are as follows:  
  - AEAD_AES_256_GCM  
  - AEAD_AES_128_GCM  
  - AES_CM_128_HMAC_SHA1_80  
  - AES_CM_128_HMAC_SHA1_32 |
Prerequisites for SRTP-SRTP Interworking

- Cisco IOS XE Everest Release 16.5.1b or later

Note

SRTP-SRTP Interworking feature is not supported on Cisco ISR G2 Series Routers.

Restrictions for SRTP-SRTP Interworking

- Asymmetric SRTP fallback configuration is not supported
- Call admission control (CAC) is not supported
- Call Progress Analysis (CPA) is not supported
- Transcoding calls are not supported
- SRTCP-RTCP interworking is not supported
- More than one audio and video m-line is not supported
- Unified CME and Unified SRST flows and SIP-TDM flows are not supported

Information About SRTP-SRTP Interworking

From Cisco IOS XE Everest Release 16.5.1b onwards, when SRTP is enabled, by default Cisco Unified Border Element supports secure calls between networks using different cipher suites. The cipher suites supported for SRTP-SRTP interworking with default preference order is as follows:

- AEAD_AES_256_GCM
- AEAD_AES_128_GCM
- AES_CM_128_HMAC_SHA1_80
- AES_CM_128_HMAC_SHA1_32
CUBE allows you to change the list of preference order of the cipher-suites. Cipher-suite preference can be configured globally (under voice service voip >> sip), on a voice class tenant, or on a dial-peer.

The preference range is from 1 to 4, where 1 represents highest preference. CUBE offers SRTP cipher-suites in SDP offer based on the preference configured. For SDP answer, the highest configured preference cipher-suite that matches the offer from peer is selected.

**Supplementary Services Support**

The following supplementary services are supported:

- Midcall codec change with voice class codec configuration
- Reinvite-based call hold and resume
- Music on hold (MoH) invoked from the Cisco Unified Communications Manager (Cisco UCM), where the call leg changes between SRTP and RTP for an MoH source
- Reinvite-based call forward and call transfer
- Call transfer based on a REFER message, with local consumption or pass-through of the REFER message on the CUBE
- Call forward based on a 302 message, with local consumption or pass-through of the 302 message on the CUBE
- T.38 fax switchover
- Fax pass-through switchover

For call transfers involving REFER and 302 messages (messages that are locally consumed on CUBE), end-to-end media renegotiation is initiated from CUBE only when you configure the supplementary-service media-renegotiate command in voice service voip configuration mode.
Any call-flow wherein there is a switchover from RTP to SRTP on the same SIP call-leg requires the **supplementary-service media-renegotiate** command enabled in global or voice service voip configuration mode to ensure there is 2-way audio.

Example call-flows:

- RTP -SRTP transfer on CUCM side
- Non-secure MOH being played during secure call hold or resume

When supplementary services are invoked from the end points, the call can switch between SRTP and RTP during the call duration. Hence, Cisco recommends that you configure such SIP trunks for SRTP fallback. For information on configuring SRTP fallback, refer Enabling SRTP Fallback, on page 631.

# How to Configure SRTP-SRTP Interworking

## Configuring SRTP

### SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice tag voip
4. destination-pattern string
5. session protocol sipv2
6. session target ipv4:destination-address
7. incoming called-number string
8. srtp
9. codec codec
10. end
11. dial-peer voice tag voip
12. Repeat Steps 4, 5, 6, and 7 to configure a second dial peer.
13. srtp
14. codec codec
15. exit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>Step 2</td>
<td><strong>configure terminal</strong>&lt;br&gt;Example: <code>Device# configure terminal</code></td>
</tr>
<tr>
<td>Step 3</td>
<td><strong>dial-peer voice</strong> <em>tag voip</em>&lt;br&gt;Example: <code>Device(config)# dial-peer voice 201 voip</code></td>
</tr>
</tbody>
</table>
| | | • In the example, the following parameters are set:  
  • Dial peer 201 is defined.  
  • VoIP is shown as the method of encapsulation. |
| Step 4 | **destination-pattern** *string*<br>Example: `Device(config-dial-peer)# destination-pattern 5550111` | Specifies either the prefix or the full E.164 telephone number to be used for a dial peer string. |
| | | • In the example, 5550111 is specified as the pattern for the telephone number. |
| Step 5 | **session protocol sipv2**<br>Example: `Device(config-dial-peer)# session protocol sipv2` | Specifies a session protocol for calls between local and remote routers using the packet network. |
| | | • In the example, the `sipv2` keyword is configured so that the dial peer uses the SIP protocol. |
| Step 6 | **session target ipv4:***destination-address*<br>Example: `Device(config-dial-peer)# session target ipv4: 10.13.25.102` | Designates an IP address where calls will be sent. |
| | | • In the example, calls matching this outbound dial-peer will be sent to 10.13.25.102. |
| Step 7 | **incoming called-number** *string*<br>Example: `Device(config-dial-peer)# incoming called-number 5550111` | Specifies a digit string that can be matched by an incoming call to associate the call with a dial peer. |
| | | • In the example, 5550111 is specified as the pattern for the E.164 or private dialing plan telephone number. |
| Step 8 | **srtp**<br>Example: `Device(config-dial-peer)# srtp` | Specifies that SRTP is used to enable secure calls for the dial peer. |
| Step 9 | **codec** *codec*<br>Example: `Device(config-dial-peer)# codec g711ulaw` | Specifies the voice coder rate of speech for the dial peer. |
| | | • In the example, G.711 mu-law at 64,000 bps, is specified as the voice coder rate for speech. |
### Configuring Cipher Suite Preference (optional)

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 10</strong></td>
<td>Exits dial peer voice configuration mode.</td>
</tr>
<tr>
<td><code>end</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)#end</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>Defines a particular dial peer, to specify the method of voice encapsulation, and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><code>dial-peer voice tag voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# dial-peer voice 200 voip</code></td>
<td></td>
</tr>
<tr>
<td>- In the example, the following parameters are set:</td>
<td></td>
</tr>
<tr>
<td>- Dial peer 200 is defined.</td>
<td></td>
</tr>
<tr>
<td>- VoIP is shown as the method of encapsulation.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td>Repeat Steps 4, 5, 6, and 7 to configure a second dial peer.</td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>Specifies that SRTP is used to enable secure calls for the dial peer.</td>
</tr>
<tr>
<td><code>srtp</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# srtp</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td>Specifies the voice coder rate of speech for the dial peer.</td>
</tr>
<tr>
<td><code>codec codec</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# codec g711ulaw</code></td>
<td></td>
</tr>
<tr>
<td>- In the example, G.711 mu-law at 64,000 bps, is specified as the voice coder rate for speech.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td>Exits dial peer voice configuration mode.</td>
</tr>
<tr>
<td><code>exit</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# exit</code></td>
<td></td>
</tr>
</tbody>
</table>

### Configuring Cipher Suite Preference (optional)

#### Note
No additional configurations are required if you want to configure the default preference order. Use the following procedure for changing the default preference.

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `voice class srtp-crypto tag`
4. `crypto preference cipher-suite`
5. `exit`
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
</tbody>
</table>
| `enable`          | Enables privileged EXEC mode.  
| **Example:**      |         |
| `Device> enable`  | • Enter your password if prompted. |
| **Step 2**        |         |
| `configure terminal` | Enters global configuration mode. |
| **Example:**      |         |
| `Device# configure terminal` |         |
| **Step 3**        |         |
| `voice class srtp-crypto tag` | Enters voice class configuration mode and assign an identification tag for a srtp-crypto voice class. |
| **Example:**      |         |
| `Device(config)# voice class srtp-crypto 100` |         |
| **Step 4**        |         |
| `crypto preference cipher-suite` | Specifies the preference for an SRTP cipher-suite that will be offered by Cisco Unified Border Element (CUBE) in the SDP in offer and answer.  
| **Example:**      | You can configure a maximum of four preferences. |
| `Device(config-class)# crypto 1 AEAD_AES_256_GCM` |         |
| **Step 5**        |         |
| `exit`            | Exists the present configuration mode. |
| **Example:**      |         |
| `Device(config-class)# exit` |         |

#### What to do next

Assign SRTP Crypto voice class globally, or on a voice-class tenant, or on a dial-peer. For more information, see Applying Crypto Suite Selection Preference (optional), on page 629.

### Applying Crypto Suite Selection Preference (optional)

#### Before you begin

- Ensure that an srtp voice-class is created using the `voice class srtp-crypto crypto-tag` command

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. Apply crypto suite selection preference
   - In global configuration mode:
     - `voice service voice`
Applying Crypto Suite Selection Preference (optional)

- sip
  - srtp-crypto crypto-tag

- In voice class tenant configuration mode:
  - voice class tenant tag
  - srtp-crypto crypto-tag

- In dial-peer configuration mode:
  - dial-peer voice tag voip
  - voice-class sip srtp-crypto crypto-tag

4. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td>Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> Apply crypto suite selection preference</td>
<td>Assigns previously configured crypto-suite selection preference. The crypto-tag maps to the tag created using the voice class srtp-crypto command available in global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In global configuration mode:</td>
<td></td>
</tr>
<tr>
<td>- voice service voice</td>
<td></td>
</tr>
<tr>
<td>- sip</td>
<td></td>
</tr>
<tr>
<td>- srtp-crypto crypto-tag</td>
<td></td>
</tr>
<tr>
<td>In voice class tenant configuration mode:</td>
<td></td>
</tr>
<tr>
<td>- voice class tenant tag</td>
<td></td>
</tr>
<tr>
<td>- srtp-crypto crypto-tag</td>
<td></td>
</tr>
<tr>
<td>In dial-peer configuration mode:</td>
<td></td>
</tr>
<tr>
<td>- dial-peer voice tag voip</td>
<td></td>
</tr>
<tr>
<td>- voice-class sip srtp-crypto crypto-tag</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Device> enable**  
Device# configure terminal  
Device(config)# voice service voice  
Device(conf-voi-serv)# sip  
Device(conf-serv-sip)# srtp-crypto 102 |

**In voice class tenant configuration mode:**

<table>
<thead>
<tr>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Device> enable**  
Device# configure terminal  
Device(config)# voice class tenant 100  
Device(config-serv-sip)# srtp-crypto 102 |

**In dial-peer configuration mode:**

<table>
<thead>
<tr>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Device> enable**  
Device# configure terminal  
Device(config)# dial-peer voice 300 voip  
Device(config-dial-peer)# voice-class sip srtp-crypto 102 |

#### Step 4  
**end**

**Example:**

Device(config-dial-peer)# exit

---

### Enabling SRTP Fallback

You can configure SRTP with the fallback option so that a call can fall back to RTP if SRTP is not supported by the other call end. Enabling SRTP fallback is required for supporting nonsecure supplementary services such as MoH, call forward, and call transfer.

**SUMMARY STEPS**

1. enable  
2. configure terminal  
3. Enter one of the following commands:
   1. In dial-peer configuration mode
      - dial-peer  
      - voice  
      - tag  
      - voip  
      - srtp  
      - fallback (for interworking with devices other than Cisco Unified Communications Manager)
   or
voice-class sip srtp
negotiate cisco (Enable this CLI along with srtp fallback command to support SRTP fallback with Cisco Unified Communications Manager)

• In global VoIP SIP configuration mode

voice service voip
sip
srtp
fallback (for interworking with devices other than Cisco Unified Communications Manager)
or
srtp
negotiate cisco (Enable this CLI along with srtp fallback command to support SRTP fallback with Cisco Unified Communications Manager)

4. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>Command or Action</strong></td>
</tr>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>enable</td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>Command or Action</strong></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>Command or Action</strong></td>
</tr>
<tr>
<td>Enter one of the following commands:</td>
<td>Enters call fallback to nonsecure mode.</td>
</tr>
<tr>
<td>• In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>dial-peer</td>
<td></td>
</tr>
<tr>
<td>voice</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td></td>
</tr>
<tr>
<td>voip</td>
<td></td>
</tr>
<tr>
<td>srtp</td>
<td></td>
</tr>
<tr>
<td>fallback (for interworking with devices other than Cisco Unified Communications Manager)</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>voice-class sip srtp</td>
<td></td>
</tr>
<tr>
<td>negotiate cisco (Enable this CLI along with srtp fallback command to support SRTP fallback with Cisco Unified Communications Manager)</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| • In global VoIP SIP configuration mode  
  `voice service voip`  
  `sip`  
  `srtp`  
  `fallback` (for interworking with devices other than Cisco Unified Communications Manager)  
  or  
  `srtp negotiate cisco` (Enable this CLI along with `srtp fallback` command to support SRTP fallback with Cisco Unified Communications Manager) |  |

**Example:**

```
Device(config)# dial-peer voice 10 voip
Device(config-dial-peer)# srtp fallback
```

**Example:**

```
Device(config)# dial-peer voice 10 voip
Device(config-dial-peer)# voice-class sip srtp negotiate Cisco
```

**Example:**

```
Device(config)# voice service voip
Device(config)# sip
Device(conf-voi-serv)# srtp fallback
```

**Example:**

```
Device(config)# voice service voip
Device(config)# sip
Device(conf-voi-serv)# srtp negotiate cisco
```

**Step 4**  
**exit**  
**Example:**

```
Device(conf-voi-serv)# exit
```

Exits present configuration mode and enters privileged EXEC mode.
Configuration Examples

Example: Configuring SRTP-SRTP Interworking

The following example shows how to configure support for SRTP-SRTP interworking. In this example, the incoming call leg preference is set to AEAD_AES_256_GCM crypto-suite and the outgoing call leg preference is set to AES_CM_128_HMAC_SHA1_80 crypto-suite.

Configure SRTP:

```
Device> enable
Device# configure terminal
Device(config)# dial-peer voice 300 voip
Device(config-dial-peer)# description "inbound dialpeer for 81560"
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# incoming called-number 81560
Device(config-dial-peer)# srtp
Device(config-dial-peer)# codec g711ulaw
Device(config-dial-peer)# end

Device(config)# dial-peer voice 400 voip
Device(config-dial-peer)# destination-pattern 81560
Device(config-dial-peer)# description "outbound dialpeer for 81560"
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.13.25.102
Device(config-dial-peer)# srtp
Device(config-dial-peer)# codec g711ulaw
```

Create a voice class srtp-crypto 100 and assign AEAD_AES_256_GCM crypto-suite with highest preference:

```
Device(config)# voice class srtp-crypto 100
Device(config-class)# crypto 1 AEAD_AES_256_GCM
```

Assign srtp-crypto 100 on incoming dial-peer:

```
Device(config)# dial-peer voice 300 voip
Device(config-dial-peer)# voice-class sip srtp-crypto 100
Device(config-dial-peer)# codec g711ulaw
Device(config-dial-peer)# srtp
```

Create a voice class srtp-crypto 103 and assign AES_CM_128_HMAC_SHA1_80 crypto-suite with highest preference:

```
Device> enable
Device# configure terminal
Device(config)# voice class srtp-crypto 103
Device(config-class)# crypto 1 AES_CM_128_HMAC_SHA1_80
```

Assign srtp-crypto 103 on outgoing dial-peer:

```
Device(config)# dial-peer voice 400 voip
```
Device(config-dial-peer)# voice-class sip srtp-crypto 103
Device(config-dial-peer)# codec g711ulaw
Device(config-dial-peer)# srtp

Device# show sip-ua calls
Total SIP call legs:2, User Agent Client:1, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID : 706E9625-C4FB11E6-8008AFC8-C0129831@10.25.15.63
State of the call : STATE_ACTIVE (?)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 61230
Called Number : 81560
Called URI :
Bit Flags : 0xC04018 0x80000100 0x80
CC Call ID : 2
Local UUID : d5173c8551b25b06820edc687e50ab90
Remote UUID : 2e9094e33b815992a519f82abf6e09d2
Source IP Address (Sig ) : 10.25.16.63
Destn SIP Req Addr:Port : [10.13.25.102]:14560
Destn SIP Resp Addr:Port: [10.13.25.102]:14560
Destination Name :
Number of Media Streams : 1
Number of Active Streams: 1
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 2
Stream Type : voice+dtmf (1)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (80 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : rtp-nte
Dtmf-relay Payload Type : 101
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [10.25.16.63]:8002
Media Dest IP Addr:Port : [10.13.25.102]:14240
Local Crypto Suite : AES_CM_128_HMAC_SHA1_80
Remote Crypto Suite : AES_CM_128_HMAC_SHA1_80
Local Crypto Key : bTqZxZbxgFJddA1hE9wJGV3aKxo5vPv+Z1234tVb2
Remote Crypto Key : bTqZxZbxgFJddA1hE9wJGV3aKxo5vPv+Z9876tVb2
Mid-Call Re-Assocation Count: 0
SRTP-RTP Re-Assocation DSP Query Count: 0

Options-Ping ENABLED:NO ACTIVE:NO
Number of SIP User Agent Client(UAC) calls: 1

SIP UAS CALL INFO
Call 1
SIP Call ID : 1-8614@10.41.50.13
State of the call : STATE_ACTIVE (?)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 61230
Called Number : 81560
Called URI : sip:81560@10.13.25.102:5060
Bit Flags : 0xC0401C 0x10000100 0x4
CC Call ID : 1
Example: Changing the Cipher-Suite Preference

Specify SRTP cipher-suite preference:

Device> enable
Device# configure terminal
Device(config)# voice class srtp-crypto 100
Device(config-class)# crypto 1 AEAD_AES_256_GCM
Device(config-class)# crypto 2 AEAD_AES_128_GCM
Device(config-class)# crypto 4 AES_CM_128_HMAC_SHA1_32

The following is the snippet of show running-config command output showing the cipher-suite preference:

Device# show running-config
voice class srtp-crypto 100
crypto 1 AEAD_AES_256_GCM
crypto 2 AEAD_AES_128_GCM
crypto 4 AES_CM_128_HMAC_SHA1_32

If you want to change the preference 4 to AES_CM_128_HMAC_SHA1_80, execute the following command:
Device(config-class)# crypto 4 AES_CM_128_HMAC_SHA1_80

The following is the snippet of `show running-config` command output showing the change in cipher-suite:

Device# show running-config
voice class srtp-crypto 100
crypto 1 AEAD_AES_256_GCM
crypto 2 AEAD_AES_128_GCM
crypto 4 AES_CM_128_HMAC_SHA1_80

If you want to change the preference of AES_CM_128_HMAC_SHA1_80 to 3, execute the following commands:

Device(config-class)# no crypto 4
Device(config-class)# crypto 3 AES_CM_128_HMAC_SHA1_80

The following is the snippet of `show running-config` command output showing the cipher-suite preference overwritten:

Device# show running-config
voice class srtp-crypto 100
crypto 1 AEAD_AES_256_GCM
crypto 2 AEAD_AES_128_GCM
crypto 3 AES_CM_128_HMAC_SHA1_80
Example: Changing the Cipher-Suite Preference
CHAPTER 50

SRTP-RTP Interworking

The Cisco Unified Border Element (CUBE) Support for SRTP-RTP Interworking feature allows secure network to non-secure network calls and provides operational enhancements for Session Initiation Protocol (SIP) trunks from Cisco Unified Call Manager and Cisco Unified Call Manager Express. Support for Secure Real-Time Transport Protocol (SRTP) to Real-Time Transport Protocol (RTP) interworking in a network is enabled for SIP-SIP audio calls.

- Feature Information for SRTP-RTP Interworking, on page 639
- Prerequisites for SRTP-RTP Interworking, on page 640
- Restrictions for SRTP-RTP Interworking, on page 640
- Information About SRTP-RTP Interworking, on page 640
- How to Configure Support for SRTP-RTP Interworking, on page 644
- Configuration Examples for SRTP-RTP Interworking, on page 651

Feature Information for SRTP-RTP Interworking

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco Unified Border Element Support for SRTP-RTP Interworking</td>
<td>12.4(22)YB, 15.0(1)M Cisco IOS XE 3.1S</td>
<td>This feature allows secure to non-secure enterprise calls. Support for SRTP-RTP interworking between one or multiple Cisco Unified Border Elements is enabled for SIP-SIP audio calls.</td>
</tr>
<tr>
<td>Supplementary Services Support on CUBE for SRTP-RTP Calls</td>
<td>Cisco IOS 15.2(1)T Cisco IOS XE 3.7S</td>
<td>The SRTP-RTP Interworking feature was enhanced to support supplementary services for SRTP-RTP calls.</td>
</tr>
</tbody>
</table>
Feature Information

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for AEAD_AES_GCM_256 and AEAD_AES_GCM_128 crypto-suites</td>
<td>Cisco IOS XE Everest 16.5.1b</td>
<td>AEAD_AES_GCM_256 and AEAD_AES_GCM_128 crypto suites were added to support SRTP-RTP interworking.</td>
</tr>
</tbody>
</table>

Prerequisites for SRTP-RTP Interworking

- SRTP-RTP interworking is supported with Cisco Unified CallManager 7.0 and later releases.
- DSP resources are required for platforms running on Cisco IOS Releases. For more information on configuring DSP resources, see Transcoding.
- Platforms running on Cisco IOS XE Releases do not require DSP resources.

Restrictions for SRTP-RTP Interworking

- Asymmetric SRTP fallback configuration is not supported.
- Dial peer hunting is not supported.
- Video calls are not supported on platforms running on Cisco IOS Releases.
- More than one video m-line is not supported.
- DTMF interworking is not supported on ISR G2 (2900, 3900) series routers. It is supported only on platforms running on Cisco IOS XE Releases.

Information About SRTP-RTP Interworking

To configure support for SRTP-RTP interworking, you should understand the following concepts:

Support for SRTP-RTP Interworking

The Cisco Unified Border Element Support for SRTP-RTP Interworking feature connects SRTP Cisco Unified CallManager domains with the following:

- RTP Cisco Unified CallManager domains. Domains that do not support SRTP or have not been configured for SRTP, as shown in the figure below.
- RTP Cisco applications or servers. For example, Cisco Unified MeetingPlace, Cisco WebEx, or Cisco Unity, which do not support SRTP, or have not been configured for SRTP, or are resident in a secure data center, as shown in the figure below.
- RTP to third-party equipment. For example, IP trunks to PBXs or virtual machines, which do not support SRTP.
The Cisco Unified Border Element Support for SRTP-RTP Interworking feature connects SRTP enterprise domains to RTP SIP provider SIP trunks. SRTP-RTP interworking connects RTP enterprise networks with SRTP over an external network between businesses. This provides flexible secure business-to-business communications without the need for static IPsec tunnels or the need to deploy SRTP within the enterprise, as shown in the figure below.

SRTP-RTP interworking also connects SRTP enterprise networks with static IPsec over external networks, as shown in the figure below.

SRTP-RTP interworking on the CUBE in a network topology uses single-pair key generation. Existing audio and dual-tone multifrequency (DTMF) transcoding is used to support voice calls. SRTP-RTP interworking support is provided in both flow-through and high-density mode. There is no impact on SRTP-SRTP pass-through calls.

SRTP is configured on one dial peer using the `srtp` and `srtp fallback` commands. RTP is configured on the other dial peer. The dial peer configuration takes precedence over the global configuration on the CUBE.

Fallback handling occurs if one of the call endpoints does not support SRTP. The call can fall back to RTP-RTP, or the call can fail, depending on the configuration. Fallback takes place only if the `srtp fallback` command is configured on the respective dial peer. RTP-RTP fallback occurs when no transcoding resources are available for SRTP-RTP interworking.
Using SRTP-RTP Chain for Interworking Between AES_CM_128_HMAC_SHA1_32 and AES_CM_128_HMAC_SHA1_80 Crypto Suites

A single Cisco Unified Call Manager (CUCM) device cannot terminate a Secure Real-time Transport Protocol (SRTP) connection with an IP Phone using the AES_CM_128_HMAC_SHA1_32 crypto suite and initiate an SRTP connection with an external CUBE device with the AES_CM_128_HMAC_SHA1_80 crypto suite at the same time.

For Cisco Unified Call Manager (Unified Communications Manager) and IP Phone devices that support only AES_CM_128_HMAC_SHA1_32 crypto suite, the interim SRTP-RTP interworking solution that is described below can be implemented.

- **CUCM or IP Phone side:**
  - An SRTP connection using the AES_CM_128_HMAC_SHA1_32 crypto suite exists between the IP Phone and CUBE1.
  - An RTP connection exists between CUBE1 and CUBE2.

- **SIP trunk side—**An SRTP connection using the AES_CM_128_HMAC_SHA1_80 crypto suite is initiated by CUBE2 here. In the image below, CUBE2 is the border element on the Customer Network and SBC is the border element on the Service Provider Network.

*Figure 59: SRTP-RTP Interworking Supporting AES_CM_128_HMAC_SHA1_32 crypto suite*

*Note*

- AES_CM_128_HMAC_SHA1_32 to AES_CM_128_HMAC_SHA1_80 interworking is not supported up to Cisco IOS 15.5(3)M Release and Cisco IOS XE Everest 16.4.1 Release.

- From Cisco IOS XE Everest 16.5.1b Release onwards, SRTP-SRTP interworking is supported and therefore SRTP-RTP chain is not required.
Supplementary Services Support

The following supplementary services are supported:

- Midcall codec change with voice class codec configuration
- Reinvite-based call hold and resume
- Music on hold (MoH) invoked from the Cisco Unified Communications Manager (Cisco UCM), where the call leg changes between SRTP and RTP for an MoH source
- Reinvite-based call forward and call transfer
- Call transfer based on a REFER message, with local consumption or pass-through of the REFER message on the CUBE
- Call forward based on a 302 message, with local consumption or pass-through of the 302 message on the CUBE
- T.38 fax switchover
- Fax pass-through switchover

For call transfers involving REFER and 302 messages (messages that are locally consumed on CUBE), end-to-end media renegotiation is initiated from CUBE only when you configure the `supplementary-service media-renegotiate` command in voice service voip configuration mode.

For call transfers in involving REFER and 302 messages (messagesthatarelocallyconsumedonCUBE), end-to-end mediarenegotiationisinitiatedfromCUBEonlywhenyouconfigurethesupplementary-service media-renegotiate command in voice service voip configuration mode.

---

**Note**

Any call-flow wherein there is a switchover from RTP to SRTP on the same SIP call-leg requires the `supplementary-service media-renegotiate` command enabled in global or voice service voip configuration mode to ensure there is 2-way audio.

Example call-flows:

- RTP -SRTP transfer on CUCM side
- Non-secure MOH being played during secure call hold or resume

---

When supplementary services are invoked from the end points, the call can switch between SRTP and RTP during the call duration. Hence, Cisco recommends that you configure such SIP trunks for SRTP fallback. For information on configuring SRTP fallback, refer [Enabling SRTP Fallback, on page 631](#).
How to Configure Support for SRTP-RTP Interworking

Configuring SRTP-RTP Interworking Support

Note
From Cisco IOS XE Everest Release 16.5.1b onwards, the following crypto suites are enabled by default on the SRTP leg:

- AEAD_AES_256_GCM
- AEAD_AES_128_GCM
- AES_CM_128_HMAC_SHA1_80
- AES_CM_128_HMAC_SHA1_32

Use the following procedure for changing the default preference list.

Perform the task in this section to enable SRTP-RTP interworking support between one or multiple Cisco Unified Border Elements for SIP-SIP audio calls. In this task, RTP is configured on the incoming call leg and SRTP is configured on the outgoing call leg.

Note
This feature is available only on Cisco IOS images with security package.

SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice tag voip
4. destination-pattern string
5. session protocol sipv2
6. session target ipv4: destination-address
7. incoming called-number string
8. codec codec
9. end
10. dial-peer voice tag voip
11. Repeat Steps 4, 5, 6, and 7 to configure a second dial peer.
12. srtp
13. codec codec
14. exit
**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**  
  enable  
  Example: Device> enable     | Enables privileged EXEC mode.                                             |
|                 | • Enter your password if prompted.                                      |
| **Step 2**  
  configure terminal  
  Example: Device# configure terminal | Enters global configuration mode.                                         |
| **Step 3**  
  dial-peer voice tag voip  
  Example: Device(config)# dial-peer voice 201 voip | Defines a particular dial peer, to specify the method of voice encapsulation, and enters dial peer voice configuration mode. |
|                 | • In the example, the following parameters are set:                      |
|                 | • Dial peer 201 is defined.                                             |
|                 | • VoIP is shown as the method of encapsulation.                         |
| **Step 4**  
  destination-pattern string  
  Example: Device(config-dial-peer)# destination-pattern 5550111 | Specifies either the prefix or the full E.164 telephone number to be used for a dial peer string. |
|                 | • In the example, 5550111 is specified as the pattern for the telephone number. |
| **Step 5**  
  session protocol sipv2  
  Example: Device(config-dial-peer)# session protocol sipv2 | Specifies a session protocol for calls between local and remote routers using the packet network. |
|                 | • In the example, the sipv2 keyword is configured so that the dial peer uses the SIP protocol. |
| **Step 6**  
  session target ipv4: destination-address  
  Example: Device(config-dial-peer)# session target ipv4:10.13.25.102. | Designates an IPv4 destination address where calls will be sent. |
|                 | • In the example, calls matching this outbound dial-peer will be sent to 10.13.25.102. |
| **Step 7**  
  incoming called-number string  
  Example: Device(config-dial-peer)# incoming called-number 5550111 | Specifies a digit string that can be matched by an incoming call to associate the call with a dial peer. |
|                 | • In the example, 5550111 is specified as the pattern for the E.164 or private dialing plan telephone number. |
| **Step 8**  
  codec codec  
  Example: | Specifies the voice coder rate of speech for the dial peer. |
<p>|                 | • In the example, G.711 mu-law at 64,000 bps, is specified as the voice coder rate for speech. |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config-dial-peer)# codec g711ulaw</td>
<td></td>
</tr>
</tbody>
</table>

**Step 9**

end

Example:

Device(config-dial-peer)#end

Exits dial peer voice configuration mode.

**Step 10**

dial-peer voice tag voip

Example:

Device(config)# dial-peer voice 200 voip

Defines a particular dial peer, to specify the method of voice encapsulation, and enters dial peer voice configuration mode.

- In the example, the following parameters are set:
  - Dial peer 200 is defined.
  - VoIP is shown as the method of encapsulation.

**Step 11**

Repeat Steps 4, 5, 6, and 7 to configure a second dial peer.

--

**Step 12**

srtp

Example:

Device(config-dial-peer)# srtp

Specifies that SRTP is used to enable secure calls for the dial peer.

**Step 13**

codec codec

Example:

Device(config-dial-peer)# codec g711ulaw

Specifies the voice coder rate of speech for the dial peer.

- In the example, G.711 mu-law at 64,000 bps, is specified as the voice coder rate for speech.

**Step 14**

exit

Example:

Device(config-dial-peer)# exit

Exits dial peer voice configuration mode.

---

**Configuring Crypto Authentication**

*Note*

Effective Cisco IOS XE Everest Releases 16.5.1b, `srtp-auth` command is deprecated. Although this command is still available in Cisco IOS XE Everest software, executing this command does not cause any configuration changes. Use `voice class srtp-crypto` command to configure the preferred cipher-suites for the SRTP call leg (connection). For more information, see SRTP-SRTP Interworking.

**SUMMARY STEPS**

1. enable
2. configure terminal
### 3. Execute the commands based on your configuration mode

- In dial-peer configuration mode:
  ```
dial-peer voice tag voip
voice-class sip srtp-auth {sha1-32 | sha1-80 | system}
  ```

- In global VoIP SIP configuration mode:
  ```
voice service voip
sip
srtp-auth {sha1-32 | sha1-80}
  ```

### 4. end

### Detailed Steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Execute the commands based on your configuration mode</td>
<td>Configures an SRTP connection on CUBE using the preferred crypto suite.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# dial-peer voice 15 voip Device(config-dial-peer)# voice-class sip srtp-auth sha1-80</td>
<td>• The default value is <code>sha1-32</code>.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# voice service voip Device(conf-voi-serv)# sip Device(conf-serv-sip)# srtp-auth sha1-80</td>
<td></td>
</tr>
</tbody>
</table>
### Enabling SRTP Fallback

You can configure SRTP with the fallback option so that a call can fall back to RTP if SRTP is not supported by the other call end. Enabling SRTP fallback is required for supporting nonsecure supplementary services such as MoH, call forward, and call transfer.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. Enter one of the following commands:
   - In dial-peer configuration mode
     ```
     dial-peer
     voice
tag
     voip
     srtpt
     fallback  
     ```
     (for interworking with devices other than Cisco Unified Communications Manager)
     or
     ```
     voice-class sip srtpt
     negotiate cisco  
     ```
     (Enable this CLI along with `srtpt fallback` command to support SRTP fallback with Cisco Unified Communications Manager)
   - In global VoIP SIP configuration mode
     ```
     voice service voip
     sip
     srtpt
     fallback  
     ```
     (for interworking with devices other than Cisco Unified Communications Manager)
     or
     ```
     srtpt
     negotiate cisco  
     ```
     (Enable this CLI along with `srtpt fallback` command to support SRTP fallback with Cisco Unified Communications Manager)
4. `exit`
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | **enable**<br>Example: **Device> enable** | Enables privileged EXEC mode.  
- Enter your password if prompted. |
| Step 2 | **configure terminal**<br>Example: **Device# configure terminal** | Enters global configuration mode. |
| Step 3 | Enter one of the following commands:<br>- In dial-peer configuration mode  
  ```bash
dial-peer  
  voice  
tag  
voip  
  srtp  
  fallback (for interworking with devices other than Cisco Unified Communications Manager)  
or  
voice-class sip srtp  
egotiate cisco (Enable this CLI along with srtp fallback command to support SRTP fallback with Cisco Unified Communications Manager)  
- In global VoIP SIP configuration mode  
  ```bash
voice service voip  
sip  
  srtp  
  fallback (for interworking with devices other than Cisco Unified Communications Manager)  
or  
srtp  
egotiate cisco (Enable this CLI along with srtp fallback command to support SRTP fallback with Cisco Unified Communications Manager)  
Example: <br>**Device(config)# dial-peer voice 10 voip**  
**Device(config-dial-peer)# srtp fallback** | Enables call fallback to nonsecure mode. |
Troubleshooting Tips

The following commands help in troubleshooting SRTP-RTP supplementary services support:

- debug ccsip all
- debug voip ccaipi inout

Verifying SRTP-RTP Supplementary Services Support

Perform this task to verify the configuration for SRTP-RTP supplementary services support.

**SUMMARY STEPS**

1. enable
2. show call active voice brief

**DETAILED STEPS**

**Step 1** enable

Enables privileged EXEC mode.

**Example:**

```
Device(config)##
```
Step 2  show call active voice brief

Displays call information for voice calls in progress.

Example:

Device# show call active voice brief
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 2
multicast call-legs: 0
Total call-legs: 4

0 1 21:45:256 IST Fri Jun 3 2011.1 +29060 pid:1 Answer 10008001 connected
dur 00:01:19 tx:1653/271092 rx:2831/464284 dscp:0 media:0
IP 10.45.40.40:7892 SRTP: on rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g711ulaw TextRelay: off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a

0 2 21:49:45.256 IST Fri Jun 3 2011.2 +29060 pid:22 Originate 20009001 connected
dur 00:01:19 tx:2831/452960 rx:1653/264480 dscp:0 media:0
IP 10.45.40.40:7893 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g711ulaw TextRelay: off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a

0 3 12:50:14.326 IST Fri Jun 3 2011.1 +0 pid:0 Originate connecting
dur 00:01:19 tx:2831/452960 rx:1653/264480 dscp:0 media:0
IP 10.45.34.252:2000 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms  g711ulaw TextRelay: off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a

0 5 12:50:14.326 IST Fri Jun 3 2011.2 +0 pid:0 Originate connecting
dur 00:01:19 tx:2831/452960 rx:1653/264480 dscp:0 media:0
IP 10.45.34.252:2000 SRTP: on rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms  g711ulaw TextRelay: off
media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a

Configuration Examples for SRTP-RTP Interworking

Example: SRTP-RTP Interworking

The following example shows how to configure support for SRTP-RTP interworking. In this example, the incoming call leg is RTP and the outgoing call leg is SRTP.

%SYS-5-CONFIG_I: Configured from console by console
dial-peer voice 201 voip
destination-pattern 5550111
session protocol sipv2
session target ipv4:10.13.25.102
incoming called-number 5550112
codec g711ulaw
Example: Configuring Crypto Authentication

---

Example: Configuring Crypto Authentication (Dial Peer Level)

The following example shows how to configure Cisco UBE to support an SRTP connection using the AES_CM_128_HMAC_SHA1_80 crypto suite at the dial peer level:

```
Device> enable
Device# configure terminal
Device(config)# dial-peer voice 15 voip
Device(config-dial-peer)# voice-class sip srtp-auth sha1-80
Device(config-dial-peer)# end
```

Example: Configuring Crypto Authentication (Global Level)

The following example shows how to configure Cisco UBE to support an SRTP connection using the AES_CM_128_HMAC_SHA1_80 crypto suite at the global level:

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(config-voi-serv)# sip
Device(config-serv-sip)# srtp-auth sha1-80
Device(config-serv-sip)# end
```
SRTP-SRTP Pass-Through

SRTP-SRTP pass-through feature allows pass-through of encrypted media from one call-leg to the other.

- Feature Information for Support of SRTP-SRTP Pass-Through Calls, on page 653
- Information About SRTP-SRTP Pass-Through, on page 654
- Configure Pass-Through of Unsupported Crypto Suites for a Specific Dial Peer, on page 655
- Configure Pass-Through of Unsupported Crypto Suites Globally, on page 657
- Configuration Examples for SRTP-SRTP Pass-Through, on page 658

Feature Information for Support of SRTP-SRTP Pass-Through Calls

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 75: Feature Information for SRTP-SRTP Pass-Through

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for SRTP-SRTP Basic calls</td>
<td>12.4.15XZ</td>
<td>This feature introduced support for basic SRTP-SRTP pass-through calls.</td>
</tr>
<tr>
<td>Support for AES_CM_128_HMAC_SHA1_80 crypto suite</td>
<td>Cisco IOS 15.4(1)T</td>
<td>Support AES_CM_128_HMAC_SHA1_80 crypto suite on the Session Initiation Protocol (SIP) Trunk interface was introduced. The following commands were introduced or modified: <code>show sip-ua srtp</code>, <code>srtp-auth</code> and <code>voice-class sip srtp-auth</code>.</td>
</tr>
<tr>
<td>Support for AES_CM_128_HMAC_SHA1_80 crypto suite</td>
<td>Cisco IOS XE 3.11S</td>
<td></td>
</tr>
</tbody>
</table>
**Information About SRTP-SRTP Pass-Through**

Cisco Unified Border Element supports SIP calls between endpoints using Transport Layer Security (TLS) for SIP signaling encryption and Secure Real-Time Protocol (SRTP) to provide RTP media encryption. However, these two encryption mechanisms may not be deployed simultaneously, depending on the required call flow invoked on the associated configuration.

The following are conditions of the SRTP Passthrough feature:

- SRTP Passthrough must be configured on both legs of the call. If the target adjacency does not support SRTP Passthrough, then the call is rejected by error message 415 (Unsupported Media Type).
- "m=.. RTP/SAVP .." and a="crypto:..." fields coming in on an Invite from one adjacency are passed on in an Invite to the target adjacency.
- "m=...RTP/SAVP..." is a required field in the Invite to trigger SRTP Passthrough behavior in the SBC.

**Pass-Through of Unsupported Crypto Suites**

---

**Note**

Effective from Cisco IOS XE Everest Release 16.5.1b, CUBE supports AEAD_AES_128_GCM and AEAD_AES_256_GCM crypto-suites. For more information, see [SRTP-SRTP Interworking](#).

CUBE supports transparent passthrough of all (supported and unsupported) crypto suites.

Until Cisco IOS Release 15.6(1)T and Cisco IOS XE Release 3.17S, CUBE and DSP supported SRTP pass-through only for AES_CM_128_HMAC_SHA1_80 crypto suite.

From Cisco IOS Release 15.6(1)T and Cisco IOS XE Release 3.17S onwards, CUBE supports pass-through of the following unsupported crypto suites:

- AEAD_AES_128_GCM
- AEAD_AES_256_GCM
- AEAD_AES_128_CCM

---

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Enhanced Support for SRTP-SRTP Pass-Through | Cisco IOS 15.6(1)T Cisco IOS XE 3.17S | Introduced support for pass-through of the following unsupported crypto suites:  
• AEAD_AES_128_GCM  
• AEAD_AES_256_GCM  
• AEAD_AES_128_CCM  
• AEAD_AES_256_CCM  

The srtplib command was modified to add *pass-thru* keyword.
- AEAD_AES_256_CCM

CUBE has the ability to pass across crypto attributes (containing any unsupported crypto suites) as well as media packets (encrypted with unsupported crypto suites).

If SRTP pass-thru feature is enabled, media interworking will not be supported. Ensure that you have symmetric configuration on both the incoming and outgoing dial-peers to avoid media-related issues.

## Configure Pass-Through of Unsupported Crypto Suites for a Specific Dial Peer

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `dial-peer voice tag voip`
4. `destination-pattern string`
5. `session protocol sipv2`
6. `sessiontarget ipv4: destination-address`
7. `incoming called-number string`
8. `srtp pass-thru`
9. `codec codec`
10. `end`
11. `dial-peer voice tag voip`
12. Repeat Steps 4, 5, 6, and 7 to configure a second dial peer.
13. `srtp pass-thru`
14. `codec codec`
15. `exit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: <code>Device&gt; enable</code></td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: <code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>3</td>
<td>dial-peer voice tag voip</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 201 voip</td>
</tr>
<tr>
<td>4</td>
<td>destination-pattern string</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# destination-pattern 5550111</td>
</tr>
<tr>
<td>5</td>
<td>session protocol sipv2</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td>6</td>
<td>session target ipv4: destination-address</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session target ipv4:10.13.25.102</td>
</tr>
<tr>
<td>7</td>
<td>incoming called-number string</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# incoming called-number 5550111</td>
</tr>
<tr>
<td>8</td>
<td>srtp pass-thru</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# srtp pass-thru</td>
</tr>
<tr>
<td>9</td>
<td>codec codec</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# codec g711ulaw</td>
</tr>
<tr>
<td>10</td>
<td>end</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)#end</td>
</tr>
</tbody>
</table>
## Configure Pass-Through of Unsupported Crypto Suites Globally

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `voice service voip`
4. `srtppass-thru`
5. `end`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | `enable` | Enables privileged EXEC mode.  
  * Enter your password if prompted. |
| Example: | Device> `enable` | |
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td>Example:</td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td><code>voice service voip</code></td>
<td>Example:</td>
</tr>
<tr>
<td><code>Device(config)# voice service voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enables transparent passthrough of all crypto suites globally.</td>
</tr>
<tr>
<td><code>srtp pass-thru</code></td>
<td>Example:</td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# srtp pass-thru</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Exits dial peer voice configuration mode.</td>
</tr>
<tr>
<td><code>end</code></td>
<td>Example:</td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# end</code></td>
<td></td>
</tr>
</tbody>
</table>

### Configuration Examples for SRTP-SRTP Pass-Through

**Example for SRTP=SRTP Pass-Through**

```bash
enable
cfgure terminal
dial-peer voice 201 voip
destination-pattern 5550111
session protocol sipv2
session target ipv4:10.13.25.102
incoming called-number 5550111
srtp
codec g711ulaw
end
dial-peer voice 200 voip
destination-pattern 5550111
session protocol sipv2
session target ipv4:10.13.25.101
incoming called-number 5550111
srtp
codec g711ulaw
end
```

**Example for Pass-Through of Unsupported Crypto Suites for a specific dial peer**

```bash
enable
cfgure terminal
```
Example for Pass-Through of Unsupported Crypto Suites Globally

```
enable
configure terminal
voice service voip
srtp pass-thru
end
```
Configuration Examples for SRTP-SRTP Pass-Through
PART XIV

High Availability

• High Availability Overview, on page 663
• DSP High Availability Support, on page 667
• Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 671
• CVP Survivability TCL support with High Availability, on page 685
High Availability Overview

The High Availability (HA) feature allows Cisco UBE to preserve calls when system resources become unavailable. Cisco UBE is configured on two routers, where one router is active and the other a standby. When the active Cisco UBE becomes unavailable, standby Cisco UBE seamlessly takes over the call processing.

Figure 60: Cisco UBE High Availability

- Information About High Availability, on page 663
- Restrictions for CUBE High Availability, on page 665

Information About High Availability

Inbox versus Box-to-Box Redundancy

Refer to the next section in this document. For detailed information about inbox and box-to-box redundancies, refer to the chapter titled “Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices” in the Cisco Unified Border Element Configuration Guide.

Route Processor Redundancy

Route Processor Redundancy (RPR) allows you to configure a standby RP. When you configure RPR, the standby RP loads the Cisco IOS software on bootup and initializes itself in standby mode. In the event of a fatal error on the active RP, the system switches to the standby RP, which reinitializes itself as the active RP. In this event, the entire system is rebooted, so the switchover with RPR is slower than with other High Availability switchover features such as Nonstop Forwarding/Stateful Switchover (NSF/SSO).
Stateful Switchover

Stateful switchover (SSO) provides media preservation of calls and post-switchover teardown of calls, in case of active RP failure. This means that the CUBE calls would continue to be active even after the active RP card goes down (provided a redundant RP is present). The standby RP would become active and service new CUBE(ENT) calls. The context of the CUBE (ENT) calls that were switched over from the Active card would be present on the new active card. Hold/Resume or any other supplementary services will work after switchover. In SSO, both media and signaling session context are preserved on failover.

Note

The terms failover and switchover are used interchangeably.

Nonstop Forwarding

Nonstop forwarding (NSF) helps to suppress routing flaps in devices that are enabled with stateful switchover (SSO), thereby reducing network instability. NSF allows forwarding of data packets to continue along the known routes while the routing protocol information is being restored after a switchover. Non Stop Forwarding (NSF) works together with SSO and allows the routing protocols to reestablish their routing information by requesting their network neighbors to resend all of the routing information when a switchover occurs.

HA Checkpointing

Checkpointing refers to the facility or architecture to implement stateful switchover (SSO). It provides the mechanisms to help synchronize state data between the active and standby route processors or chassis in a consistent, repeatable, and well-ordered manner.

From Cisco IOS Release 15.6(1)T and Cisco IOS-XE Release 3.17S onwards, checkpointing mechanism is enhanced to provide support for multimedia endpoints with larger SDP up to 6000 bytes. With the new enhancement, CUBE supports preservation of media up to a maximum of 6 m-lines/streams (audio, video, and application).

CUBE HA is enhanced to support the preservation of Record-Route and Contact header information. After SSO, all subsequent midcall SIP messages will be routed based on the correct Record-Route and Contact headers.

CUBE High Availability Options

The CUBE HA implementation supports full stateful failover for active SIP-to-SIP calls. Stateful failover means both media and signaling session information is preserved after switchover. CUBE supports three types of high availability (HA) options.

Hot Standby Routing Protocol

The Cisco Unified Border Element (CUBE) provides high availability (HA) using box-to-box redundancy configurations when implemented on a Cisco ISR-G2 platform. CUBE box-to-box redundancy on ISR-G2 is based on the Hot Standby Routing Protocol (HSRP) router technology, and HSRP is specific to ISR-G2.

HSRP technology provides high network availability by routing IP traffic from hosts on networks without relying on the availability of any single router. HSRP is used in a group of routers for selecting an Active router and a Standby router. HSRP monitors both the inside and outside interfaces—if any interface goes
down, the whole device is considered down, the standby device becomes active and takes over the responsibilities of the active router. Box-to-box high availability is supported using virtual IP addresses for the signaling and media.

**Redundancy Group Infrastructure**

Cisco ASR 1000 Series Router, Cisco 4000 Series ISR, and Cisco CSR 1000v with Cisco IOS XE Release 3.11S or later, use the Redundancy Group (RG) Infrastructure to form an active/standby pair of routers. The active/standby pair shares the same virtual IP address (VIP) and continually exchange status messages. CUBE session information is check-pointed across the active/standby pair of routers enabling the Standby router to take over immediately all CUBE call processing responsibilities if the Active router goes out of service. RG Infrastructure also is supported on ASR 1006 with a single Route Processor and an Embedded Services Processor (ESP).

**Considerations for Choosing an HA Configuration**

When considering HA design, the following VoIP aspects apply:

- Media preservation of active calls
- Calls that are currently being signaled
- Signaling protocol state preservation for active calls (supplementary services will work after switchover)
- Transcoded calls
- H323-to-SIP and H323-to-H323 calls
- Licensing implications

**Note**

In High Availability deployments, CUBE uses physical IP address to communicate the Smart Licensing information.

**Restrictions for CUBE High Availability**

- IPv6 is not supported.
- All SCCP-based media resources (Conference bridge, Transcoding, Hardware MTP, and Software MTP) are not supported.
- Cisco Unified Survivable Remote Site Telephony (Unified SRST) or TDM Gateway co-location on Cisco UBE HA is not supported.
DSP High Availability Support

Cisco Unified Border Element (CUBE) DSP High Availability support for SIP-to-SIP calls is added for Box-to-Box and Inbox configurations. Earlier, calls that required DSP resources were not checkpointed. As a result, both the media and signaling sessions were not preserved after switchover resulting in call failure.

**Note**
DSP HA is supported only for SIP-to-SIP calls.

- Feature Information for DSP High Availability Support on CUBE, on page 667
- Prerequisites for DSP High Availability, on page 668
- Features Supported with DSP High Availability, on page 668
- Restrictions for DSP High Availability, on page 668
- Troubleshooting DSP HA Support on CUBE, on page 669
- How to Configure High Availability, on page 669
- Configuration Examples for DSP HA, on page 669

**Feature Information for DSP High Availability Support on CUBE**

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP HA Support on CUBE</td>
<td>Cisco IOS 15.5(2)T</td>
<td>Provides DSP High availability support for SIP-to-SIP calls on Box-to-Box and Inbox redundancies.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.15S</td>
<td></td>
</tr>
</tbody>
</table>

Table 76: Feature Information for DSP HA Support on CUBE
Prerequisites for DSP High Availability

- LTI Transcoding
- DSP HA is supported only on the following routers and its corresponding modules:
  - Cisco ISR G2 series (PVDM3)
  - Cisco ASR 1000 series (SPA-DSP)
  - Cisco ISR 4000 series (PVDM4)
- The same type and capacity DSP modules must be used in the Active and Standby CUBE devices (box-to-box)
- The DSP modules must be installed in the same slot and subslot in the Active and Standby CUBE devices (box-to-box)
- The Active and Standby CUBE devices must have the same DSPFARM configurations (box-to-box)

Features Supported with DSP High Availability

- Transcoding with Supplementary Services
- Voice Class Codec
- G.711 in-band -> RFC2833 (RTP-NTE) DTMF interworking variant
- SRTP-RTP Interworking (ISR-G2 only)
- Fax calls with transcoder invoked for codec mis-match

Restrictions for DSP High Availability

- Media flow-around calls are not supported.
- SDP passthrough calls are not supported.
- Audio Transrating is not supported.
- Call Progress Analysis is not supported.
- Dolby Noise Reduction (NR) and Acoustic Shock Protection (ASP) are not supported.
- All SCCP-based media resources (Conference bridge, Transcoding, HW MTP, and SW MTP) are not supported with Cisco Unified Border Element High Availability.
Troubleshooting DSP HA Support on CUBE

You can use the following debug commands to troubleshoot DSP HA:

- debug voip dsmp all
- debug voip dsm all
- debug ccsip message
- debug voip ipipgw
- debug voip ipipgw high-availability
- debug voip high-availability all
- debug media resource provisioning all
- debug dsp-resource-manager flex dspfarm
- debug dsp-resource-manager flex function
- debug dsp-resource-manager flex error

How to Configure High Availability

The Cisco Unified Border Element (CUBE) provides high availability (HA) through box-to-box redundancy configurations when implemented on a Cisco Integrated Services Router Generation 2 router (ISR G2) platform. CUBE box-to-box redundancy is achieved using the router-based Hot Standby Routing Protocol (HSRP) router technology.

To configure DSP HA, follow the general configuration procedure for HA at Cisco Unified Border Element High Availability (HA) Using HSRP Configuration Example.

Configuration Examples for DSP HA

**Active Configuration**

```
---
voice-card 0
dsp services dspfarm

dspfarm profile 2 transcode universal
codec g711ulaw
codec g711alaw
codec g729ar8
codec g729abr8
maximum sessions 100
associate application CUBE
```
Standby Configuration

-----------------
voice-card 0
dsp services dspfarm
dspfarm profile 2 transcode universal
codec g711ulaw
codec g711alaw
codec g729ar8
codec g729abr8
maximum sessions 100
associate application CUBE

The following examples show the DSP HA output for the active and standby configurations:

On Active:

Mang-Active#show dspfarm dsp active
SLOT DSP VERSION STATUS CHNL USE TYPE RSC_ID BRIDGE_ID PKTS_TXED PKTS_RXED
0 0 13 39.0.0 UP xcode 1 16558
0 0 3005 3007
0 0 3004 3005

Total number of DSPFARM DSP channel(s) 1

On Standby:

Mang-Standby#show dspfarm dsp active
SLOT DSP VERSION STATUS CHNL USE TYPE RSC_ID BRIDGE_ID PKTS_TXED PKTS_RXED
0 0 13 39.0.0 UP xcode 1 16558
0 0 0
0 0 0

Total number of DSPFARM DSP channel(s) 1
Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices

Stateful switchover provides protection for network edge devices with dual Route Processors (RPs) that represent a single point of failure in the network design, and where an outage might result in loss of service for customers.

- Feature Information for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 671
- Prerequisites for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 672
- Restrictions for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 673
- Information About Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 673

### Feature Information for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices</td>
<td>Cisco IOS XE Release 3.2S</td>
<td>Provides protection for network edge devices with dual Route Processors (RPs) that represent a single point of failure in the network design, and where an outage might result in loss of service for customers.</td>
</tr>
<tr>
<td>Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices</td>
<td>Cisco IOS Release 15.2(3)T</td>
<td>Provides protection for network edge devices with dual Route Processors (RPs) that represent a single point of failure in the network design, and where an outage might result in loss of service for customers.</td>
</tr>
<tr>
<td>Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices</td>
<td>Cisco IOS XE Release 3.8S 15.3(1)T</td>
<td>Provides support for call escalation and de-escalation with stateful switchover.</td>
</tr>
</tbody>
</table>
### Prerequisites for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices

**Cisco Unified Border Element (Enterprise)**

- Cisco IOS XE Release 3.2 or a later release must be installed and running on your Cisco ASR 1000 Series Router.

**Cisco Unified Border Element**

- Cisco IOS Release 15.2(3)T or a later release must be installed and running on your Cisco Unified Border Element.
Restrictions for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices

- Call escalation and de-escalation are not supported in REFER consumption mode.
- Session Description Protocol (SDP) passthru calls are not supported.
- Resource Reservation Protocol (RSVP) is not supported.
- Alternative Network Address Types (ANAT) for IPv4 or IPv6 interworking is not supported.
- SDP passthrough calls are not supported for media forking.
- Media flow-around fork calls are not checkpointed.
- For high availability PROTECTED mode, redundancy group (RG) is not supported on cross-over cable. However, if cross-over cable is used and the connection flaps or if the RG link is connected using a switch and the switch resets, or if there is a switchover, then both the devices will go into PROTECTED mode resulting in no VoIP functionality.

Information About Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices

In specific Cisco networking devices that support dual RPs, stateful switchover takes advantage of Route Processor redundancy to increase network availability. When two route processors (RPs) are installed, one RP acts as the active RP, and the other acts as a backup, or standby RP. Following an initial synchronization between the two processors if the active RP fails, or is manually taken down for maintenance or removed, the standby RP detects the failure and initiates a switchover. During a switchover, the standby RP assumes control of the router, connects with the network interfaces, and activates the local network management interface and system console. Stateful switchover dynamically maintains Route Processor state information between them. The following conditions and restrictions apply to the current implementation of SSO:

- Calls that are handled by nondefault session application (TCL/VXML) will not be checkpointed prebridge.
- Flow-through calls whose state has not been accurately checkpointed will be cleared with media inactivity-based clean up. This condition could occur if active failure happens when:
  - Some check point data has not yet been sent to the standby.
  - The call leg was in the middle of a transaction.
  - Flow around calls whose state has not been accurately checkpointed (due to either of the reasons mentioned above) can be cleared with the clear call voice causecode command.

For more information about the Stateful Switchover feature and for detailed procedures for enabling this feature, see the "Configuring Stateful Switchover" chapter of the Cisco IOS High Availability Configuration Guide, Release 12.2SR.
Call Escalation with Stateful Switchover

The call escalation workflow is as follows:

1. The call starts as an audio call between Phone A (video-capable) and Phone B (only audio-capable) registered to two different Cisco Unified Communications Manager (CUCM) clusters connected using Cisco Unified Border Element (Cisco UBE).

2. The call is then transferred to Phone C, which is a video-capable phone.

3. The media parameters within the reinvite are renegotiated end-to-end.

4. The call is escalated to a video call.

If the Cisco UBE switchover happens at any instance, then audio calls will be preserved before escalation and video calls will be preserved after escalation.

Call De-escalation with Stateful Switchover

The call de-escalation workflow is as follows:

1. The call starts as a video call between Phone A and Phone B registered to two different Cisco Unified Communications Manager (CUCM) clusters connected using Cisco Unified Border Element (Cisco UBE).

2. The call is then transferred to Phone C, which is an audio-only phone.

3. The media parameters within the reinvite are renegotiated end-to-end.

4. The call is de-escalated to an audio-only call.
If the Cisco UBE switchover happens at any instance, then video calls will be preserved before de-escalation and audio calls will be preserved after de-escalation.

Figure 62: Call De-escalation

Media Forking with High Availability

Media forking with high availability is supported on ISR G2, ISR G3 and ASR platforms. When a primary call is connected and a forked call-leg is established on an active Cisco UBE device, both the primary and the forked call-leg will be checkpointed in the standby Cisco UBE device. If the active device goes down, the standby device ensures that the forking call is active and is able to exchange further transactions with the recording server with preserved calls such as hold/resume, transfer, conference, and so on. A recording server is a Session Initiation Protocol (SIP) user agent that archives media for extended durations, providing search and retrieval of the archived media. The recording server is a storage place of the recorded session metadata. The active and standby devices must have the same configurations for checkpointing to happen correctly. The recorder can be configured both ways with a media profile and directly on a media class. The media profile can be associated under media class, and the media class can be applied to the incoming or outgoing dial-peer to start recording.

For more information, see the “Network-based Recording Using Cisco UBE” module in the Cisco Unified Border Element Protocol-Independent Features and Setup Configuration Guide.

High Availability Protected Mode and Box-to-Box Redundancy for ASR

To configure box-to-box high availability (HA) support for ASRs, use the mode rpr command (rpr is route processor redundancy) in redundancy configuration mode.
• Use the same hardware for both the ASR boxes in the active or standby pair to ensure compatibility before and after failover.

• A separate physical interface must be used for checkpointing calls between the active and standby devices.

Self-reload in a voice HA-enabled device helps to recover the box-to-box HA pair from out-of-sync conditions. Instead of self-reload, you can configure the device to transition into protected mode. In protected mode:

• Bulk sync request, call checkpointing, and incoming call processing are disabled.

• The device in protected mode needs to be manually reloaded to come out of this state.

To enabled the protected mode, use the no redundancy-reload command under “voice service voip” configuration mode. The default is redundancy-reload, which reloads control when the redundancy group (RG) fails.

Support for Box-to-Box High Availability with Virtual IP Addresses

The OPTIONS ping with CUBE high availability feature adds the ability to match the incoming dial-peer in the context of the OPTIONS message, allowing response with the virtual IP address shared between the active and standby CUBEs. Box-to-box high availability is supported using virtual IP addresses for the signaling and media, by enhancing the CUBE response to an inbound OPTIONS ping message. This is possible because dial-peer matching of a request URI that does not have a user part is supported.

Important

When OPTIONS Ping SIP Trunk (from CUCM) is configured to CUBE that is running in HA mode, the SIP Trunk goes down whenever the active interface goes down. The SIP Trunk comes back in service, when the OPTIONS Ping next retry happens to CUBE HA node. The default retry time is 60 seconds.

Note

For configuration examples, see the Examples section about configuring interfaces (ISR and ASR) and configuring SIP binding.

Monitoring Call Escalation and De-escalation with Stateful Switchover

Perform this task to monitor calls before and after escalation or de-escalation and before and after stateful switchover on active and standby Cisco UBE devices. The show commands can be entered in any order.

SUMMARY STEPS

1. enable
2. show call active voice compact
3. show call active video compact
4. show call active voice stats
5. show call active video stats
## DETAILED STEPS

### Step 1
**enable**

Enables privileged EXEC mode.

**Example:**

```
Device> enable
```

### Step 2
**show call active voice compact**

Displays a compact version of call information for the voice calls in progress.

**Example:**

```
Device# show call active voice compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 2
   512 ANS T1 g711ulaw VOIP Psipp 9.45.38.39:6016
   513 ORG T1 g711ulaw VOIP P123 10.104.46.222:6000
```

### Step 3
**show call active video compact**

Displays a compact version of call information for the video calls in progress.

**Example:**

```
Device# show call active video compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 2
   512 ANS T19 H263 VOIP-VIDEO Psipp 9.45.38.39:1699
   513 ORG T19 H263 VOIP-VIDEO P123 10.104.46.222:1697
```

### Step 4
**show call active voice stats**

Displays information about digital signal processing (DSP) voice quality metrics.

**Example:**

```
Device# show call active voice stats
dur 00:00:16 tx:2238/85044 rx:1618/61484 dscp:0 media:0 audio tos:0xB8 video tos:0x0
IP 9.45.33.32:58300 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g711ulaw TextRelay: off Transcoded: No
dur 00:00:16 tx:1618/61484 rx:2238/85044 dscp:0 media:0 audio tos:0xB8 video tos:0x0
IP 9.45.33.32:58400 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g711ulaw TextRelay: off Transcoded: No
```

### Step 5
**show call active video stats**

Displays information about digital signal processing (DSP) video quality metrics.

**Example:**

```
Device# show call active video stats
dur 00:00:00 tx:27352/1039376 rx:36487/1386506 dscp:0 media:0 audio tos:0xB8 video tos:0x88
IP 9.45.25.33:1697 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off Transcoded: No
dur 00:00:00 tx:36487/1386506 rx:27352/1039376 dscp:0 media:0 audio tos:0xB8 video tos:0x88
```
Monitoring Media Forking with High Availability

Perform this task to monitor media forking calls with high availability on active and standby Cisco UBE devices. The `show` commands can be entered in any order.

**SUMMARY STEPS**

1. enable
2. show call active voice compact
3. show voip rtp connections
4. show voip recmsp session
5. show voip rtp forking
6. show voip rtp forking

**DETAILED STEPS**

**Step 1**  
**enable**

Enables privileged EXEC mode.

**Example:**

```markdown
Device> enable
```

**Step 2**  
**show call active voice compact**

Displays a compact version of call information for the voice calls in progress. In the output shown, the first and second connections are for the basic call and the third connection is for the forked leg.

**Example:**

```markdown
Device# show call active voice compact
```

```
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 3
4423 ANS T28 g711ulaw VOIP P9538390040 173.39.67.102:22792
4424 ORG T28 g711ulaw VOIP 9.42.30.189:26300
4426 ORG T27 g711ulaw VOIP 10.104.46.201:56356
```

**Step 3**  
**show voip rtp connections**

Displays real-time transport protocol (RTP) named event packets. In the output shown, two additional call legs are shown on the Cisco UBE device. Both the active and standby devices will have the same number of connections.

**Example:**

```markdown
Device# show voip rtp connections
```

```
VoIP RTP active connections :
No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP
1 4439 4440 16646 19022 10.104.46.251 173.39.67.102
2 4440 4439 16648 22950 9.42.30.213 9.42.30.189
```
Found 4 active RTP connections

**Step 4**  show voip recmsp session

Displays active recording Media Service Provider (MSP) session information. In the output shown, the fork leg details and the number of forking calls are displayed. Both the active and standby devices will have the same call information.

**Example:**

```
Device# show voip recmsp session
RECMSP active sessions:
MSP Call-ID AnchorLeg Call-ID ForkedLeg Call-ID
4441 4440 4442
Found 1 active sessions
```

**Step 5**  show voip rtp forking

Displays the RTP media-forking connections. In the output shown, on the active device, packets will be sent.

**Example:**

```
Device# show voip rtp forking
VoIP RTP active forks :
Fork 1
  stream type voice-only (0): count 0
  stream type voice+dtmf (1): count 0
  stream type dtmf-only (2): count 0
  stream type voice-nearend (3): count 1
    remote ip 10.104.46.201, remote port 36840, local port 16650
    codec g711ulaw, logical ssrc 0x53
    packets sent 30788, packets received 0
  stream type voice+dtmf-nearend (4): count 0
  stream type voice-farend (5): count 1
    remote ip 10.104.46.201, remote port 54754, local port 16652
    codec g711ulaw, logical ssrc 0x55
    packets sent 30663, packets received 0
  stream type voice+dtmf-farend (6): count 0
  stream type video (7): count 0
  stream type application (8): count 0
```

**Step 6**  show voip rtp forking

Displays the RTP media-forking connections. In the output shown, on the standby device, packets will not be sent. After the switchover happens, packets will be sent from the new active device.

**Example:**

```
Device# show voip rtp forking
VoIP RTP active forks :
Fork 1
  stream type voice-only (0): count 0
  stream type voice+dtmf (1): count 0
  stream type dtmf-only (2): count 0
  stream type voice-nearend (3): count 1
    remote ip 10.104.46.201, remote port 36840, local port 16650
    codec g711ulaw, logical ssrc 0x53
    packets sent 0, packets received 0
  stream type voice+dtmf-nearend (4): count 0
  stream type voice-farend (5): count 1
    remote ip 10.104.46.201, remote port 54754, local port 16652
    codec g711ulaw, logical ssrc 0x55
```
Verifying the High Availability Protected Mode

Perform this task to verify the configuration for high availability protected mode, assuming the local device is ACTIVE and the peer device went into PROTECTED mode.

**SUMMARY STEPS**

1. enable
2. `show voice high-availability rf-client` (active device)
3. `show voice high-availability rf-client` (standby device)

**DETAILED STEPS**

**Step 1**  
**enable**

*Example:*

```
Router> enable
```

Enables privileged EXEC mode.

**Step 2**  
**show voice high-availability rf-client** (active device)

*Example:*

```
Device# show voice high-availability rf-client
```

FUNCTIONING RF DOMAIN: 0x2

```
-----
RF Domain: 0x0
Voice HA Client Name: VOIP RF CLIENT
Voice HA RF Client ID: 1345
Voice HA RF Client SEQ: 128
My current RF state ACTIVE (13)
Peer current RF state DISABLED (1)

Current VOIP HA state [LOCAL / PEER] :
[(ACTIVE (13) / UNKNOWN (0))]

-----
RF Domain: 0x2 [RG: 1]
Voice HA Client Name: VOIP RG CLIENT
Voice HA RF Client ID: 4054
Voice HA RF Client SEQ: 448
My current RF state ACTIVE (13)
Peer current RF state STANDBY HOT (8)

Current VOIP HA state [LOCAL / PEER] :
```
Step 3 show voice high-availability rf-client (standby device)

Example:

Device# show voice high-availability rf-client

RF Domain: 0x0
Voice HA Client Name: VOIP RF CLIENT
Voice HA RF Client ID: 1345
Voice HA RF Client SEQ: 128
My current RF state ACTIVE (13)
Peer current RF state DISABLED (1)

Current VOIP HA state [LOCAL / PEER]:

-----

RF Domain: 0x2 [RG: 1]
Voice HA Client Name: VOIP RG CLIENT
Voice HA RF Client ID: 4054
Voice HA RF Client SEQ: 448
My current RF state STANDBY HOT (8)
Peer current RF state ACTIVE (13)

Current VOIP HA state [LOCAL / PEER]:

Support for REFER and BYE/Also after Stateful Switch-Over

REFER based supplementary services with high availability is supported post-stateful switchover on CUBE. Support is also provided for SIP-to-SIP BYE/Also calls.

Use the show sip-ua handoff stats command to display the call handoff statistics for calls handed off successfully after switchover. Following are the statistics displayed:

- Total number of calls handed off
- Total number of successful calls handoffs
- Total numbers of unsuccessful call handoffs

The following sample output displays the call handoff statistics:

2951-CUBE#show sip-ua handoff stats
Total Calls Handed Off = 1
Successful Call Hand offs = 1
Un-Successful Call Hand offs = 0
2951-CUBE#

Troubleshooting Tips

Use the following commands to troubleshoot call escalation and de-escalation with stateful switchover:
• debug voip ccapi all
• debug voip ccapi service
• debug voice high-availability all
• debug voip rtp error
• debug voip rtp inout
• debug voip rtp high-availability
• debug voip rtp function
• debug ccsp all

Use the following commands to troubleshoot media forking support on high availability:

• debug ccsp all
• debug voip high-availability all
• debug voip ccapi inout
• debug voip recmsp all

Use the following commands to troubleshoot PROTECTED mode on high availability:

• debug voice high-availability rf
• debug voice high-availability inout
• debug redundancy progression
• debug redundancy application group faults all
• debug redundancy application group protocol all
• debug voip ccapi inout
• debug cch323 session
• debug cch323 function
• debug cch323 error
• debug ccsp all

Use the following debug commands to troubleshoot issues related to handling of REFER based supplementary services:

• debug ccsp verbose
• debug voip application all
• debug voip ccapi all
• debug voice high-availability all
Example: Configuring the Interfaces for ISR-G2 Devices

ISR-G2 (HSRP-based)

```
interface GigabitEthernet0/0/0
ip address 10.10.25.14 255.255.255.0
duplex auto
keepalive
speed auto
standby delay minimum 30 reload 60
standby version 2
standby 0 ip 10.10.25.1
standby 0 preempt
standby 0 priority 50
standby 0 track 2 decrement 10
standby 0 name SB
```

Example: Configuring the Interfaces for ASR Devices

ASR (RG Infra-based)

```
interface GigabitEthernet0/0/0
ip address 10.13.25.190 255.255.0.0
negotiation auto
redundancy rii 1
redundancy group 1 ip 10.13.25.123 exclusive
```

Example: Configuring SIP Binding

```dial-peer voice inbound-dial-peer-tag voip
   session protocol sipv2
   incoming uri from mydesturi
   voice-class sip call-route url
     voice-class sip bind control source-interface GigabitEthernet 0/0/0
!
```

voice class uri mydesturi
    host abc.com
CVP Survivability TCL support with High Availability

Call survivability features are supported in Cisco Unified Border Element (CUBE) high availability mode for all active calls handled by Cisco Voice Portal (CVP).

- Feature Information for CVP Survivability TCL support with High Availability, on page 685
- Prerequisites, on page 686
- Restrictions, on page 686
- Recommendations, on page 686
- CVP Survivability TCL support with High Availability, on page 686
- Configuring CVP Survivability TCL support with High Availability, on page 686

Feature Information for CVP Survivability TCL support with High Availability

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVP Survivability TCL support with High Availability</td>
<td>Cisco IOS 15.6(2)T Cisco IOS XE Denali 16.3.1</td>
<td>This feature enables CUBE support call survivability features in CUBE high availability mode for all active calls handled by CVP.</td>
</tr>
</tbody>
</table>
Prerequisites

• CVP survivability TCL application is configured on incoming dial-peer

Restrictions

• If there is a courtesy callback (CCB) registered with CVP, then post switchover, CCB is not supported.

• Only call survivability TCL script is supported with CUBE high availability. Other TCL based services are not supported.

• Only the active calls will be checkpointed. (Calls which are connected - 200OK / ACK transaction completed). Calls in transition state will not be checkpointed.

Recommendations

• Configure TCP session transport for the SIP trunk between CUBE and CVP.

CVP Survivability TCL support with High Availability

Contact Center Deployments use call survivability TCL script on CUBE to provide basic Call survivability services when downstream CVP nodes are not reachable. From Cisco IOS Release 15.6(2)T onwards, call survivability features are supported in CUBE High Availability mode. Post switchover, all events received on the calls handled by CVP are posted to Call Survivability TCL application for further processing. Thus, call survivability features are supported in CUBE high availability mode for all active calls handled by CVP.


Configuring CVP Survivability TCL support with High Availability

Existing configuration of applying the survivability TCL application on incoming dial-peer is sufficient. No additional configuration required.
PART XV

ICE-Lite Support on CUBE

• ICE-Lite Support on CUBE, on page 689
ICE-Lite Support on CUBE

Interactive Connectivity Establishment (ICE) is a protocol for Network Address Translator (NAT) traversal for UDP-based multimedia sessions established with the offer-answer model. ICE makes use of the Session Traversal Utilities for NAT (STUN) protocol and its extension, Traversal Using Relay NAT (TURN), and can be used by any protocol utilizing the offer-answer model, such as the Session Initiation Protocol (SIP).

The ICE-Lite Support on CUBE feature enables the remote peers of CUBE (that may be behind a NAT and doing ICE) to use the ICE semantics in the session description protocol (SDP) and perform an offer-answer exchange of SDP messages. The CUBE can also interwork with endpoints that support or do not support ICE. ICE agents (devices) that are always attached to the public Internet have a special type of implementation called Lite. CUBE will be in ICE-lite mode only. CUBE supports the ICE-lite feature from Cisco IOS Release 15.5(2)S.

• Feature Information for ICE-Lite Support on CUBE, on page 689
• Restrictions for ICE-lite Support on CUBE, on page 690
• Information About ICE-Lite Support on CUBE, on page 690
• How to Configure ICE-Lite Support on CUBE, on page 692
• Additional References, on page 701

Feature Information for ICE-Lite Support on CUBE

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 78: Feature Information for ICE-Lite Support on CUBE

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICE-Lite Support on CUBE</td>
<td>Cisco IOS 15.5(3)M</td>
<td>The ICE-Lite Support on CUBE feature enables the remote peers of CUBE (that may be behind a NAT and doing ICE) to use the ICE semantics in the session description protocol (SDP) and perform an offer-answer exchange of SDP messages. The CUBE can also interwork with endpoints that support or do not support ICE. ICE agents (devices) that are always attached to the public Internet have a special type of implementation called Lite. CUBE will be in ICE-lite mode only. The following commands were introduced or modified: <code>debug voip icelib</code>, <code>show voip ice global-stats</code>, <code>show voip ice instance call-id call-id</code>, <code>show voip ice summary</code>, and <code>stun usage ice</code></td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.16S</td>
<td></td>
</tr>
</tbody>
</table>

Restrictions for ICE-lite Support on CUBE

The following features are not supported with ICE:

- IPv6
- Alternative Network Address Types (ANAT)
- ANAT-ICE interworking
- Media anti-trombone
- High availability support for video calls
- Codec Transparent
- SDP passthrough
- Media flow-around
- Resource Reservation Protocol (RSVP)
- SIP-to-TDM gateway support
- Media Termination Point (MTP)
- VXML and TCL Scripts

Information About ICE-Lite Support on CUBE

Characteristics

The following are some of the key characteristics of ICE-lite.

- A CLI configured for ICE-lite.
ICE Candidate

To execute ICE, an agent has to identify all of its address candidates. A candidate is a transport address—a combination of IP address and port for a transport protocol, such as UDP. A candidate can be derived from physical or logical network interfaces, or discoverable using STUN and TURN. A viable candidate is a transport address obtained directly from a local interface; such a candidate is called a host candidate. The local interface could be ethernet or WiFi, or it could be one that is obtained through a tunnel mechanism, such as a Virtual Private Network (VPN) or Mobile IP (MIP). In all cases, such a network interface appears to the agent as a local interface from which ports (and thus candidates) can be allocated.

Note
Refer to RFC 5245 for more information about ICE candidates.

ICE Lite

ICE agents (devices) that are always attached to the public Internet have a special type of implementation called Lite. For ICE to be used in a call, both the endpoints (agents) must support it. An ICE agent that supports Lite neither gathers ICE candidates nor triggers ICE connectivity checks; however, the agent responds to connectivity checks and includes only host candidates for any media stream. An ICE agent that supports the lite mode is called an ICE-lite endpoint.

Note
Refer to RFC 5245 for more information about ICE-lite implementation and connectivity checks.

High Availability Support with ICE

High availability (HA) is supported only for audio calls that use ICE. For video calls, as the size of SDP is larger, HA will not work. Some of the design considerations are the following:

• No new checkpoint module for ICE instance.

• ICE instance will be re-created on the standby device from SIP HA re-creation path by using source SDP, destination SDP, and configuration profile.
As no information related to ICE is checkpointed, in the standby device, the ICE valid list (created after connectivity checks are done) is populated from currently used media address.

### How to Configure ICE-Lite Support on CUBE

#### Configuring ICE on the CUBE

ICE lite can be configured under STUN, and the decision to use ICE for a session is based on the offer/answer. This configuration is used for outbound dial-peers of CUBE to decide whether to offer ICE in SDP or not. For an incoming offer, the decision to do ICE is based on what the remote end offers in SDP.

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `voice class stun-usage tag`
4. `stun usage ice lite`
5. `end`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Sets STUN usage global parameters, and enters voice class configuration mode.</td>
</tr>
<tr>
<td><code>voice class stun-usage tag</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# voice class stun-usage 5</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Configures ICE in ICE-Lite mode.</td>
</tr>
<tr>
<td><code>stun usage ice lite</code></td>
<td>• You can use the <code>stun usage ice full</code> command to configure ICE in ICE-Full mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-class)# stun usage ice lite</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><code>end</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-class)# end</code></td>
<td></td>
</tr>
</tbody>
</table>
Verifying ICE-Lite on the CUBE (Success Flow Calls)

The following show commands can be used to verify ICE for success flow calls. The show commands can be entered in any order.

**SUMMARY STEPS**

1. show call active video compact
2. show voip rtp connections
3. show voip ice instance call-id call-id-1
4. show voip ice instance call-id call-id-2
5. show voip ice summary
6. show voip ice global-stats

**DETAILED STEPS**

**Step 1**

show call active video compact

**Example:**

```
Device# show call active video compact
```

<table>
<thead>
<tr>
<th>&lt;callID&gt;</th>
<th>A/O FAX T&lt;sec&gt; Codec type</th>
<th>Peer Address</th>
<th>IP R&lt;ip&gt;:&lt;udp&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 ANS</td>
<td>T189 H264 VOIP-VIDEO</td>
<td>P8181</td>
<td>72.163.212.137:2328</td>
</tr>
<tr>
<td>30 ORG</td>
<td>T189 H264 VOIP-VIDEO</td>
<td>P9191</td>
<td>9.45.46.16:8028</td>
</tr>
<tr>
<td>35 ANS</td>
<td>T189 H264 VOIP-VIDEO</td>
<td>P8181</td>
<td>9.45.46.16:8008</td>
</tr>
<tr>
<td>36 ORG</td>
<td>T189 H264 VOIP-VIDEO</td>
<td>P9191</td>
<td>72.163.212.163:2328</td>
</tr>
</tbody>
</table>

**Step 2**

show voip rtp connections

The following sample output displays the VoIP RTP usage information and RTP active connections.

**Example:**

```
Device# show voip rtp connections
```

**VoIP RTP Port Usage Information:**


```
VoIP RTP Port Usage Information:
```

**VoIP RTP active connections:**

```
VoIP RTP active connections:
No. CallId dstCallLocalRTP RmtRTP LocalIP RemoteIP MPSS
1  25 30 8000 2326 10.104.45.107 72.163.212.137 NO
2  26 31 8002 2328 10.104.45.107 72.163.212.137 NO
3  27 32 8036 2454 10.104.45.107 72.163.212.137 NO
4  28 33 8004 2330 10.104.45.107 72.163.212.137 NO
5  29 34 8038 2332 10.104.45.107 72.163.212.137 NO
6  30 25 8006 8016 9.45.46.16 9.45.46.16 NO
7  31 26 8008 8028 9.45.46.16 9.45.46.16 NO
8  32 27 8010 8030 9.45.46.16 9.45.46.16 NO
9  33 28 8012 8032 9.45.46.16 9.45.46.16 NO
10 34 29 8014 8034 9.45.46.16 9.45.46.16 NO
11 35 36 8016 8006 9.45.46.16 9.45.46.16 NO
```
Verifying ICE-Lite on the CUBE (Success Flow Calls)

Step 3

show voip ice instance call-id call-id-1

The following sample output displays the active ICE sessions on the ICE-full and the ICE-lite legs where there are ICE negotiations.

Example:

Device# show voip ice instance call-id 25

Interactive Connectivity Check (ICE) Instance details:
Call-ID is 25
Instance is 0x7FC617FC0508
Overall ICE-State is COMPLETED
LocalAgent's mode is ICE-CONTROLLED
RemoteAgent's mode is ICE-CONTROLLING
m-line:1
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107 port 8000 type host RemoteIP 72.163.212.137 port 2326 type host
m-line:2
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107 port 8002 type host RemoteIP 72.163.212.137 port 2328 type host
LocalIP 10.104.45.107 port 8003 type host RemoteIP 72.163.212.137 port 2329 type host
m-line:3
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107 port 8036 type host RemoteIP 72.163.212.137 port 2454 type host
m-line:4
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107 port 8004 type host RemoteIP 72.163.212.137 port 2330 type host
LocalIP 10.104.45.107 port 8005 type host RemoteIP 72.163.212.137 port 2331 type host
m-line:5
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107 port 8038 type host RemoteIP 72.163.212.137 port 2332 type host

Total Rx STUN Bind Req 22
Total Tx STUN Bind Succ Resp 22
Total Tx STUN Bind failure resp 0

Step 4

show voip ice instance call-id call-id-2
The following sample output displays the idle ICE sessions on the ICE-lite and the ICE-lite legs where there are no ICE negotiations.

**Example:**

Device# `show voip ice instance call-id 30`

Interactive Connectivity Check(ICE) Instance details:
Call-ID is 30
Instance is 0x7FC617FC03F8
Overall ICE-State is RUNNING
LocalAgent's mode is ICE-CONTROLLED
RemoteAgent's mode is ICE-CONTROLLING
m-line:1
--------
ICE-State: IDLE
No candidate has been nominated

m-line:2
--------
ICE-State: IDLE
No candidate has been nominated

m-line:3
--------
ICE-State: IDLE
No candidate has been nominated

m-line:4
--------
ICE-State: IDLE
No candidate has been nominated

m-line:5
--------
ICE-State: IDLE
No candidate has been nominated

Total Rx STUN Bind Req 0
Total Tx STUN Bind Succ Resp 0
Total Tx STUN Bind failure resp 0

**Step 5**  
**show voip ice summary**

The following sample output displays a summary of active ICE sessions.

**Example:**

Device# `show voip ice summary`

<table>
<thead>
<tr>
<th>CALL-ID</th>
<th>ICE-STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>30</td>
<td>RUNNING</td>
</tr>
<tr>
<td>35</td>
<td>RUNNING</td>
</tr>
<tr>
<td>36</td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>

**Step 6**  
**show voip ice global-stats**

The following sample output displays the global ICE statistics.

**Example:**
ICE-Lite on CUBE (Error Flow Calls)

The following are the show command sample outputs followed by the system logs for error flow calls. The show commands can be entered in any order.

**SUMMARY STEPS**

1. show call active voice compact
2. show voip rtp connections
3. show voip ice instance call-id call-id
4. show voip ice instance call-id call-id
5. show voip ice summary
6. show voip ice global-stats

**DETAILED STEPS**

**Step 1**  
show call active voice compact

**Example:**

Device# show call active video compact

<table>
<thead>
<tr>
<th>CallID</th>
<th>A/O</th>
<th>FAX</th>
<th>Codec</th>
<th>Type</th>
<th>Peer Address</th>
<th>IP</th>
<th>R&lt;ip&gt;:&lt;udp&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>173.39.64.79:7078</td>
<td></td>
</tr>
</tbody>
</table>

**Step 2**  
show voip rtp connections

The following sample output displays the VoIP RTP usage information and RTP active connections.

**Example:**

Device# show voip rtp connections

VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 2

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Min Port</th>
<th>Max Port</th>
<th>Available Ports</th>
<th>Reserved Ports</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media Pool</td>
<td>8000</td>
<td>48198</td>
<td>19999</td>
<td>101</td>
<td>2</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No.</th>
<th>CallId</th>
<th>dstCallLocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
<th>MPSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>57</td>
<td>58</td>
<td>8040</td>
<td>7078</td>
<td>173.39.64.79</td>
<td>NO</td>
</tr>
</tbody>
</table>
Step 3  show voip ice instance call-id call-id

The following sample output displays the ICE sessions.

Example:

Device# show voip ice instance call-id 57

Interactive Connectivity Check(ICE) Instance details:
Call-ID is 57
Instance is 0x7FC617FC03F8
Overall ICE-State is RUNNING
LocalAgent's mode is ICE-CONTROLLED
RemoteAgent's mode is ICE-CONTROLLING
m-line:1
--------
ICE-State: IDLE
No candidate has been nominated

Total Rx STUN Bind Req 2
Total Tx STUN Bind Succ Resp 0
Total Tx STUN Bind failure resp 2

Step 4  show voip ice instance call-id call-id

The following sample output displays the ICE sessions.

Example:

Device# show voip ice instance call-id 58

Interactive Connectivity Check(ICE) Instance details:
Call-ID is 58
Instance is 0x7FC617FC0508
Overall ICE-State is RUNNING
LocalAgent's mode is ICE-CONTROLLED
RemoteAgent's mode is ICE-CONTROLLING
m-line:1
--------
ICE-State: IDLE
No candidate has been nominated

Total Rx STUN Bind Req 2
Total Tx STUN Bind Succ Resp 0
Total Tx STUN Bind failure resp 2

Step 5  show voip ice summary

The following sample output displays a summary of active ICE sessions.

Example:

Device# show voip ice summary

CALL-ID    ICE-STATE
-----------
57         RUNNING
58         RUNNING
Total number of sessions: 2
Step 6  
show voip ice global-stats

The following sample output displays the global ICE statistics.

Example:

Device# show voip ice global-stats

Interactive Connectivity Establishment (ICE) global stats:
Total Rx Stun BindingRequests : 47
Total Tx Stun BindingSuccessResponses: 43
Total Tx Stun BindingErrorResponses : 4

The following are the sys logs for invalid message integrity and for sending ICE-controlled parameter.

Sys Log for invalid message integrity:

004012: *Aug 8 14:25:30.876 IST: %CISCO_STUN-4-INVALID_MESSAGE_INTEGRITY: Invalid Message-Integrity attribute in the received STUN message on UDP IP address 10.104.45.107 port 8040
###STUN Message structure start###
Message Type : STUN_MSG_TYPE_BINDING_REQ
Magic Cookie : 2112A442
Transaction ID : 01C61B24C077331ECD27A5B
Mapped Address : Not Set/Present
User Name : Not Set/Present
Error code not present
Alternate Server : Not Set/Present
Realm : Not Set/Present
nonce : Not Set/Present
Xormapped Address : Not Set/Present
Server : Not Set/Present
ICE Priority : Not Set/Present
ICE Controlled : Not Set/Present
ICE Controlling : Not Set/Present
Cisco-flowdata : cisco-flowdata is not present
Message Integrity : Not Set/Present
Finger Print : Not Set/Present
###STUN Message structure End###
004013: *Aug 8 14:25:30.876 IST:://%CUBE-AREA-4-INVALID_MESSAGE_INTEGRITY: Invalid Message-Integrity attribute in the received STUN message on UDP IP address 10.104.45.107 port 8040


---

ICE-Lite Support on CUBE
STUN Message Length = 44
004036: *Aug 8 14:25:30.876 IST: //57/91300134802E/STUN/Detail/stunSendMsgToNetwork: Message sending from, 10.104.45.107:8040, to 173.39.64.79:7078
004037: *Aug 8 14:25:30.876 IST: //57/91300134802E/STUN/Detail/stunSendMsgToNetwork: Stun Message:
004039: *Aug 8 14:25:30.876 IST: //1/xxxxxxxxxxxx/STUN/Detail/stunSendMsg: ** Sent Stun Packet to Network **
###STUN Message structure start###
Message Type : STUN_MSG_TYPE_BINDING_ERR_RESP
Magic Cookie : 2112A442
Transaction ID : F1CF84958CE76D15C83059D9
Mapped Address : Not Set/Present
User Name : Not Set/Present
Error Code : Number = 400, Reason = Bad Request
Alternate Server : Not Set/Present
Realm : Not Set/Present
nonce : Not Set/Present
Server : Not Set/Present
ICE Priority : Not Set/Present
ICE Controlled : Not Set/Present
ICE Controlling : Not Set/Present
cisco-flowdata : cisco-flowdata is not present
Message Integrity : D0E2E828944BF3D07CC5C06D026D8909B85EF3E9
004040: *Aug 8 14:25:30.876 IST: Finger Print : Not Set/Present
###STUN Message structure End###
Sys Log for sending ICE-controlled parameter instead of ICE-controlling parameter:


Cisco Unified Border Element Configuration Guide
ICE-Lite Support on CUBE

ICE-Lite on CUBE (Error Flow Calls)

Mapped Address : Not Set/Present
User Name : GAah4WY
Error code not present
Alternate Server : Not Set/Present
Realm : Not Set/Present
nonce : Not Set/Present
Xormapped Address : Not Set/Present
Server : Cisco
ICE Priority : 1862270975
ICE Controlled : 11920035603547232620
ICE Controlling : Not Set/Present

Cisco-flowdata :
cisco-flowdata is not present
Message Integrity : 0AF4B8C2378CB90AB08A3806507D766BF5CD1DD

###STUN Message structure End###

###STUN Message structure Start###

Message Type : STUN_MSG_TYPE_BINDING_ERR_RESP
Magic Cookie : 2112A442
Transaction ID : F1CF84958CE76D15C83059D9
Mapped Address : Not Set/Present
User Name : Not Set/Present
Troubleshooting ICE-Lite Support on CUBE

You can use the following `debug` commands to troubleshoot the ICE-lite support on CUBE feature. Use these commands to enable ICE debugs for each call.

- `debug voip icelib all`
- `debug voip icelib default`
- `debug voip icelib detail`
- `debug voip icelib error`
- `debug voip icelib event`
- `debug voip icelib inout`
- `debug voip stun all`
- `debug voip stun default`
- `debug voip stun detail`
- `debug voip stun error`
- `debug voip stun event`
- `debug voip stun inout`
- `debug voip stun message`
- `debug voip stun packet`
<table>
<thead>
<tr>
<th>Standard/RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 5766</td>
<td>Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)</td>
</tr>
<tr>
<td>RFC 5768</td>
<td>Indicating Support for Interactive Connectivity Establishment (ICE) in the Session Initiation Protocol (SIP)</td>
</tr>
<tr>
<td>RFC 3840</td>
<td>Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)</td>
</tr>
<tr>
<td>RFC 7584</td>
<td>Session Traversal Utilities for NAT (STUN) Message Handling for SIP Back-to-Back User Agents (B2BUAs)</td>
</tr>
</tbody>
</table>

**Technical Assistance**

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
<tr>
<td>To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds.</td>
<td></td>
</tr>
<tr>
<td>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td></td>
</tr>
</tbody>
</table>
PART XVI

SIP Protocol Handling

• Mid-call Signaling Consumption, on page 705
• Early Dialog UPDATE Block, on page 715
• Consumption of Forked 18x Responses with SDP During Early Dialog, on page 721
• Support for Pass-Through of Unsupported Content Types in SIP INFO Messages, on page 727
• Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element, on page 729
Mid-call Signaling Consumption

The Cisco Unified Border Element BE Mid-call Signaling support aims to reduce the interoperability issues that arise due to consuming mid-call RE-INVITES/UPDATES.

Mid-call Re-INVITEs/UPDATEs can be consumed in the following ways:

- Mid-call Signaling Passthrough - Media Change
- Mid-call Signaling Block
- Mid-call Signaling Codec Preservation

Note

This feature should be used as a last resort only when there is no other option in CUBE. This is because configuring this feature can break video-related features. For Delay-offer Re-INVITE, the configured codec will be passed as an offer in 200 message to change the codec, the transcoder is added in the answer.

Feature Information for Mid-call Signaling

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
### Prerequisites

- Enable CUBE application on a device
- Cisco IOS Release 15.2(1)T or later, or Cisco IOS-XE Release 15.2(2)S or later must be installed.
- `supplementary-service media-renegotiate` must be configured in global voice service voip mode.

#### Table 79: Feature Information for Mid-call Signaling

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mid-call Re-INVITE Consumption</td>
<td>Cisco IOS 15.2(1)T</td>
<td>The Mid-call Re-INVITE consumption feature consumes mid-call Re-INVITEs from CUBE and helps to avoid interoperability issues because of these re-invites.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.6S</td>
<td>The following commands were introduced or modified: <code>midcall-signaling</code>.</td>
</tr>
<tr>
<td>Mid-call Codec Preservation</td>
<td>Cisco IOS 15.3(2)T</td>
<td>The Mid-call Codec Preservation feature helps to disables codec negotiation in the middle of a call and preserves the codec negotiated before the call.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.9S</td>
<td>The following commands were introduced or modified: <code>midcall-signaling preserve-codec</code>, <code>voice-class sip midcall-signaling preserve-codec</code>.</td>
</tr>
<tr>
<td>Mid-call Re-INVITE Consumption</td>
<td>Cisco IOS 15.5(3)M</td>
<td>Mid-call signaling Re-INVITE consumption is enhanced to support:</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.16S</td>
<td>• Re-INVITE based call transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Call transfer with REFER Consume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Normalization of call hold in a call set-up</td>
</tr>
</tbody>
</table>
Mid-call Signaling Passthrough - Media Change

Passthrough media change method optimizes or consumes mid-call, media-related signaling within the call. Mid-call signaling changes will be passed through only when bidirectional media like T.38 or video is added. The command `midcall-signaling passthru media-change` needs to be configured to enable passthrough media change.

Restrictions for Mid-Call Signaling Passthrough - Media Change

- SIP-H.323 calls are not supported.
- TDM Gateways are not supported.
- Session Description Protocol (SDP) - passthrough is not supported.
- When `codec T` is configured, the offer from CUBE has only audio codecs, and so the video codecs are not consumed.
- Re-invites are not consumed if media flow-around is configured.
- Re-invites are not consumed if media anti-tromboning is configured.
- De-escalation re-invites are consumed. So, one call leg might be de-escalated to audio only while the other call leg continues to support audio and video.
- In the RTP-RTP scenario, re-invites from the SRTP leg are not consumed if the `srtp fallback` is configured.
- Re-invites with media direction changes are consumed.
- Video transcoding is not supported.
- Secure Real-time Protocol - Real-time Protocol (SRTP-RTP) supplementary services are not supported if re-invites come from SRTP.
- Multicast Music On Hold (MMOH) is not supported.
- When the `midcall-signaling passthru media-change` command is configured and high-density transcoder is enabled, there might be some impact on Digital Signal Processing (DSP) resources as the transcoder might be used for all the calls.
- Session timer is handled leg by leg whenever this feature is configured.
- More than two m-lines in the SDP is not supported.
- Alternative Network Address Types (ANAT) is not supported.
- Video calls and Application streams are not supported when mid-call signaling block is configured.

Behavior of Mid-call Re-INVITE Consumption

- If mid-call signaling block is enabled on either of call-legs, video parameters and application streams are not negotiated, and are rejected in the answer.
- When flow around and offer-all is configured, CUBE performs codec renegotiation even if mid-call signaling block is configured globally.
• Below behavior is for refer consume scenario:
  • REFER consume is supported for blind, alert and consult call transfers.
  • Existing codecs or DTMF is used for local bridging of new call legs. No Re-INVITE or UPDATE is sent for media re-negotiation after REFER.
  • Call gets dropped when DSP is required but not available.
  • A call can be escalated to video only if transferee and transfer-to dial-peers do not have mid-call signaling block configured.
  • Video calls are de-escalated if mid-call signaling block configuration on transfer-to dial-peer.
  • For Re-INVITE based call-transfer involving Cisco Unified Communications Manager, all Re-INVITE are locally answered and transcoder is invoked if negotiated codecs are different than the codecs before call-transfer.

• The following table provides the details of the behavior when the initial call is establish without 'sendrecv' parameter, that means, the initial call is established with 'sendonly', 'recvonly' or 'inactive'.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>If an Offer is received with 'sendonly' and mid-call block is configured on any or both call legs</td>
<td>Offer is sent with 'sendrecv'.</td>
</tr>
<tr>
<td>If an Answer is received with 'sendonly' and the peer leg supports mid-call signaling</td>
<td>Answer is sent with 'sendonly'. Resume transaction is end-to-end.</td>
</tr>
<tr>
<td>If an Answer is received with 'sendonly' and the peer leg does not support mid-call signaling</td>
<td>Answer is sent with 'sendrecv'. Resume transaction is consumed.</td>
</tr>
<tr>
<td>If Offer as well as Answer is received with 'sendonly' and Offering leg does not support mid-call signaling</td>
<td>Answer is sent with 'recvonly'. Resume from Offering leg is end-to-end. Resume from answering leg is consumed.</td>
</tr>
<tr>
<td>If Offer as well as Answer is received with 'sendonly' and Answering leg does not support mid-call signaling</td>
<td>Answer is sent with 'inactive'. Resume from Offering leg is consumed. Resume from answering leg is end-to-end.</td>
</tr>
<tr>
<td>If Offer as well as Answer is received with 'sendonly' and both legs do not support mid-call signaling</td>
<td>Answer is sent with 'recvonly'. Resume transaction is consumed.</td>
</tr>
</tbody>
</table>

**Configuring Passthrough of Mid-call Signalling**

Perform this task to configure passthrough of mid-call signaling (as Re-invites) only when bidirectional media is added.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. Configure passthrough of mid-call signaling changes only when bidirectional media is added.
   • In Global VoIP SIP configuration mode
midcall-signaling passthru media-change
  • In dial-peer configuration mode
voice-class sip mid-call signaling passthru media-change

4. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>Configure passthrough of mid-call signaling changes only when bidirectional media is added.</td>
<td></td>
</tr>
<tr>
<td>• In Global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>midcall-signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td>• In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>voice-class sip mid-call signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Re-Invites are passed through only when bidirectional media is added.</td>
</tr>
<tr>
<td>In Global VoIP SIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# midcall-signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>In Dial-peer configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 2 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip mid-call signaling passthru media-change</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

Example Configuring Passthrough SIP Messages at Dial Peer Level

The following example shows how to passthrough SIP messages at the dial peer Level:

dial-peer voice 600 voip
destination-pattern 2222222222
session protocol sipv2
session target ipv4:9.45.38.39:9001
voice-class sip mid-call signaling passthru media-change
Example Configuring Passthrough SIP Messages at the Global Level

The following example shows how to passthrough SIP messages at the global level:

```bash
Device(config)# voice service voip
Device(conf-voi-serv)# no ip address trusted authenticate
Device(conf-voi-serv)# allow-connections sip to sip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# midcall-signaling passthru media-change
```

Mid-call Signaling Block

The Block method blocks all mid-call media-related signaling to the specific SIP trunk. The command `midcall-signaling block` needs to be configured to enable this behavior. Video escalation and T.38 call flow are rejected when the `midcall-signaling block` command is configured. This command should be configured only when basic call is the focus and mid-call can be consumed.

Restrictions for Mid-Call Signaling Block

- SIP-H.323 calls are not supported.
- TDM Gateways are not supported.
- Session Description Protocol (SDP) passthrough is not supported
- Video calls and Application streams are not supported.
- When media flow-around is configured, Mid-call INVITE is rejected with 488 error message.
- Re-invites are not consumed if media anti-tromboning is configured.
- SRTP-RTP supplementary services are not supported.
- Multicast Music On Hold (MMOH) is not supported.
- When the `midcall-signaling passthru media-change` command is configured and high-density transcoding is enabled, there might be some impact on Digital Signal Processing (DSP) resources as the transcoder might be used for all the calls.
- Session timer is handled leg by leg whenever this feature is configured.
- More than two m-lines in the SDP is not supported.
- Alternative Network Address Types (ANAT) is not supported.
• When mid-call signaling block is configured, you can either configure REFER consume or enable TCL script. Mid-call signaling block is not supported if both REFER consume and TCL script are enabled. We also recommend not to configure supplementary-service media-renegotiate command.

### Blocking Mid-Call Signaling

Perform this task to block mid-call signaling:

**SUMMARY STEPS**

1. enable
2. configure terminal
3. Configure blocking of mid-call signaling changes:
   - In Global VoIP SIP configuration mode
     - midcall-signaling block
   - In dial-peer configuration mode
     - voice-class sip mid-call signaling block
4. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>Configure blocking of mid-call signaling changes:</td>
<td>Mid-call signaling is always blocked.</td>
</tr>
<tr>
<td></td>
<td>• In Global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>midcall-signaling block</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>voice-class sip mid-call signaling block</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Global VoIP SIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-serv-sip)# midcall-signaling block</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Dial-peer configuration mode:</td>
<td></td>
</tr>
</tbody>
</table>
Example Blocking SIP Messages at Dial Peer Level

dial-peer voice 107 voip
destination-pattern 74000
session protocol sipv2
session target ipv4:9.45.36.9
incoming called-number 84000
voice-class codec 1 offer-all

dial-peer voice 110 voip
destination-pattern 84000
session protocol sipv2
session target ipv4:9.45.35.2
incoming called-number 74000
voice-class codec 1 offer-all
voice-class sip mid-call signaling block

Example: Blocking SIP Messages at the Global Level

The following example shows how to block SIP messages at the global Level

Device(config)#voice service voip
Device(config-voi-serv)#no ip address trusted authenticate
Device(config-voi-serv)#allow-connections sip to sip
Device(config-voi-serv)#sip
Device(config-serv-sip)#midcall-signaling block

Mid Call Codec Preservation

Mid call codec preservation defines whether a codec can be negotiated after a call has been initiated. You can enable or disable codec negotiation in the middle of a call.

Configuring Mid Call Codec Preservation

This task disables codec negotiation in the middle of a call and preserves the codec negotiated before the call.

SUMMARY STEPS

1. enable
2. configure terminal
3. Enter one of the following to disable midcall codec renegotiation:
   • In Global VoIP SIP configuration mode
midcall-signaling preserve-codec
• In dial-peer configuration mode

voice-class sip midcall-signaling preserve-codec

4. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> Enter one of the following to disable midcall codec renegotiation:</td>
<td>Disables codec negotiation in the middle of a call and preserves the codec negotiated before the call.</td>
</tr>
<tr>
<td>• In Global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>midcall-signaling preserve-codec</td>
<td></td>
</tr>
<tr>
<td>• In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>voice-class sip midcall-signaling preserve-codec</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# midcall-signaling preserve-codec</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# dial-peer voice 10 voip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-dial-peer)# voice-class sip</td>
<td></td>
</tr>
<tr>
<td>midcall-signaling preserve-codec</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(conf-serv-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>

**Example: Configuring Mid Call Codec Preservation at the Dial Peer Level**

dial-peer voice 107 voip
destination-pattern 74000
session protocol sipv2
session target ipv4:9.45.36.9
incoming called-number 84000
Example: Configuring Mid Call Codec Preservation at the Global Level

Device(config)# voice service voip
Device(conf-voi-serv)# no ip address trusted authenticate
Device(conf-voi-serv)# allow-connections sip to sip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# midcall-signaling preserve-codec
Early Dialog UPDATE Block

This feature enables CUBE to consume UPDATE requests with SDP, received during an early dialog. UPDATE requests are blocked at CUBE and are not passed through from one leg to the other leg.

If the UPDATE request contains changes in caller-ID, transcoder insertion or deletion, or video escalation or de-escalation, then, CUBE can renegotiate the capabilities by sending a DO invite after the call is established.

- Feature Information for Early Dialog UPDATE Block, on page 715
- Prerequisites, on page 716
- Restrictions, on page 716
- Information about Early Dialog UPDATE Block, on page 716
- Configuring Early Dialog UPDATE Block, on page 717
- Configuring Early Dialog UPDATE Block Renegotiate, on page 718
- Troubleshooting Tips, on page 719

Feature Information for Early Dialog UPDATE Block

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Dialog UPDATE Block</td>
<td>Cisco IOS 15.5(3)M</td>
<td>This feature allows CUBE to consume the UPDATE requests with SDP received during an early dialog.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.16S</td>
<td>The following command is introduced: early-media update block.</td>
</tr>
</tbody>
</table>

Table 80: Feature Information for Mid-call Signaling
Prerequisites

• rel1xx require "100rel" command needs to be configured in global voice service voip sip configuration mode.

Restrictions

• Switch over to fax calls are not supported.
• Session Description Protocol (SDP) passthrough is not supported.
• Alternative Network Address Types (ANAT) is not supported.

Information about Early Dialog UPDATE Block

UPDATE request with SDP received during an early dialog is consumed by CUBE and hence is not passed from one leg to the other leg. This feature can be configured only for the UPDATE requests with SDP.

To pass through the information in UPDATE requests containing changes in caller-ID, transcoder insertion or deletion, or video escalation or de-escalation, CUBE can renegotiate the capabilities by sending a DO invite after the call is established. Thus both the user agents are synchronized and this helps in effective utilization of resources.

Renegotiation can be configured only for the UPDATE requests containing the following changes:

• Caller ID
• Transcoder insertion or deletion
• Video escalation or de-escalation

'Early Dialog UPDATE Block' and 'Early Dialog UPDATE Block Renegotiate' can be configured at dial peer level and also at global voice service voip sip configuration level.

Important Characteristics of Early Dialog UPDATE Block

The following are a few important characteristics of Early Dialog UPDATE block:

• If vcc codec is offered by the user agent through an UPDATE, first codec common between received and configured in in-leg at dial-peer is sent in 200OK.
• UPDATE request is consumed, if an UPDATE request with SDP is received after CUBE sends out 200 OK for an INVITE and before ACK is received.
• A 200 Ok is sent for an UPDATE even if there is no transcoder available ONLY for DTMF (rtp-nte to inband). CUBE falls back to inband.
• If Transcoder is unavailable, only the first codec received in the UPDATE request is sent in 200OK.
• CUBE sends 488 message if transcoder is required but unavailable for codec changes, SRTP-RTP inter-working, and transcoding.
• When a video escalation is received via UPDATE, CUBE sends 200 OK with video port as ZERO. No Video RTP or DP sessions are created.

• When a video de-escalation is received via UPDATE, CUBE sends 200 ok with video port as ZERO. RTP or DP sessions for video are made as INACTIVE instead of deleting. So, effectively there will be four RTP connections or 2 DP connections present with remote video port as ZERO.

• Early-media UPDATE renegotiation takes precedence over DO-EO renegotiation.

• If an early dialog UPDATE is received from one leg to change the caller-ID and the other leg supports UPDATE method, CUBE sends across the caller-id UPDATE to other side and there wont be any renegotiation.

• If Re-Invite is received before triggering DO invite, then DO is not triggered.

• If no update-callerid command is enabled and UPDATE request contains only caller-ID changes, then re-negotiation does not happen for any early dialog caller-ID changes. If UPDATE request contains transcoder changes or video escalation or de-escalation, re-negotiation happens even if no update-callerid command is enabled.

• If mid-call signaling block is configured, DO invite is not triggered.

### Configuring Early Dialog UPDATE Block

Configuring early dialog UPDATE Block enables CUBE to block all early dialog UPDATE requests from passing through to the user agents.

#### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Enter one of the following commands to block early dialog UPDATE requests:
   - In the dial-peer configuration mode
     ```
     voice-class sip early-media update block
     ```
   - In the global VoIP SIP configuration mode
     ```
     early media update block
     ```
4. **end**

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Enter one of the following commands to block early dialog UPDATE requests:</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• In the dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>voice-class sip early-media update block</td>
<td></td>
</tr>
<tr>
<td>• In the global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>early media update block</td>
<td></td>
</tr>
</tbody>
</table>

### Example:

In dial-peer configuration mode

```
! Applying Early Dialog UPDATE block to one dial peer only
Device (config)# dial-peer voice 10 voip
Device (config-dial-peer)# Voice-class sip early-media update block
Device (config-dial-peer)# end
```

**Example:**

In global VoIP SIP configuration mode

```
! Applying Early Dialog UPDATE block globally
Device (config)# voice service voip
Device (config-voi-serv)# sip
Device (config-voi-sip)# early media update block
Device (config-voi-sip)# end
```

### Configuring Early Dialog UPDATE Block Renegotiate

Configuring Early Dialog UPDATE Block Renegotiate enables CUBE to renegotiate the call if UPDATE request with SDP contains changes caller-ID, transcoder insertion or deletion, or video escalation or deletion. CUBE renegotiates by sending a DO invite after the call is established.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Enter one of the following commands:
   - In the dial-peer configuration mode
     
     voice-class sip early-media update block re-negotiate
   - In the global VoIP configuration mode
     
     early media update block re-negotiate
4. **end**

| Step 4 | end | Exits VoIP SIP configuration mode and enters privileged EXEC mode. |
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  • Enter your password if prompted. |
| **Step 2** configure terminal | Enters global configuration mode. |
| **Step 3** Enter one of the following commands: |
  - In the dial-peer configuration mode  
    ```
    voice-class sip early-media update block re-negotiate
    ```
  - In the global VoIP configuration mode  
    ```
    early media update block re-negotiate
    ```
| Example: |
| In dial-peer configuration mode |

! Applying Early Dialog UPDATE block re-negotiate to one dial peer only  
Device (config)# dial-peer voice 10 voip  
Device (config-dial-peer)# voice-class sip  
Device (config-dial-peer)# early-media update block re-negotiate  
Device (config-dial-peer)# end

| Example: |
| In global VoIP SIP configuration mode |

! Applying Early Dialog UPDATE block re-negotiate globally  
Device (config)# voice service voip  
Device (config-voi-serv)# sip  
Device (config-voi-sip)# early media update block re-negotiate  
Device (config-voi-sip)# end

| **Step 4** end | Exits VoIP SIP configuration mode and enters privileged EXEC mode. |

## Troubleshooting Tips

Use the following command for debugging information:

- `debug cc sip all`
- `debug voip ccapi inout`
- `show voip rtp connections`
Consumption of Forked 18x Responses with SDP During Early Dialog

The Cisco Unified Border Element supports consumption of forked 18x responses with SDP, under certain conditions during an early dialog, to reduce the interoperability issues that arise due to signaling forking.

When CUBE receives forked 18x responses with SDP, the media negotiation by default is end-to-end. This means that CUBE has to send an UPDATE with SDP on the inbound leg to renegotiate the new media offer. Under certain conditions, the inbound leg may not be able to support sending UPDATE messages with SDP for media renegotiation. This results in CUBE consuming the forked 18x responses with SDP and may result in DSP resources being used for media interworking. Media parameters such as direction change, and call escalation or de-escalation is not propagated end-to-end. If required, these media changes can be renegotiated end-to-end, after the calls are connected, using a DO re-INVITE.

- Feature Information for Consumption of Multiple Forked 18x Responses with SDP During Early Dialog, on page 721
- Prerequisites, on page 722
- Restrictions, on page 722
- Information About Consumption of Forked 18x Responses with SDP During Early Dialog, on page 722
- Configuring Consumption of Forked 18x Responses with SDP During Early Dialog, on page 723
- Configuring Consumption of Forked 18x Responses with SDP During Early Dialog Renegotiate, on page 724
- Troubleshooting Tips, on page 726

Feature Information for Consumption of Multiple Forked 18x Responses with SDP During Early Dialog

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 81: Feature Information for Consumption of Multiple Forked 18x Responses with SDP During Early Dialog

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Forked 18x Responses with SDP during Early Dialog</td>
<td>Cisco IOS 15.6(3)M</td>
<td>This feature allows CUBE to consume multiple forked 18x responses with SDP received during an early dialog.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE Denali 16.3.1</td>
<td></td>
</tr>
</tbody>
</table>

### Prerequisites

- Re-negotiation is triggered only if the renegotiate `early media update block re-negotiate` CLI is enabled

### Restrictions

The following features or call-flows are not supported:

- SIP Delayed-Offer to Delayed-Offer call flows
- Session Description Protocol (SDP) passthrough mode
- Secure Real-Time Transport Protocol (SRTP) passthrough calls
- Alternative Network Address Types (ANAT)
- Media flow-around
- Media anti-trombone
- Early-dialog UPDATE block

### Information About Consumption of Forked 18x Responses with SDP During Early Dialog

Forked 18x responses for INVITE requests with SDP during early dialog will be consumed by CUBE to reduce interoperability issues between user agents.

### Characteristics of Forked 18x Responses with SDP during Early Dialog

- If PRACK or UPDATE is not supported on the inbound leg, by default, CUBE consumes the forked 18x responses
- If PRACK or UPDATE is not supported and CUBE has to initiate renegotiation after call connect, then the `early media update block re-negotiate` CLI must be enabled
- When PRACK and UPDATE are supported on the inbound leg and CUBE has to consume the forked 18x responses, the `early media update block` CLI must be enabled
If PRACK and UPDATE are supported and CUBE has to consume the forked 18x responses and initiate renegotiation after call connect, then the **early media update block renegotiate** CLI must be enabled.

If mid-call signaling block or mid-call signaling passthrough media changes are configured, DO invite is not triggered.

---

**Note**

CUBE utilizes the EARLY UPDATE BLOCK functionality to configure the forked 18x responses with SDP during early dialog. The **early media update block** command is used to consume the forked 18x responses and the **early media update block renegotiate** command is used to renegotiate the forked 18x responses after the call connect.

Renegotiation (when enabled via configuration) is triggered for the forked 18x responses containing the following changes:

- DSP Transcoder insertion
- Video escalation or de-escalation
- Media directional changes

---

**Note**

It is recommended to configure the **early media update block re-negotiate** command whenever there are transcoding, DTMF interworking, or video changes.

---

### Configuring Consumption of Forked 18x Responses with SDP During Early Dialog

Perform the following procedure to enable CUBE to block all early dialog forked 18x requests from passing through to the user agents.

**SUMMARY STEPS**

1. **enable**
2. **configure terminal**
3. Enter one of the following commands to block the forked 18x responses with SDP during early dialog:
   - In the dial-peer configuration mode
     ```
     voice-class sip early-media update block
     ```
   - In the global VoIP SIP configuration mode
     ```
     early media update block
     ```
4. **end**
## Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enter one of the following commands to block the forked 18x responses with SDP during early dialog:</td>
</tr>
<tr>
<td></td>
<td>• In the dial-peer configuration mode</td>
</tr>
<tr>
<td></td>
<td>voice-class sip early-media update block</td>
</tr>
<tr>
<td></td>
<td>• In the global VoIP SIP configuration mode</td>
</tr>
<tr>
<td></td>
<td>early media update block</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>In dial-peer configuration mode</td>
</tr>
<tr>
<td>!Applying Early Dialog UPDATE block to one dial peer only Device (config)# dial-peer voice 10 voip Device (config-dial-peer)# voice-class sip early-media update block Device (config-dial-peer)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>In global VoIP SIP configuration mode</td>
</tr>
<tr>
<td>! Applying Early Dialog UPDATE block globally Device (config)# voice service voip Device (config-voi-serv)# sip Device (config-voi-sip)# early media update block Device (config-voi-sip)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> end</td>
<td>Exits VoIP SIP configuration mode and enters privileged EXEC mode.</td>
</tr>
</tbody>
</table>

### Configuring Consumption of Forked 18x Responses with SDP During Early Dialog Renegotiate

Perform the following procedure to enable CUBE to renegotiate forked 18x calls with SDP during early dialog after consumption of these forked 18x responses. CUBE renegotiates by sending a DO invite after the call is established.

### Summary Steps

1. enable
2. **configure terminal**

3. Enter one of the following commands:
   - In the dial-peer configuration mode
     
     ```
     voice-class sip early-media update block re-negotiate
     ```
   - In the global VoIP configuration mode
     
     ```
     early media update block re-negotiate
     ```

4. **end**

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
</tr>
<tr>
<td></td>
<td>Displays privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
</tr>
<tr>
<td></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enter one of the following commands:</td>
</tr>
<tr>
<td></td>
<td>• In the dial-peer configuration mode</td>
</tr>
<tr>
<td></td>
<td>voice-class sip early-media update block re-negotiate</td>
</tr>
<tr>
<td></td>
<td>• In the global VoIP configuration mode</td>
</tr>
<tr>
<td></td>
<td>early media update block re-negotiate</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>In dial-peer configuration mode</td>
</tr>
<tr>
<td></td>
<td>! Applying Early Dialog UPDATE block re-negotiate to one dial peer only</td>
</tr>
<tr>
<td></td>
<td>Device (config)# dial-peer voice 10 voip</td>
</tr>
<tr>
<td></td>
<td>Device (config-dial-peer)# voice-class sip</td>
</tr>
<tr>
<td></td>
<td>early-media update block re-negotiate</td>
</tr>
<tr>
<td></td>
<td>Device (config-dial-peer)# end</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>In global VoIP SIP configuration mode</td>
</tr>
<tr>
<td></td>
<td>! Applying Early Dialog UPDATE block re-negotiate globally</td>
</tr>
<tr>
<td></td>
<td>Device(config)# voice service voip</td>
</tr>
<tr>
<td></td>
<td>Device (config-voi-serv)# sip</td>
</tr>
<tr>
<td></td>
<td>Device (config-voi-sip)# early media update block re-negotiate</td>
</tr>
<tr>
<td></td>
<td>Device (config-voi-sip)# end</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>end</td>
</tr>
<tr>
<td></td>
<td>Exits VoIP SIP configuration mode and enters privileged EXEC mode.</td>
</tr>
</tbody>
</table>
Troubleshooting Tips

Use the following command for debugging information:

- debug ccsip verbose
- show voip rtp connections detail
- show call active voice brief
- show dspfarm dsp active
- show voice dsmp stream brief
- show platform hardware qfp active feature sbc global
Support for Pass-Through of Unsupported Content Types in SIP INFO Messages

This feature allows the CUBE to pass-through all unsupported content types in a SIP INFO message.

- Feature Information for Support for Pass-Through of Unsupported Content Types in SIP INFO Messages, on page 727
- Prerequisites, on page 727
- Information About Pass-Through of Unsupported Content Types in SIP INFO Messages, on page 728

Feature Information for Support for Pass-Through of Unsupported Content Types in SIP INFO Messages

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for pass-through of unsupported content types in SIP INFO messages</td>
<td>Cisco IOS 15.5(3)M, Cisco IOS XE 3.16S</td>
<td>This feature allows CUBE to pass-through SIP INFO methods or request message types with unsupported content types. Media negotiation and media exchange is completely end-to-end.</td>
</tr>
</tbody>
</table>

Prerequisites

You must enable the pass-thru content unsupp command to pass-through all unsupported content types in a SIP INFO message. There is no additional configuration task required for this feature.
Information About Pass-Through of Unsupported Content Types in SIP INFO Messages

The Support for Pass-Through of Unsupported Content Types in SIP INFO Messages feature allows the CUBE to pass-through all unsupported content types in a SIP INFO message.

Upon receipt of a SIP INFO message with unsupported content type, CUBE triggers a SIP INFO message on the outgoing peer call leg. The response received for this SIP INFO message is triggered on the incoming peer call leg and information flows end-to-end. Prior to releases 15.5(3)M and 3.16S, on reception of SIP INFO message with unsupported content type, CUBE will respond with the “415 Unsupported Media Type” error response.

Supported content types include the following:

- application/sdp
- application/qsig
- application/media-control+xml
- application/x-q931
- application/gtd
- application/simple-message-summary
- application/kpml-response+xml
- application/dtmf-relay
- application/broadsoft
- message/sipfrag
- audio/telephone-event
- multipart/mixed
Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element

The figure below shows a typical network topology where the Cisco Unified Border Element is configured to route messages between a call manager system (such as the Cisco Unified Call Manager) and a Next Generation Network (NGN).

Figure 63: Cisco Unified Border Element and Next Generation Topology

Devices that connect to an NGN must comply with the User-Network Interface (UNI) specification. The Cisco Unified Border Element supports the NGN UNI specification and can be configured to interconnect NGN with other call manager systems, such as the Cisco Unified Call Manager.

The Cisco Unified Border Element supports the following:

- the use of P-Preferred Identity (PPID), P-Asserted Identity (PAID), Privacy, P-Called Party Identity (PCPID), in INVITE messages
- the translation of PAID headers to PPID headers and vice versa
- the translation of RPID headers to PAID or PPID headers and vice versa
- the configuration and/or pass through of privacy header values
- the use of the PCPID header to route INVITE messages
- the use of multiple PAURI headers in the response messages (200 OK) it receives to REGISTER messages

P-Preferred Identity and P-Asserted Identity Headers

NGN servers use the PPID header to identify the preferred number that the caller wants to use. The PPID is part of INVITE messages sent to the NGN. When the NGN receives the PPID, it authorizes the value, generates a PAID based on the preferred number, and inserts it into the outgoing INVITE message towards the called party.
However, some call manager systems, such as Cisco Unified Call Manager 5.0, use the Remote-Party Identity (RPID) value to send calling party information. Therefore, the Cisco Unified Border Element must support building the PPID value for an outgoing INVITE message to the NGN, using the RPID value or the From: value received in the incoming INVITE message. Similarly, CUBE supports building the RPID and/or From: header values for an outgoing INVITE message to the call manager, using the PAID value received in the incoming INVITE message from the NGN.

In non-NGN systems, the Cisco Unified Border Element can be configured to translate between PPID and PAID values, and between From: or RPID values and PAID/PPID values, at global and dial-peer levels.

In configurations where all relevant servers support the PPID or PAID headers, the Cisco Unified Border Element can be configured to transparently pass the header.

If the NGN sets the From: value to anonymous, the PAID is the only value that identifies the caller.

The table below describes the types of INVITE message header translations supported by the Cisco Unified Border Element. It also includes information on the configuration commands to use to configure P-header translations.

The table below shows the P-header translation configuration settings only. In addition to configuring these settings, you must configure other system settings (such as the session protocol).

### Table 82: P-header Configuration Settings

<table>
<thead>
<tr>
<th>Incoming Header</th>
<th>Outgoing Header</th>
<th>Configuration Notes</th>
</tr>
</thead>
</table>
| From:            | RPID            | To enable the translation to RPID headers in the outgoing header, use the **remote-party-id** command in SIP user-agent configuration mode. For example: Router(config-sip-ua)# remote-party-id
If both, **remote-party-id** and **asserted-id** commands are configured, then the **asserted-id** command takes precedence over the **remote-part-id** command. |
| PPID             | PAID            | To enable the translation to PAID privacy headers in the outgoing header at a global level, use the **asserted-id pai** command in voice service VoIP SIP configuration mode. For example: Router(conf-serv-sip)# asserted-id pai
To enable the translation to PAID privacy headers in the outgoing header on a specific dial peer, use the **voice-class sip asserted-id pai** command in dial peer voice configuration mode. For example: Router(config-dial-peer)# voice-class sip asserted-id pai |
| PPID             | RPID            | To enable the translation to RPID headers in the outgoing header, use the **remote-party-id** command in SIP user-agent configuration mode. For example: Router(config-sip-ua)# remote-party-id
This is the default system behavior. |
<table>
<thead>
<tr>
<th>Incoming Header</th>
<th>Outgoing Header</th>
<th>Configuration Notes</th>
</tr>
</thead>
</table>
| PAID            | PPID            | To enable the translation to PPIID privacy headers in the outgoing header at a global level, use the `asserted-id ppi` command in voice service VoIP SIP configuration mode. For example: `Router(conf-serv-sip)# asserted-id ppi`  
To enable the translation to PPIID privacy headers in the outgoing header on a specific dial peer, use the `voice-class sip asserted-id ppi` command in dial peer voice configuration mode. For example: `Router(config-dial-peer)# voice-class sip asserted-id ppi` |
| PAID            | RPID            | To enable the translation to RPID headers in the outgoing header, use the `remote-party-id` command in SIP user-agent configuration mode. For example: `Router(config-sip-ua)# remote-party-id`  
This is the default system behavior. |
| RPID            | PPID            | To enable the translation to PPIID privacy headers in the outgoing header at a global level, use the `asserted-id ppi` command in voice service VoIP SIP configuration mode. For example: `Router(conf-serv-sip)# asserted-id ppi`  
To enable the translation to PPIID privacy headers in the outgoing header on a specific dial peer, use the `voice-class sip asserted-id ppi` command in dial peer voice configuration mode. For example: `Router(config-dial-peer)# voice-class sip asserted-id ppi` |
| RPID            | PAID            | To enable the translation to PAID privacy headers in the outgoing header at a global level, use the `asserted-id pai` command in voice service VoIP SIP configuration mode. For example: `Router(conf-serv-sip)# asserted-id pai`  
To enable the translation to PAID privacy headers in the outgoing header on a specific dial peer, use the `voice-class sip asserted-id pai` command in dial peer voice configuration mode. For example: `Router(config-dial-peer)# voice-class sip asserted-id pai` |
| RPID            | From:           | By default, the translation to RPID headers is enabled and the system translates PPIID headers in incoming messages to RPID headers in the outgoing messages. To disable the default behavior and enable the translation from PPIID to From: headers, use the `no remote-party-id` command in SIP user-agent configuration mode. For example: `Router(config-sip-ua)# no remote-party-id` |

Privacy functions are not initialized on Unified Border Element without configuring `asserted-id pai` or `asserted-id ppi`. Ensure that you configure `asserted-id pai` or `asserted-id ppi` to support privacy functions on Unified Border Element.

The CUBE can be configured to transparently pass the PAID and PPIID headers in the incoming and outgoing Session Initiation Protocol (SIP) requests or response messages from end-to-end.

- Requests include: INVITEs and UPDATEs
- Responses include: 18x and 200 OK
The priority of P-headers are in the following order: PAID, PPID, and RPID.

Table 83: PAID and PPID header configuration settings for mid-call requests and responses

<table>
<thead>
<tr>
<th>Incoming Header</th>
<th>Outgoing Header</th>
<th>Configuration Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAID</td>
<td>PPID</td>
<td>To enable the translation to PPID headers in the outgoing header at a global level, use the <code>asserted-id ppi</code> command in voice service VoIP SIP configuration mode. For example: <code>Router(conf-serv-sip)# asserted-id ppi</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPID</td>
<td>PPID</td>
<td>To enable the translation to PPID headers in the outgoing header at a global level, use the <code>asserted-id ppi</code> command in voice service VoIP SIP configuration mode. For example: <code>Router(conf-serv-sip)# asserted-id ppi</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>To enable the translation to PPID headers in the outgoing header on a specific dial peer, use the <code>voice-class sip asserted-id ppi</code> command in dial peer voice configuration mode. For example: <code>Router(config-dial-peer)# voice-class sip asserted-id ppi</code></td>
</tr>
<tr>
<td>Incoming Header</td>
<td>Outgoing Header</td>
<td>Configuration Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>PPID</td>
<td>PPID</td>
<td>To enable the translation to PPID headers in the outgoing header at a global level, use the <code>asserted-id ppi</code> command in voice service VoIP SIP configuration mode. To enable the translation to PPID headers in the outgoing header on a specific dial peer, use the <code>voice-class sip asserted-id ppi</code> command in dial peer voice configuration mode. For example: <code>Router(config-dial-peer)# voice-class sip asserted-id ppi</code></td>
</tr>
<tr>
<td>PAID</td>
<td>PAID</td>
<td>To enable the translation to PAID headers in the outgoing header at a global level, use the <code>asserted-id pai</code> command in voice service VoIP SIP configuration mode. To enable the translation to PAID headers in the outgoing header on a specific dial peer, use the <code>voice-class sip asserted-id pai</code> command in dial peer voice configuration mode. For example: <code>Router(config-dial-peer)# voice-class sip asserted-id pai</code></td>
</tr>
<tr>
<td>RPID</td>
<td>PAID</td>
<td>To enable the translation to PAID headers in the outgoing header at a global level, use the <code>asserted-id pai</code> command in voice service VoIP SIP configuration mode. To enable the translation to PAID headers in the outgoing header on a specific dial peer, use the <code>voice-class sip asserted-id pai</code> command in dial peer voice configuration mode.</td>
</tr>
<tr>
<td>Incoming Header</td>
<td>Outgoing Header</td>
<td>Configuration Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>PPID</td>
<td>PAID</td>
<td>To enable the translation to PAID headers in the outgoing header at a global level, use the <code>asserted-id pai</code> command in voice service VoIP SIP configuration mode. To enable the translation to PAID headers in the outgoing header on a specific dial peer, use the <code>voice-class sip asserted-id pai</code> command in dial peer voice configuration mode.</td>
</tr>
<tr>
<td>PAID</td>
<td>RPID</td>
<td>To enable the translation to RPID headers in the outgoing header, use the <code>remote-party-id</code> command in SIP user-agent configuration mode. For example: <code>Router(config-sip-ua)# remote-party-id</code>. <strong>Note</strong> PAID and PPID headers are not configured in this case.</td>
</tr>
<tr>
<td>RPID</td>
<td>RPID</td>
<td>To enable the translation to RPID headers in the outgoing header, use the <code>remote-party-id</code> command in SIP user-agent configuration mode. For example: <code>Router(config-sip-ua)# remote-party-id</code>. <strong>Note</strong> PAID and PPID headers are not configured in this case.</td>
</tr>
<tr>
<td>PPID</td>
<td>RPID</td>
<td>To enable the translation to RPID headers in the outgoing header, use the <code>remote-party-id</code> command in SIP user-agent configuration mode. For example: <code>Router(config-sip-ua)# remote-party-id</code></td>
</tr>
<tr>
<td>FROM</td>
<td>FROM</td>
<td>No configuration required except for the <code>remote-party-id</code> header.</td>
</tr>
<tr>
<td>Incoming Header</td>
<td>Outgoing Header</td>
<td>Configuration Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>FROM</td>
<td>RPID</td>
<td>To enable the translation to RPID headers in the outgoing header, use the <code>remote-party-id</code> command in SIP user-agent configuration mode. For example: <code>Router(config-sip-ua)# remote-party-id</code></td>
</tr>
<tr>
<td>PAID</td>
<td>PAID</td>
<td>Enables PPID headers on the incoming dial-peer and PAID headers on the outgoing dial-peer.</td>
</tr>
<tr>
<td>RPID</td>
<td>PAID</td>
<td>Enables PPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PPID</td>
<td>PAID</td>
<td>Enables PPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PAID</td>
<td>PAID</td>
<td>Enables RPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>RPID</td>
<td>PAID</td>
<td>Enables RPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PPID</td>
<td>PAID</td>
<td>Enables RPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PAID</td>
<td>PPID</td>
<td>Enables PAID headers on incoming dial-peer and PPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>RPID</td>
<td>PPID</td>
<td>Enables PAID headers on incoming dial-peer and PPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PPID</td>
<td>PPID</td>
<td>Enables PAID headers on incoming dial-peer and PPID on outgoing dial-peer.</td>
</tr>
<tr>
<td>PAID</td>
<td>PPID</td>
<td>Enables RPID headers on incoming dial-peer and PPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>RPID</td>
<td>PPID</td>
<td>Enables RPID headers on incoming dial-peer and PPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>Incoming Header</td>
<td>Outgoing Header</td>
<td>Configuration Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>PPID</td>
<td>PPID</td>
<td>Enables RPID headers on incoming dial-peer and PPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PAID</td>
<td>RPID</td>
<td>Enables PPID headers on incoming dial-peer and RPID headers on outgoing dial-peer. Note PAID headers will be given priority and RPID headers will be created using the PAID header information.</td>
</tr>
<tr>
<td>RPID</td>
<td>RPID</td>
<td>Enables PPID headers on incoming dial-peer and RPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PPID</td>
<td>RPID</td>
<td>Enables PPID headers on incoming dial-peer and RPID headers on outgoing dial-peer. Note PPID headers will be given priority and RPID headers will be created using the PPID header information.</td>
</tr>
<tr>
<td>PAID</td>
<td>RPID</td>
<td>Enables PAID headers on incoming dial-peer and RPID headers on outgoing dial-peer. Note PAID headers will be given priority and RPID headers will be created using the PAID header information.</td>
</tr>
<tr>
<td>RPID</td>
<td>RPID</td>
<td>Enables PAID headers on incoming dial-peer and RPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td>PPID</td>
<td>RPID</td>
<td>Enables PAID headers on incoming dial-peer and RPID headers on outgoing dial-peer. Note PPID headers will be given priority and RPID headers will be created using the PPID header information.</td>
</tr>
</tbody>
</table>
Privacy

If the user is subscribed to a privacy service, the Cisco Unified Border Element can support privacy using one of the following methods:

- Using prefixes

The NGN dial plan can specify prefixes to enable privacy settings. For example, the dial plan may specify that if the caller dials a prefix of 184, the calling number is not sent to the called party.

The dia pl plan may also specify that the caller can choose to send the calling number to the called party by dialing a prefix of 186. Here, the Cisco Unified Border Element transparently passes the prefix as part of the called number in the INVITE message.

The actual prefixes for the network are specified in the dial plan for the NGN, and can vary from one NGN to another.

- Using the Privacy header

If the Privacy header is set to None, the calling number is delivered to the called party. If the Privacy header is set to a Privacy:id value, the calling number is not delivered to the called party.

- Using Privacy values from the peer call leg

If the incoming INVITE has a Privacy header or a RPID with privacy on, the outgoing INVITE can be set to Privacy: id. This behavior is enabled by configuring `privacy pstn` command globally or `voice-class sip privacy pstn` command on the selected dial-per.

Incoming INVITE can have multiple privacy header values, id, user, session, and so on. Configure the `privacy-policy passthru` command globally or `voice-class sip privacy-policy passthru` command to transparently pass across these multiple privacy header values.

Some NGN servers require a Privacy header to be sent even though privacy is not required. In this case the Privacy header must be set to none. The Cisco Unified Border Element can add a privacy header with the value None while forwarding the outgoing INVITE to NGN. Configure the `privacy-policy send-always` globally or `voice-class sip privacy-policy send-always` command in dial-peer to enable this behavior.

If the user is not subscribed to a privacy service, the Cisco Unified Border Element can be configured with no Privacy settings.

Note
For the Privacy functions to work as intended, the command `asserted-id {pai|ppi}` must be configured.

P-Called Party Identity

The Cisco Unified Border Element can be configured to use the PCPID header in an incoming INVITE message to route the call, and to use the PCPID value to set the To: value of outgoing INVITE messages.

The PCPID header is part of the INVITE messages sent by the NGN, and is used by Third Generation Partnership Project (3GPP) networks. The Cisco Unified Border Element uses the PCPID from incoming INVITE messages (from the NGN) to route calls to the Cisco Unified Call Manager.

Note
The PCPID header supports the use of E.164 numbers only.
P-Associated URI

The Cisco Unified Border Element supports the use of PAURI headers sent as part of the registration process. After the Cisco Unified Border Element sends REGISTER messages using the configured E.164 number, it receives a 200 OK message with one or more PAURIs. The number in the first PAURI (if present) must match the contract number. The Cisco Unified Border Element supports a maximum of six PAURIs for each registration.

Note
The Cisco Unified Border Element performs the validation process only when a PAURI is present in the 200 OK response.

The registration validation process works as follows:

- The Cisco Unified Border Element receives a REGISTER response message that includes PAURI headers that include the contract number and up to five secondary numbers.

- The Cisco Unified Border Element validates the contract number against the E.164 number that it is registering:
  - If the values match, the Cisco Unified Border Element completes the registration process and stores the PAURI value. This allows administration tools to view or retrieve the PAURI if needed.
  - If the values do not match, the Cisco Unified Border Element unregisters and then reregisters the contract number. The Cisco Unified Border Element performs this step until the values match.

Random Contact Support

The Cisco Unified Border Element can use random-contact information in REGISTER and INVITE messages so that user information is not revealed in the contact header.

To provide random contact support, the Cisco Unified Border Element performs SIP registration based on the random-contact value. The Cisco Unified Border Element then populates outgoing INVITE requests with the random-contact value and validates the association between the called number and the random value in the Request-URI of the incoming INVITE. The Cisco Unified Border Element routes calls based on the PCPID, instead of the Request-URI which contains the random value used in contact header of the REGISTER message.

The default contact header in REGISTER messages is the calling number. The Cisco Unified Border Element can generate a string of 32 random alphanumeric characters to replace the calling number in the REGISTER contact header. A different random character string is generated for each pilot or contract number being registered. All subsequent registration requests will use the same random character string.

The Cisco Unified Border Element uses the random character string in the contact header for INVITE messages that it forwards to the NGN. The NGN sends INVITE messages to the Cisco Unified Border Element with random-contact information in the Request URI. For example: INVITE sip:FefhH3zIHHe9i8ImcGjDD1PEc5XiFy51G@10.12.1.46:5060.

The Cisco Unified Border Element will not use the To: value of the incoming INVITE message to route the call because it might not identify the correct user agent if supplementary services are invoked. Therefore, the Cisco Unified Border Element must use the PCPID to route the call to the Cisco Unified Call Manager. You can configure routing based on the PCPID at global and dial-peer levels.

- Feature Information for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element, on page 739
Feature Information for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 84: Feature Information for PAID and PPID Headers on Cisco Unified Border Element (CUBE)

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| PAID and PPID Headers in mid-call re-INVITE and UPDATE request and responses on Cisco Unified Border Element | Cisco IOS 15.5(3)MCisco IOS XE 3.16S | This feature enables CUBE platforms to support:  
  • P-Preferred Identity (PPID) and P-Asserted Identity (PAID) in mid-call re-INVITE messages and responses from end-to-end.  
  • P-Preferred Identity (PPID) and P-Asserted Identity (PAID) in mid-call UPDATE messages and responses from end-to-end.  
  • Configuration and/or pass through of PAID and PPID header values. |

Feature History Table entry for the Cisco Unified Border Element and Cisco Unified Border Element (Enterprise).
### Table 85: Feature Information for PAID, PPID, Privacy, PCPID, and PAURI Headers on CUBE

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| PAID, PPID, Privacy, PCPID, and PAURI Headers on the Cisco Unified Border Element | 12.4(22)YB 15.0(1)M Cisco IOS XE Release 3.1S | This feature enables CUBE platforms to support:  
• P-Preferred Identity (PPID), P-Asserted Identity (PAID), Privacy, P-Called Party Identity (PCPID), in INVITE messages  
• Translation of PAID headers to PPID headers and vice versa  
• Translation of From: or RPID headers to PAID or PPID headers and vice versa  
• Configuration and/or pass through of privacy header values  
• PCPID header to route INVITE messages  
• Multiple PAURI headers in the response messages (200 OK) it receives to REGISTER messages  
• P-Preferred Identity and P-Asserted Identity Headers  

The following commands were introduced: `call-route p-called-party-id`, `privacy-policy`, `random-contact`, `random-request-uri validate`, ` voice-class sip call-route p-called-party-id`, `voice-class sip privacy-policy`, `voice-class sip random-contact`, and `voice-class sip random-request-uri validate`. |

### Prerequisites for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element

**Cisco Unified Border Element**  
• Cisco IOS Release 12.4(22)YB or a later release must be installed and running on your Cisco Unified Border Element.  

**Cisco Unified Border Element (Enterprise)**  
• Cisco IOS XE Release 3.1S or a later release must be installed and running on your Cisco ASR 1000 Series Router.
Restrictions for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element

- To enable random-contact support, you must configure the Cisco Unified Border Element to support SIP registration with random-contact information. In addition, you must configure random-contact support in VoIP voice-service configuration mode or on the dial peer.

- If random-contact support is configured for SIP registration only, the system generates the random-contact information, includes it in the SIP REGISTER message, but does not include it in the SIP INVITE message.

- If random-contact support is configured in VoIP voice-service configuration mode or on the dial peer only, no random contact is sent in either the SIP REGISTER or INVITE message.

- Passing of “+” is not supported with PAID PPID Privacy PCPID and PAURI Headers.

Configuring P-Header and Random-Contact Support on the Cisco Unified Border Element

To enable random contact support you must configure the Cisco Unified Border Element to support Session Initiation Protocol (SIP) registration with random-contact information, as described in this section.

To enable the Cisco Unified Border Element to use the PCPID header in an incoming INVITE message to route the call, and to use the PCPID value to set the To: value of outgoing INVITE messages, you must configure P-Header support as described in this section.

Configuring P-Header Translation on a Cisco Unified Border Element

To configure P-Header translations on a Cisco Unified Border Element, perform the steps in this section.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. asserted-id header-type
6. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>
### Configuring P-Header Translation on an Individual Dial Peer

To configure P-Header translation on an individual dial peer, perform the steps in this section.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `dial-peer voice tag voip`
4. `voice-class sip asserted-id header-type`
5. `exit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>
## Purpose

### Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router&gt; enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>- Enter your password if prompted.</td>
</tr>
</tbody>
</table>

### Step 2

**Example:**

```bash
Router# configure terminal
```

### Step 3

**Example:**

```bash
Router(config)# dial-peer voice 2611 voip
```

### Step 4

**Example:**

```bash
Router(config-dial-peer)# voice-class sip asserted-id header-type
```

### Step 5

**Example:**

```bash
Router(config-dial-peer)# exit
```

---

## Configuring P-Called-Party-Id Support on a Cisco Unified Border Element

To configure P-Called-Party-Id support on a Cisco Unified Border Element, perform the steps in this section.

### SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. call-route p-called-party-id
6. random-request-uri validate
7. exit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>- Enter your password if prompted.</td>
</tr>
</tbody>
</table>

```bash
Router> enable
```
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong> configure terminal <strong>Example:</strong> Router# configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip <strong>Example:</strong> Router(config)# voice service voip</td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong> sip <strong>Example:</strong> Router(config-voi-serv)# sip</td>
<td>Enters voice service VoIP SIP configuration mode.</td>
</tr>
<tr>
<td><strong>Step 5</strong> call-route p-called-party-id <strong>Example:</strong> Router(config-serv-sip)# call-route p-called-party-id</td>
<td>Enables the routing of calls based on the PCPID header.</td>
</tr>
<tr>
<td><strong>Step 6</strong> random-request-uri validate <strong>Example:</strong> Router(config-serv-sip)# random-request-uri validate</td>
<td>Enables the validation of the random string in the Request URI of the incoming INVITE message.</td>
</tr>
<tr>
<td><strong>Step 7</strong> exit <strong>Example:</strong> Router(config-serv-sip)# exit</td>
<td>Exits the current mode.</td>
</tr>
</tbody>
</table>

### Configuring P-Called-Party-Id Support on an Individual Dial Peer

To configure P-Called-Party-Id support on an individual dial peer, perform the steps in this section.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice tag voip
4. voice-class sip call-route p-called-party-id
5. voice-class sip random-request-uri validate
6. exit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | enable | Enables privileged EXEC mode.  
Example:  
Router> enable |  
- Enter your password if prompted. |
| Step 2 | configure terminal | Enters global configuration mode.  
Example:  
Router# configure terminal |
| Step 3 | dial-peer voice tag voip | Defines the dial peer, specifies the method of voice encapsulation, and enters dial peer voice configuration mode.  
Example:  
Router(config)# dial-peer voice 2611 voip |
| Step 4 | voice-class sip call-route p-called-party-id | Enables the routing of calls based on the PCPID header on this dial peer.  
Example:  
Router(config-dial-peer)# voice-class sip call-route p-called-party-id |
| Step 5 | voice-class sip random-request-uri validate | Enables the validation of the random string in the Request URI of the incoming INVITE message on this dial peer.  
Example:  
Router(config-dial-peer)# voice-class sip random-request-uri validate |
| Step 6 | exit | Exits the current mode.  
Example:  
Router(config-dial-peer)# exit |

---

## Configuring Privacy Support on a Cisco Unified Border Element

To configure privacy support on a Cisco Unified Border Element, perform the steps in this section.

### SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. privacy privacy-option
6. privacy-policy privacy-policy-option
7. exit
### Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  Example:  
  Router> enable |
| **Step 2** configure terminal | Enters global configuration mode.  
  Example:  
  Router# configure terminal |
| **Step 3** voice service voip | Enters VoIP voice-service configuration mode.  
  Example:  
  Router(config)# voice service voip |
| **Step 4** sip | Enters voice service VoIP SIP configuration mode.  
  Example:  
  Router(conf-voi-serv)# sip |
| **Step 5** privacy privacy-option | Enables the privacy settings for the header.  
  Example:  
  Router(conf-serv-sip)# privacy id |
| **Step 6** privacy-policy privacy-policy-option | Specifies the privacy policy to use when passing the privacy header from one SIP leg to the next.  
  Example:  
  Router(conf-serv-sip)# privacy-policy passthru |
| **Step 7** exit | Exits the current mode.  
  Example:  
  Router(conf-serv-sip)# exit |

### Configuring Privacy Support on an Individual Dial Peer

To configure privacy support on an individual dial peer, perform the steps in this section.

### Summary Steps

1. enable
2. configure terminal
3. dial-peer voice tag voip
4. voice-class sip privacy privacy-option
5. `voice-class sip privacy-policy privacy-policy-option`
6. `exit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>enable</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;<code>Router&gt; enable</code></td>
</tr>
<tr>
<td></td>
<td>Enables privileged EXEC mode.&lt;br&gt;• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>configure terminal</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;<code>Router# configure terminal</code></td>
</tr>
<tr>
<td></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>dial-peer voice tag voip</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;<code>Router(config)# dial-peer voice 2611 voip</code></td>
</tr>
<tr>
<td></td>
<td>Defines the dial peer, specifies the method of voice encapsulation, and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>voice-class sip privacy privacy-option</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;<code>Router(config-dial-peer)# voice-class sip privacy id</code></td>
</tr>
<tr>
<td></td>
<td>Enables the privacy settings for the header on this dial peer.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>voice-class sip privacy-policy privacy-policy-option</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;<code>Router(config-dial-peer)# voice-class sip privacy-policy passthru</code></td>
</tr>
<tr>
<td></td>
<td>Specifies the privacy policy to use when passing the privacy header from one SIP leg to the next, on this dial peer.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>exit</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;<code>Router(config-dial-peer)# exit</code></td>
</tr>
<tr>
<td></td>
<td>Exits the current mode.</td>
</tr>
</tbody>
</table>

### Configuring Random-Contact Support on a Cisco Unified Border Element

To configure random-contact support on a Cisco Unified Border Element, perform the steps in this section.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sip-ua**
4. **credentials username username password password realm domain-name**
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
</tbody>
</table>
| **Example:**  
  Router> enable  |  • Enter your password if prompted.  |
| **Step 2** configure terminal  | Enters global configuration mode.  |
| **Example:**  
  Router# configure terminal  |  |
| **Step 3** sip-ua  | Enters SIP user-agent configuration mode.  |
| **Example:**  
  Router(config)# sip-ua  |  |
| **Step 4** credentials username  
  password  
  realm  
  domain-name  | Sends a SIP registration message from the Cisco Unified Border Element.  |
| **Example:**  
  Router(config-sip-ua)# credentials username 123456  
  password cisco  
  realm cisco  |  |
| **Step 5** registrar ipv4:  
  destination-address  
  random-contact  
  expires expiry  | Enables the SIP gateways to register E.164 numbers on behalf of analog telephone voice ports (FXS), IP phone virtual voice ports (EFXS), and Skinny Client Control Protocol (SCCP) phones with an external SIP proxy or SIP registrar.  |
| **Example:**  
  Router(config-sip-ua)# registrar ipv4:10.1.2.2  
  random-contact expires 200  |  • The `random-contact` keyword configures the Cisco Unified Border Element to send the random string from the REGISTER message to the registrar.  |
| **Step 6** exit  | Exits the current mode.  |
| **Example:**  
  Router(config-sip-ua)# exit  |  |
| **Step 7** voice service  
  voip  | Enters VoIP voice-service configuration mode.  |
| **Example:**  
  Router(config-sip-ua)# voice service voip  |  |
### Command or Action | Purpose
--- | ---
Router(config)# voice service voip | Enters voice service VoIP SIP configuration mode.

**Step 8**

**sip**
**Example:**
Router(conf-voi-serv)# sip

**Step 9**

**random-contact**
**Example:**
Router(conf-serv-sip)# random-contact

**Step 10**

**exit**
**Example:**
Router(conf-serv-sip)# exit

---

**Configuring Random-Contact Support for an Individual Dial Peer**

To configure random-contact support for an individual dial peer, perform the steps in this section.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. sip-ua
4. credentials username username password password realm domain-name
5. registrar ipv4: destination-address random-contact expires expiry
6. exit
7. dial-peer voice tag voip
8. voice-class sip random-contact
9. exit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**
  enable
  **Example:**
  Router> enable | Enables privileged EXEC mode.
  • Enter your password if prompted. |
| **Step 2**
  configure terminal
  **Example:**
  Router# configure terminal | Enters global configuration mode. |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td>sip-ua Example: Router(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Sends a SIP registration message from the Cisco Unified Border Element.</td>
</tr>
<tr>
<td>credentials username <strong>username</strong> password <strong>password</strong> realm domain-name Example: Router(config-sip-ua)# credentials username 123456 password cisco realm cisco</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Enables the SIP gateways to register E.164 numbers on behalf of FXS, EFXS, and SCCP phones with an external SIP proxy or SIP registrar.</td>
</tr>
<tr>
<td>registrar ipv4: destination-address <strong>random-contact</strong> expires expiry Example: Router(config-sip-ua)# registrar ipv4:10.1.2.2 random-contact expires 200</td>
<td>* The <strong>random-contact</strong> keyword configures the Cisco Unified Border Element to send the random string from the REGISTER message to the registrar.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>exit Example: Router(config-sip-ua)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Defines the dial peer, specifies the method of voice encapsulation, and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>dial-peer voice <strong>tag</strong> voip Example: Router(config)# dial-peer voice 2611 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Enables random-contact support on this dial peer.</td>
</tr>
<tr>
<td>voice-class sip random-contact Example: Router(config-dial-peer)# voice-class sip random-contact</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>exit Example: Router(config-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>
PART XVII

SIP Supplementary Services

• Dynamic Refer Handling, on page 753
• Cause Code Mapping, on page 759
CHAPTER 62

Dynamic Refer Handling

When a dial-peer match occurs, CUBE passes the REFER message from an in leg to an out leg. Also, the host part of the Refer-to header is modified with the IP address.

The Dynamic REFER handling feature provides configurations to pass across or consume the REFER message. When an endpoint invokes a supplementary service such as a call transfer, the endpoint generates and sends an in-dialog REFER request towards the Cisco UBE. If the REFER message is consumed, an INVITE is sent towards refer-to dial-peer.

- Feature Information for Dynamic REFER Handling, on page 753
- Prerequisites, on page 754
- Restrictions, on page 754
- Configuring REFER Passthrough with Unmodified Refer-to, on page 754
- Configuring REFER Consumption, on page 756
- Troubleshooting Tips, on page 758

Feature Information for Dynamic REFER Handling

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFER Consume (Enhancements)</td>
<td>IOS 15.5(1)T</td>
<td>REFER Consume (Enhancements) provides additional configurations to conditionally forward the REFER message. The following commands were introduced: refer consume.</td>
</tr>
<tr>
<td></td>
<td>IOS XE 3.14.0 S</td>
<td></td>
</tr>
</tbody>
</table>
Prerequisites

- Transcoding configuration is required on the CUBE for midcall transcoder insertion, deletion, or modification during call transfers.

Restrictions

- Only Session Initiation Protocol (SIP)-to-SIP call transfers are supported.
- Call escalation and de-escalation are not supported.
- Video transcoding is not supported.
- Session Description Protocol (SDP) pass-through is not supported.
- In REFER consume scenario, if TCL script is enabled, then supplementary-service media-renegotiate command should not be configured.

Configuring REFER Passthrough with Unmodified Refer-to

This task configures the passthrough of REFER message from the in leg to the out leg on a dial-peer match. A REFER is sent towards inbound dial peer. This task also ensures that the host part of the Refer-to header is unmodified and not changed to the IP address during passthrough.

<table>
<thead>
<tr>
<th>supplementary service refer</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>REFER is passed through from the in leg to the out leg</td>
</tr>
<tr>
<td>no</td>
<td>INVITE is sent towards refer-to-dial-peer</td>
</tr>
</tbody>
</table>

Note

This configuration in this task can be overridden by the refer consume command. Refer to the Configuring REFER Consumption task for more information.
### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. Configure REFER passthrough:
   - `supplementary-service sip refer` in global VoIP configuration mode.
   - `supplementary-service sip refer` in dial-peer configuration mode.
4. (Optional) Configure unmodified Refer-to:
   - `refer-to-passing` in Global VoIP SIP configuration mode.
   - `voice-class sip refer-to-passing [system]` in dial-peer configuration mode.
5. `end`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device&gt; enable</code></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(config)# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>Configure REFER passthrough:</td>
<td>Configures REFER passthrough. A REFER is sent towards the inbound dial peer</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(config)# voice service voip Device(conf-voi-serv)# supplementary-service sip refer</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Global VoIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device{config)# dial-peer voice 22 voip Device{config-dial-peer)# supplementary-service sip refer</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>(Optional) Configure unmodified Refer-to:</td>
<td>Ensures that the refer-to header is unmodified and not changed to the IP address during passthrough</td>
</tr>
<tr>
<td></td>
<td>• <code>refer-to-passing</code> in Global VoIP SIP configuration mode.</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring REFER Consumption

This task configures the consumption of REFER message on a dial-peer match. An INVITE is sent towards the Refer-to dial peer.

**Table 87: Configurations for REFER Consumption**

<table>
<thead>
<tr>
<th>supplementary service refer</th>
<th>refer consume</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>REFER is sent towards inbound dial-peer</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>INVITE is sent towards refer-to dial-peer</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>INVITE is sent towards refer-to dial-peer</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>INVITE is sent towards refer-to dial-peer</td>
</tr>
</tbody>
</table>

**SUMMARY STEPS**

1. **enable**
2. **configure terminal**
3. Enter one of the following:
   - **no supplementary-service sip refer** in global VoIP configuration mode.
   - **no supplementary-service sip refer** in dial-peer configuration mode.
4. **refer consume** in global VoIP configuration mode.
5. **supplementary-service media-renegotiate** in global VoIP configuration mode.
6. **end**
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> Enter one of the following:</td>
<td>Configures REFER consumption. An INVITE is sent towards the Refer-to dial peer.</td>
</tr>
<tr>
<td>• no supplementary-service sip refer in global VoIP configuration mode.</td>
<td></td>
</tr>
<tr>
<td>• no supplementary-service sip refer in dial-peer configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In global VoIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# no supplementary-service sip refer</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In dial-peer configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 22 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# no supplementary-service sip refer</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> refer consume in global VoIP configuration mode.</td>
<td>Configures REFER consumption.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In dial-peer configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 22 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# refer consume</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> (Optional) supplementary-service media-renegotiate in global VoIP configuration mode.</td>
<td>Enables end-to-end media renegotiation during the call transfer in REFER consumption mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In global VoIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# supplementary-service media-renegotiate</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Exits to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
Troubleshooting Tips

Use any of the following debug commands:

- `debug ccsip all`
- `debug voip ccap inout`
- `debug sccp messages`
- `debug voip application supplementary-service`
- `debug voip application state`
- `debug voip application media negotiation`
CHAPTER 63

Cause Code Mapping

With the Cause Code Mapping feature, the NOTIFY message sent by CUBE to a Customer Voice Portal (CVP) contains a proper reason for failure of call transfer based on the information received by CUBE from the caller instead of a 503 Service Unavailable message for all scenarios.

- Feature Information for Cause Code Mapping, on page 759
- Cause Code Mapping, on page 760
- Configuring Cause Code Mapping, on page 761
- Verifying Cause Code Mapping, on page 762

Feature Information for Cause Code Mapping

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 88: Feature Information for Cause Code Mapping

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cause Code Mapping</td>
<td>Cisco IOS 15.5(1)T</td>
<td>With the Cause Code Mapping feature, the NOTIFY message sent by CUBE to a Customer Voice Portal (CVP) contains a proper reason for failure of call transfer based on the information received by CUBE from the caller. Following are the cause codes supported:</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.14S</td>
<td>• 17—486 Busy Here</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 15.5(1)T3</td>
<td>• 19—503 Service Unavailable</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 15.5(1)S3</td>
<td>• 21—403 Forbidden</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 15.5(2)T1</td>
<td>• 31—480 Temporarily Unavailable</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 15.5(2)S1</td>
<td>• 102—504 Server Time-out</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 15.4(3)M4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 15.4(3)S4</td>
<td></td>
</tr>
</tbody>
</table>
With the Cause Code Mapping (Enhancement) feature, additional NOTIFY messages are introduced to inform CVP the proper reason for call failures based on the information received by CUBE from the caller instead of a 503 Service Unavailable message for all scenarios.

The following cause codes were introduced:

- **1**—404 Not Found
- **20**—480 Temporarily Unavailable
- **27**—502 Bad Gateway
- **28**—484 Address Incomplete
- **38**—503 Service Unavailable

### Cause Code Mapping

If CUBE is configured to consume REFERs that it receives, the following actions occur:

1. CUBE consumes the REFER that it receives from a Customer Voice Portal (CVP).
2. CUBE sends an INVITE (instead of a REFER) to the outbound leg (towards the caller).
3. CUBE receives a status from the caller.
4. CUBE sends a NOTIFY message to the CVP.

*Figure 64: Refer Consume in CUBE*

Previously, the NOTIFY message sent in step 4 included a 503 Service Unavailable message irrespective of the reason for failure of call transfer in step 3.

With the Cause Code Mapping feature, the NOTIFY message contains proper reason for failure of call transfer so that the CVP can take an appropriate action.

*Table 89: Cause Code Mappings*

<table>
<thead>
<tr>
<th>Status Message received by CUBE (Step 3)</th>
<th>Cause Code</th>
<th>Notify message sent to CVP (Step 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>486</td>
<td>17</td>
<td>486 Busy Here</td>
</tr>
</tbody>
</table>
### Configuring Cause Code Mapping

#### SUMMARY STEPS

1. enable
2. configure terminal
3. sip-ua
4. reason-header override
5. end

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enters privileged EXEC mode.  
  Example:  
  Device> enable |

Cause code mappings for cause code 19 and 21 require configurations mentioned in Configuring Cause Code Mapping, on page 761.

This mapping is only for the REFER consume scenario and not for REFER passthrough.
### Command or Action

<table>
<thead>
<tr>
<th>Step 2</th>
<th>configure terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>configure terminal</code></td>
</tr>
</tbody>
</table>

**Purpose:** Enters global configuration mode.

<table>
<thead>
<tr>
<th>Step 3</th>
<th>sip-ua</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>sip-ua</code></td>
</tr>
</tbody>
</table>

**Purpose:** Enters the SIP user agent configuration mode.

<table>
<thead>
<tr>
<th>Step 4</th>
<th>reason-header override</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>reason-header override</code></td>
</tr>
</tbody>
</table>

**Purpose:** Configures the sending of a proper reason for failure of call transfer in the NOTIFY message so that the Customer Voice Portal (CVP) can take an appropriate action.

<table>
<thead>
<tr>
<th>Step 5</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>end</code></td>
</tr>
</tbody>
</table>

**Purpose:** Exits to privileged EXEC mode.

---

### Verifying Cause Code Mapping

#### SUMMARY STEPS

1. Enter the following:
   - debug ccsip function
   - debug ccsip message
   - debug voip application state
   - debug voip application core
   - debug voip ccapi inout

#### DETAILED STEPS

Enter the following:

- debug ccsip function
- debug ccsip message
- debug voip application state
- debug voip application core
- debug voip ccapi inout

**Example:**
486 Received by CUBE:
**SIP Supplementary Services**

**Verifying Cause Code Mapping**

Received:
SIP/2.0 486 Busy Here
Via: SIP/2.0/UDP 9.40.3.231:5060;branch=z9hG4bK1C15625F7
From: <sip:2222@9.40.3.231>;tag=49B0964D-213C
To: <sip:3333@9.0.0.174>;tag=1
Call-ID: 7D7073E4-3F3B11E4-917BF9A9-A90B2232@9.40.3.231
CSeq: 101 INVITE
Allow-Events: telephone-event
Content-Length: 0
Reason: Q.850; **cause=17**

**486 Busy here response sent in NOTIFY by CUBE**

Sent:
NOTIFY sip:1111@9.0.0.174:9000 SIP/2.0
Via: SIP/2.0/UDP 9.40.3.231:5060;branch=z9hG4bK1C1571767
From: <sip:2222@9.40.3.231:5060>;tag=49B08E64-1374
To: <sip:1111@9.0.0.174>;tag=1
Call-ID: 1-25970@9.0.0.174
CSeq: 102 NOTIFY
Max-Forwards: 70
Date: Fri, 19 Sep 2014 13:55:46 GMT
User-Agent: Cisco-SIPGateway/IOS-15.5.20140712.124355.
Event: refer
Subscription-State: terminated;reason=noresource
Contact: <sip:2222@9.40.3.231:5060>
Content-Type: message/sipfrag
Content-Length: 25

**SIP/2.0 486 Busy here**
Verifying Cause Code Mapping
PART XVIII

Hosted and Cloud Services

• Hosted and Cloud Services Delivery with Cisco UBE, on page 767
• Cisco UBE SIP Registration Proxy, on page 769
• Survivability for Hosted and Cloud Services, on page 787
Hosted and Cloud Services Delivery with Cisco UBE

Cisco Unified Border Element (Cisco UBE) delivers hosted and cloud based communication services at customer sites by managing registration traffic and ensuring uninterrupted service, when the remote call control platform becomes unreachable.

Figure 65: Cisco UBE in Hosted and Cloud Services
Cisco UBE SIP Registration Proxy

The Cisco UBE SIP Registration Proxy feature allows service providers to control the flow of registration messages between a customer's private network and their hosted communications platform. By controlling routine registration traffic at the customer site, service providers can ensure service availability to local endpoints, while protecting core services from high message loads.

- Registration Pass-Through Modes, on page 769
- Registration Overload Protection, on page 774
- Registration Rate-limiting, on page 775
- Prerequisites for SIP Registration Proxy on Cisco UBE, on page 777
- Restrictions, on page 777
- Configuring Cisco UBE SIP Registration Proxy, on page 777
- Configuration Example—Cisco UBE SIP Registration Proxy, on page 786
- Feature Information for Cisco UBE SIP Registration Proxy, on page 786

Registration Pass-Through Modes

Cisco UBE uses the following two modes for registration pass-through:

End-to-End Mode

In the end-to-end mode, Cisco UBE collects the registrar details from the Uniform Resource Identifier (URI) and passes the registration messages to the registrar. The registration information contains the expiry time for rate-limiting, the challenge information from the registrar, and the challenge response from the user.

Cisco UBE also passes the challenge to the user if the register request is challenged by the registrar. The registrar sends the 401 or 407 message to the user requesting for user credentials. This process is known as challenge.

Cisco UBE ignores the local registrar and authentication configuration in the end-to-end mode. It passes the authorization headers to the registrar without the header configuration.

End-to-End Mode—Call Flows

This section explains the following end-to-end pass-through mode call flows:
Register Success Scenario

The figure below shows an end-to-end registration pass-through scenario where the registration request is successful.

Figure 66: End-to-End Registration Pass-through Mode—Register Success Scenario

The register success scenario for the end-to-end registration pass-through mode is as follows:

1. The user sends the register request to Cisco UBE.
2. Cisco UBE matches the request with a dial peer and forwards the request to the registrar.
3. Cisco UBE receives a success response message (200 OK message) from the registrar and forwards the message to the endpoint (user).
4. The registrar details and expiry value are passed to the user.

Registrar Challenging the Register Request Scenario

The figure below shows an end-to-end registration pass-through scenario where the registrar challenges the register request.
The following scenario explains how the registrar challenges the register request:

1. The user sends the register request to Cisco UBE.
2. Cisco UBE matches the register request with a dial peer and forwards it to the registrar.
3. The registrar challenges the register request.
4. Cisco UBE passes the registrar response and the challenge request, only if the registrar challenges the request to the user.
5. The user sends the register request and the challenge response to the Cisco UBE.
6. Cisco UBE forwards the response to the registrar.
7. Cisco UBE receives success message (200 OK message) from the registrar and forwards it to the user.

**Peer-to-Peer Mode**

In the peer-to-peer registration pass-through mode, the outgoing register request uses the registrar details from the local Cisco UBE configuration. Cisco UBE answers the challenges received from the registrar using the configurable authentication information. Cisco UBE can also challenge the incoming register requests and authenticate the requests before forwarding them to the network.

In this mode, Cisco UBE sends a register request to the registrar and also handles register request challenges. That is, if the registration request is challenged by the registrar (registrar sends 401 or 407 message), Cisco UBE forwards the challenge to the user and then passes the challenge response sent by the user to the registrar. In the peer-to-peer mode, Cisco UBE can use the `authentication` command to calculate the authorization header and then challenge the user depending on the configuration.
The registrar command must be configured in peer-to-peer mode. Otherwise, the register request is rejected with the 503 response message.

Peer-to-Peer Mode--Call Flows

This section explains the following peer-to-peer pass-through mode call flows:

Register Success Scenario

The figure below shows a peer-to-peer registration pass-through scenario where the registration request is successful.

Register Challenging the Register Request Scenario

The figure below shows a peer-to-peer registration pass-through scenario where the registration request is challenged by the registrar.
The following scenario explains how the registrar challenges the register request:

1. The user sends the register request to Cisco UBE.
2. Cisco UBE matches the register request with a dial peer and forwards the register request to the registrar.
3. The user responds to the challenge request.
4. Cisco UBE validates the challenge response and forwards the register request to the registrar.
5. Cisco UBE receives a success message from the registrar and forwards it to the endpoint (user).

**Registration in Different Registrar Modes**

This section explains SIP registration pass-through in the following registrar modes:

**Primary-Secondary Mode**

In the primary-secondary mode the register message is sent to both the primary and the secondary registrar servers simultaneously.

The register message is processed as follows:

- The first successful response is passed to the phone as a SUCCESS message.
- All challenges to the request are handled by Cisco UBE.
- If the final response received from the primary and the secondary servers is an error response, the error response that arrives later from the primary or the secondary server is passed to the phone.
- If only one registrar is configured, a direct mapping is performed between the primary and the secondary server.
- If no registrar is configured, or if there is a Domain Name System (DNS) failure, the "503 service not available" message is sent to the phone.
DHCP Mode

In the DHCP mode the register message is sent to the registrar server using DHCP.

Multiple Register Mode

In the multiple register mode, you can configure a dial peer to select and enable the indexed registrars. Register messages must be sent only to the specified index registrars.

The response from the registrar is mapped the same way as in the primary-secondary mode.

Registration Overload Protection

The registration overload protection functionality enables Cisco UBE to reject the registration requests that exceed the configured threshold value.

To support the registration overload protection functionality, Cisco UBE maintains a global counter to count all the pending outgoing registrations and prevents the overload of the registration requests as follows:

- The registration count is decremented if the registration transaction is terminated.
- The outgoing registrations are rejected if the count goes beyond a configured threshold.
- The incoming register request is rejected with the 503 response if the outgoing registration is activated by the incoming register request.
- A retry timer set for a random value is used for attempting the registration again if the registrations are originated from Cisco UBE or a gateway.

The registration overload protection functionality protects the network from the following:

- Avalanche Restart--All the devices in the network restart at the same time.
- Component Failures--Sudden burst of load is routed through the device due to a device failure.

Registration Overload Protection--Call Flow

The figure below shows the call flow when the register overload protection functionality is configured on Cisco UBE:
The following steps explain the register overload protection scenario:

1. The user sends a register request to Cisco UBE.
2. Cisco UBE matches the request with a dial peer and forwards the register request to the registrar.
3. The registration is rejected with a random retry value when the registration threshold value is reached.

**Note**

The call flow for the DNS query on the Out Leg is the same for the end-to-end and peer-to-peer mode.

## Registration Rate-limiting

The registration rate-limiting functionality enables you to configure different SIP registration pass-through rate-limiting options. The rate-limiting options include setting the expiry time and the fail count value for a Cisco UBE. You can configure the expiry time to reduce the load on the registrar and the network. Cisco UBE limits the reregistration rate by maintaining two different timers— an in-registration timer and an out-registration timer.

The initial registration is triggered based on the incoming register request. The expiry value for the outgoing register is selected based on the Cisco UBE configuration. On receiving the 200 OK message (response to the BYE message) from the registrar, a timer is started using the expiry value available in the 200 OK message. The timer value in the 200 OK message is called the out-registration timer. The success response is forwarded to the user. The expiry value is taken from the register request and the timer is started accordingly. This timer is called the in-registration timer. There must be a significant difference between the in-registration timer and the out-registration timer values for effective rate-limiting.

## Registration Rate-limiting Success--Call Flow

The figure below shows the call flow when the rate-limiting functionality is successful:
The following steps explain a scenario where the rate-limiting functionality is successful:

1. The user sends the register request to Cisco UBE.
2. Cisco UBE matches the registration request with a dial peer and forwards it to the registrar. The outgoing register request contains the maximum expiry value if the rate-limiting functionality is configured.
3. The registrar accepts the registration.
4. Cisco UBE forwards the success response with the proposed expiry timer value.
5. The user sends the reregistration requests based on the negotiated value. Cisco UBE resends the register requests until the out-leg expiry timer value is sent.
6. Cisco UBE forwards the subsequent register request to the registrar, if the reregister request is received after the out-leg timer is reached.
Prerequisites for SIP Registration Proxy on Cisco UBE

- You must enable the local SIP registrar. See Enabling Local SIP Registrar, on page 777.
- You must configure dial peers manually for call routing and pattern matching

Restrictions

- IPv6 support is not provided.

Configuring Cisco UBE SIP Registration Proxy

Enabling Local SIP Registrar

Perform this task to enable the local SIP registrar.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. sip
5. registrar server [expires [max value] [min value]]
6. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>enable</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters voice-service configuration mode.</td>
</tr>
<tr>
<td>voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
</tbody>
</table>
Purpose
Command or Action

Step 4

sip
Example:
Device(conf-voi-serv)# sip

Purpose
Enters service SIP configuration mode.

Step 5

registrar server [expires [max value] [min value]]
Example:
Device(conf-serv-sip)# registrar server

Purpose
Enables the local SIP registrar.

• Optionally you can configure the expiry time of the registrar using the following keywords:
  • expires--Configures the registration expiry time.
  • max--Configures the maximum registration expiry time.
  • min--Configures the minimum registration expiry time.

Note
The registrar command must be configured in peer-to-peer mode. Otherwise, the register request is rejected with the 503 response message.

Step 6

end
Example:
Device(conf-serv-sip)# end

Purpose
Exits service SIP configuration mode and returns to privileged EXEC mode.

Configuring SIP Registration Proxy at the Global Level

Perform this task to configure SIP registration proxy at the global level.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. registration passthrough [system | static | dynamic [ local-fallback value] ] [rate-limit [expires value] [fail-count value]] [reg-sync value] [registrar-index index]
6. end

DETAILED STEPS

Command or Action

Step 1

enable
Example:

Purpose
Enables privileged EXEC mode.

• Enter your password if prompted.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device&gt; enable</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters voice-service configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters service SIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters service SIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> registration passthrough [system</td>
<td>Configures the SIP registration pass-through options.</td>
</tr>
<tr>
<td>[static</td>
<td>• You can specify different SIP registration pass-through</td>
</tr>
<tr>
<td>dynamic [local-fallback value]</td>
<td>options using the following keywords:</td>
</tr>
<tr>
<td>[rate-limit [expires value]</td>
<td>• <strong>dynamic</strong>—SIP Registration uses the dynamic</td>
</tr>
<tr>
<td>[fail-count value] [reg-sync value] [registrar-index index]]</td>
<td>registrar details (default).</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(conf-serv-sip)# registration passthrough</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>local-fallback</strong>—Configures Local Fallback -</td>
</tr>
<tr>
<td></td>
<td>(e2e).</td>
</tr>
<tr>
<td></td>
<td>• <strong>rate-limit</strong>—Enables rate-limiting.</td>
</tr>
<tr>
<td></td>
<td>• <strong>reg-sync</strong>—Sends REGISTER messages when</td>
</tr>
<tr>
<td></td>
<td>registrar up (p2p).</td>
</tr>
<tr>
<td></td>
<td>• <strong>registrar-index</strong>—Configures a list of registrars</td>
</tr>
<tr>
<td></td>
<td>to be used for registration. For detailed</td>
</tr>
<tr>
<td></td>
<td>information, see Configuring Multiple Registrars</td>
</tr>
<tr>
<td></td>
<td>on SIP Trunks.</td>
</tr>
<tr>
<td></td>
<td>• <strong>static</strong>—SIP Registration Use static Registrar</td>
</tr>
<tr>
<td></td>
<td>Details.</td>
</tr>
<tr>
<td></td>
<td>• <strong>system</strong>—Use system registration passthrough</td>
</tr>
<tr>
<td></td>
<td>configuration.</td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Exits service SIP configuration mode and returns to</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(conf-serv-sip)# end</td>
<td>privileged EXEC mode.</td>
</tr>
</tbody>
</table>
Configuring SIP Registration Proxy at the Tenant Level

SUMMARY STEPS

1. enable
2. configure terminal
3. voice class tenant tag
4. registrar { dhcp | [registrar index] registrarserver-address [:port] | expires value}
5. registration passthrough [system | [static | dynamic [local-fallback value]] [rate-limit [expires value] [fail-count value]] [reg-sync value] [registrar-index index]]
6. exit
7. dial-peer voice tag voip
8. voice-class sip tenant tag
9. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>voice class tenant tag</td>
<td>Enters the tenant configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# voice class tenant 1</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>registrar { dhcp</td>
<td>[registrar index] registrarserver-address [:port]</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-class)# registrar ipv4:10.65.75.45:9052 expires 3600</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>registration passthrough [system</td>
<td>[static</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-class)# registration passthrough static</td>
<td>• You can specify different SIP registration pass-through options using the following keywords:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• dynamic — SIP Registration uses the dynamic registrar details (default).</td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>local-fallback</strong>—Configures Local Fallback - (e2e).</td>
</tr>
<tr>
<td>• <strong>rate-limit</strong>—Enables rate-limiting.</td>
</tr>
<tr>
<td>• <strong>reg-sync</strong>—Sends REGISTER messages when registrar up (p2p).</td>
</tr>
<tr>
<td>• <strong>registrar-index</strong>—Configures a list of registrars to be used for registration. For detailed information, see Configuring Multiple Registrars on SIP Trunks.</td>
</tr>
<tr>
<td>• <strong>static</strong>—SIP Registration Use static Registrar Details.</td>
</tr>
<tr>
<td>• <strong>system</strong>—Use system registration passthrough configuration.</td>
</tr>
</tbody>
</table>

### Step 6

**exit**

**Example:**

Device(config-class)# exit

Exits tenant configuration mode and returns to global configuration mode.

### Step 7

**dial-peer voice tag voip**

**Example:**

Device(config)# dial-peer voice 444 voip

Enters dial peer voice configuration mode.

### Step 8

**voice-class sip tenant tag**

**Example:**

Device(config-dial-peer)# voice-class sip tenant 1

Associates the dial-peer with the tenant.

### Step 9

**exit**

**Example:**

Device(config-class)# exit

Exits dial-peer configuration mode and returns to global configuration mode.

---

**Configuring SIP Registration Proxy at the Dial Peer Level**

Perform this task to configure SIP registration proxy at the dial peer level.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice tag voip
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice  tag voip</td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# dial-peer voice 444 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class sip registration passthrough [system</td>
<td>Configures SIP registration pass-through options on a dial peer on a dial peer.</td>
</tr>
<tr>
<td>[static</td>
<td></td>
</tr>
<tr>
<td>dynamic</td>
<td>[rate-limit [expires value] [fail-count value]] [reg-sync value] [registrar-index index]]</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# voice-class sip registration passthrough static</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits dial peer voice configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
Configuring Registration Overload Protection Functionality

Perform this task to configure registration overload protection functionality on Cisco UBE.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. sip-ua
4. registration spike max-number
5. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>sip-ua</td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>registration spike max-number</td>
<td>Configures registration overload protection functionality on Cisco UBE.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-sip-ua)# registration spike 100</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>end</td>
<td>Exits SIP user-agent configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-sip-ua)# end</td>
<td></td>
</tr>
</tbody>
</table>
Configuring Cisco UBE to Route a Call to the Registrar Endpoint

Perform this task to configure Cisco UBE to route a call to the registrar endpoint.

Note
You must perform this configuration on a dial peer that is pointing towards the endpoint.

SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice tag {pots | voatm | vofr | voip}
4. session target registrar
5. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice tag {pots</td>
<td>voatm</td>
</tr>
<tr>
<td>Example: Device(config)# dial-peer voice 444 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> session target registrar</td>
<td>Configures Cisco UBE to route the call to the registrar endpoint.</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# session target registrar</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits dial peer voice configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>
Verifying the SIP Registration on Cisco UBE

Perform this task to verify the configuration for SIP registration on Cisco UBE. The `show` commands need not be entered in any specific order.

**SUMMARY STEPS**

1. `enable`
2. `show sip-ua registration passthrough status`
3. `show sip-ua registration passthrough status detail`

**DETAILED STEPS**

**Step 1**  
enable  
Enables privileged EXEC mode.  
**Example:**

```
Device> enable
```

**Step 2**  
show sip-ua registration passthrough status  
Displays the SIP user agent (UA) registration pass-through status information.  
**Example:**

```
Device# show sip-ua registration passthrough status
CallId Line peer mode In-Exp reg-I Out-Exp
---------- ---------- ----- ------ ------- -------
771 5500550055 1 p2p 64 1 64
```

**Step 3**  
show sip-ua registration passthrough status detail  
Displays the SIP UA registration pass-through status information in detail.  
**Example:**

```
Device# show sip-ua registration passthrough status detail
Configured Reg Spike Value: 0
Number of Pending Registrations: 0

Call-Id: 763
Registering Number: 5500550055
Dial-peer tag: 601
Pass-through Mode: p2p
Negotiated In-Expires: 64 Seconds
Next In-Register Due in: 59 Seconds
In-Register Contact: 9.45.36.5
Registrar Index: 1
Registrar URL: ipv4:9.45.36.4
Negotiated Out-Expires: 64 Seconds
Next Out-Register After: 0 Seconds
```
The following section will be added to the "Examples" section of the SIP to SIP chapter.

**Configuration Example—Cisco UBE SIP Registration Proxy**

```plaintext
! voice service voip
  sip
    registrar server expires max 121 min 61
      registration passthrough static rate-limit expires 9000 fail-count 5 registrar-index 1 3 5
    !
    dial-peer voice 1111 voip
      destination-pattern 1234
      voice-class sip pass-thru content unsupp
      session protocol sipv2
      session target registrar
    !
    dial-peer voice 1111 voip
      destination-pattern 1234
      voice-class sip registration passthrough static rate-limit expires 9000 fail-count 5 registrar-index 1 3 5
      authentication username 1234 password 7 075E731F1A realm cisco.com
      session protocol sipv2
      session target registrar
    !
    sip-ua
      registration spike 1000
    !
    !
```

**Feature Information for Cisco UBE SIP Registration Proxy**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Support for Cisco UBE SIP Registration Proxy | Cisco IOS XE Fuji 16.9.1 | Cisco UBE SIP Registration Proxy supports sending outbound registrations from Cisco UBE based on incoming registrations. This feature enables direct registration of SIP endpoints with the SIP registrar in hosted Unified Communications deployments. This feature also provides various benefits for handling Cisco UBE deployments with no IPPBX support.

The following commands were introduced or modified: `authentication` (dial peer), `registrar server`, `registration passthrough`, `registration spike`, `show sip-ua registration passthrough status`, `voice-class sip registration passthrough static rate-limit`. |
Survivability for Hosted and Cloud Services

The Survivability for Hosted and Cloud Services on the Cisco UBE is used to:

• Monitor the WAN status periodically from the Cisco UBE.
• Route calls and handle line-side subscriptions locally when the WAN link is down.
• Synchronize the registrations with the server when the WAN link is up.

• Information About Survivability for Hosted and Cloud Services, on page 787
• How to Configure Survivability for Hosted and Cloud Services, on page 792
• Configuration Examples—Survivability for Hosted and Cloud Services, on page 803
• Feature Information for Survivability for Hosted and Cloud Services, on page 805

Information About Survivability for Hosted and Cloud Services

Advantages of Using Cisco UBE Survivability Feature

The survivability feature on Cisco UBE addresses the following issues by providing local fallback or registration synchronization:

1. When a WAN link or registrar server comes up, it waits until each SIP phone sends the REGISTER message to the server, so that outside phones can reach that phone.
2. If the phone register timer setting is too large, the outside phone waits that much time to reach that phone, after a link flap.
3. If the phone register timer setting is too small, it floods the WAN link.
4. When the WAN link or registrar server is down, you cannot make any local calls.

Local Fallback

• Cisco UBE does not need to configure credentials, as the phones trigger registration. Although Cisco UBE receives REGISTER messages for each phone every 5 minutes; for example, it throttles and sends REGISTER messages every 1 hour to the registrar server, avoiding high WAN bandwidth usage. This addresses the issues 1, 2, and 3.
• In normal operation when the WAN link or registrar server is up, the phone’s primary server URL is the registrar server (E2E) registration.

• "OPTIONS ping" is used to monitor the registrar server link status. When the detected link is down, Cisco UBE replies with a 500 message and when the phone receives this message, it sends the REGISTER message to Cisco UBE, which is the secondary server (P2P registration). Cisco UBE replies with a 200 OK message to P2P registration when the link is down. The dial-peer keeps the dynamic registrar session target and the local call does not fail. This addresses issue 4.

Registration Synchronization

• If you configure the phones to send REGISTER messages every 1 hour (to help alleviate the WAN link), the Cisco UBE uses the credentials that were configured to respond to registrar server authentication challenge. This addresses issue 3.

• When the WAN link or registration server is down (detected by OPTIONS ping), the Cisco UBE keeps the registration database of the SIP phones that were previously registered successfully, and it does not send REGISTER messages out; Cisco UBE replies with a 200 OK message and dial-peer keeps the dynamic registrar session target. The local call does not fail, addressing issue 4.

• When the registrar link is up after a link flap, the Cisco UBE sends REGISTER message for each phone that was earlier successfully registered to the registrar server. This is throttled to avoid bulk REGISTER messages flooding WAN link and the registrar. This addresses issues 1 and 2.

Registration Through Alias Mapping

The following illustration shows how a phone (with alias mapping) registers to the service provider through Cisco UBE.

**Figure 72: SIP Phone Registration**

The addresses-of-record (AOR) sent in the REGISTER is an alias which is mapped to an extension and (or) phone number by the service provider. The service provider returns the mapping details in the 200 OK response sent to the REGISTER. Cisco UBE has the ability to cache the alias mapping details in its call routing database. When a call is made from the phone, the Request-URI of the INVITE contains the dialed number (short extension or phone number).

If WAN is up, Cisco UBE always routes the INVITE sent from the phone to the service provider without looking up at the alias mapping cache.
If WAN or the service provider is down, that is, in survivability mode, Cisco UBE routes the INVITE locally by looking up at the alias mapping cache.

**Alias Mapping—Supported Methods**

1. When the service provider returns the mapping details in the 200 OK message of the REGISTER in the following predefined format:

<table>
<thead>
<tr>
<th>Alias</th>
<th>Extension</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice10000189_1111</td>
<td>1111</td>
<td>10000189</td>
</tr>
</tbody>
</table>

2. The short extension or phone number is embedded in the AOR of the REGISTER. For example, AOR is `alice10000189_1111` and the short extension is `1111`.

An inbound sip profile can be applied to the REGISTER which extracts the extension part from the AOR and adds an X-CISCO-EXTENSION header.

### Cisco UBE when WAN is UP

The following illustration provides an example as to how a typical phone makes a call to another local phone registered in the same server when WAN or the registrar server is up in a typical hosted deployment. The circled numbers in the image indicate the numerical order in which the sequence occurs.

**Figure 73: WAN Link is UP - Cisco UBE Deployment**

The call flow scenario is as follows: Phone A initiates a call to the Phone B registered to the same server.

1. Phone A sends an initial INVITE request to Phone B to participate in a call session through Cisco UBE.
2. Cisco UBE sends this INVITE to the service provider.
3. The service provider in turn sends the INVITE to Cisco UBE. Since the WAN link is up, the service provider maps details of the user from the register server and provides details of the user, for example, alias of the user, short extension number, and phone number.
4. Cisco UBE sends INVITE with all the above mentioned information to Phone B.
5. Phone B sends a 200 OK response to Cisco UBE for the received INVITE.
6. Cisco UBE sends a 200 OK answer to the service provider.
7. The service provider responds to Cisco UBE with a 200 OK answer.
8. A final 200 OK response is sent to Phone A by Cisco UBE and the call is established between Phone A and Phone B.

**Example: Normal Mode (WAN is Up in P2P Mode)**

```
CUBE# show sip-ua registration passthrough status
```
```
CallId DirectoryNum peer mode In-Exp reg-I Out-Exp survival
-------- -------- ---- ----- ----- ----- ---------
21 NCPhone1006 1 p2p 135 /144 1 144 normal
```

**Example: Normal Mode (WAN is Up in E2E Mode)**

```
CUBE# show sip-ua registration passthrough status
```
```
CallId DirectoryNum peer mode In-Exp reg-I Out-Exp survival
-------- -------- ---- ----- ----- ----- ---------
14574 NCPhone1006 301 e2e 117 /120 -- 120 normal
```

**Cisco UBE Survivability When WAN Is Down**

In survivability mode, Cisco UBE provides end-to-end telephony services when access to the centralized servers is interrupted because of a WAN outage or other factors, like the server being down.

The following illustration shows how a call is established between two endpoints when WAN link is down during survivability by directly dialing into an extension.

![Cisco UBE Survivability When WAN Is Down](image)

Earlier, when WAN was down, User A could only contact User B using either the alias or the user-id of User B, and not using their extensions or phone numbers.

Now, in the event the WAN link or registration server is down, when a local call is made, INVITE is sent to Cisco UBE. Cisco UBE maps the details of the user like the extension number and phone-number stored during registration. Local phones can now be reached on their short extensions or phone numbers by similar phones that are subscribed to the server through the same Cisco UBE.
It is possible to register multiple contacts for a single AOR; however, if multiple contacts are registered for a single subscriber, the Cisco UBE uses only the topmost registered contact to deliver the call to that subscriber. For this reason, multiple contacts are not supported.

A few phone models, such as, Cisco IP Phone 7800 Series with Multiplatform Firmware and Cisco IP Phone 8800 Series with Multiplatform Firmware, sends register request to primary registrar only and do not send secondary REGISTER request to the secondary registrar (Cisco UBE) in E2E mode when primary registrar could not be reached. In such scenarios, phone service goes down after it receives 500 response from Cisco UBE for REGISTER request toward primary registrar.

To avoid phones getting into such error condition, Cisco UBE checks for the response from the primary registrar side. When Cisco UBE receives request timeout on WAN side or responses other than 200, 4XX, and 3XX from primary registrar, survivability will be enabled.

To enable survivability on such phones, refer Configuring Survivability for Phones Sending Single Register Request, on page 795.

**Survivability Support for Public Switched Telephone Network Access When WAN Is Down**

If WAN link going down or registrar service unavailable, you can access the phones in the Public Switched Telephone Network (PSTN) through FXO or PRI cards that are configured on Cisco Unified Border Element.

---

**Note**

Survivability support for Public Switched Telephone Network (PSTN) access is supported only for Cisco UBE running on Cisco 4000 Series Integrated Services Router.

---

*Figure 75: Survivability Support for PSTN Access When WAN Is Down*

---

**Example: Survivability Mode in P2P (regsync mode) when WAN is Down**

```
CUBE# show sip-ua registration passthrough status
CallId DirectoryNum peer mode In-Exp reg-I Out-Exp survival
------- ========= ===== ===== ========= ===== =========
38 NCPhone1008 1 p2p 3595 /3600 1 3600 regsync
```

---

Cisco Unified Border Element Configuration Guide
Example: Survivability Mode in E2E (local fallback mode) when WAN is Down

```
CUBE# show sip-ua registration passthrough status

CallId DirectoryNum peer mode In-Exp reg-I Out-Exp survival
------- --------------- ------ ----- ------ ------ ------ ------
70      NCPhone1006 1  e2e  35 /70  --  0  locfall

CallId DirectoryNum peer mode In-Exp reg-I Out-Exp survival
------- --------------- ------ ----- ------ ------ ------ ------
513     NCPhone1008 1  e2e  40 /70  --  0  locfall
```

How to Configure Survivability for Hosted and Cloud Services

Configuring Local Fallback or Registration Synchronization Globally

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `voice service voip`
4. `sip`
5. `registration passthrough local-fallback` *tag*
6. `end`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td><code>voice service voip</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# voice service voip</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td><code>sip</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# sip</code></td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

Command or Action:

Device(conf-voi-serv)# sip

### Purpose

Device(conf-voi-serv)# sip

**Step 5**  
**registration passthrough local-fallback** *tag*  
**Example:**  
Device(conf-voi-serv)# registration passthrough local-fallback 10

**Purpose**

Configures SIP registration passthrough for local fallback mode; this will locally respond to REGISTER in p2p mode when WAN is down. The *tag* is the WAN link or registrar server dial-peer tag.

- To configure the registration sync mode, you can use the `registration passthrough reg-sync *tag*` command. Use the *static* keyword to set the phone URL to p2p registration.

**Step 6**  
**end**  
**Example:**  
Device(conf-voi-serv)# end

**Purpose**

Returns to privileged EXEC mode.

### Configuring Local Fallback or Registration Synchronization at the Tenant Level

**SUMMARY STEPS**

1. enable  
2. configure terminal  
3. voice class tenant *tag*  
4. registration passthrough local-fallback *tag*  
5. exit  
6. dial-peer voice *tag* voip  
7. voice-class sip tenant *tag*  
8. exit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
| Example: Device> enable |  
| **Step 2** configure terminal | Enters global configuration mode. |
| Example: Device# configure terminal | |
| **Step 3** voice class tenant *tag* | Enters voice class tenant configuration mode. |
| Example: | |
Configuring Local Fallback or Registration Synchronization on a Dial Peer

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `dial-peer voice tag voip`
4. `voice-class sip tenant tag registration passthrough local-fallback tag`
5. `end`

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Device(config)# voice class tenant 1</code></td>
<td>Configures SIP registration passthrough for local fallback mode; this locally responds to REGISTER in p2p mode when WAN is down. The <code>tag</code> is the WAN link or registrar server dial-peer <code>tag</code>.</td>
</tr>
<tr>
<td><strong>Step 4</strong> registration passthrough local-fallback <code>tag</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> <code>Device(config-class)# registration passthrough local-fallback 10</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits tenant configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> dial-peer voice <code>tag voip</code></td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> voice-class sip tenant <code>tag</code></td>
<td>Associates the dial-peer with the tenant.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> exit</td>
<td>Exits dial-peer configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice tag voip</td>
<td>Enters dial peer VoIP configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# dial-peer voice 4 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class sip registration passthrough local-fallback tag</td>
<td>Configures SIP registration passthrough for local fallback mode; this will locally respond to REGISTER in p2p mode when WAN is down. The tag is the WAN link or registrar server dial-peer tag.</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# voice-class sip registration passthrough local-fallback 10</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device(conf-serv-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>

Configuring Survivability for Phones Sending Single Register Request

The following configuration enables Cisco UBE to always check for the response from remote side. Request timeout on WAN side or response other than 200, 4XX, and 3XX received by Cisco UBE from SBC enables the survivability.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. survivability single-register
6. end
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
  - Enter your password if prompted. |
| **Example:**  
  Device> enable | |
| **Step 2** configure terminal | Enters global configuration mode. |
| **Example:**  
  Device# configure terminal | |
| **Step 3** voice service voip | Enters voice service VoIP configuration mode. |
| **Example:**  
  Device(config)# voice service voip | |
| **Step 4** sip | Enters voice service SIP configuration mode. |
| **Example:**  
  Device(conf-voi-serv)# sip | |
| **Step 5** survivability single-register | Enables Cisco UBE to always check for the response from the remote side. Request timeout on WAN side or response other than 200, 4XX, and 3XX received by Cisco UBE from SBC enables the survivability. |
| **Example:**  
  Device(conf-serv-sip)# survivability single-register | |
| **Step 6** end | Returns to privileged EXEC mode. |
| **Example:**  
  Device(conf-serv-sip)# end | |

### Configuring OPTIONS Ping

#### SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice  
   tag  
   voip
4. voice-class sip options-keepalive up-interval  
   value  
   down-interval  
   value
5. end
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
</tr>
<tr>
<td></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>dial-peer voice tag voip</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 3 voip</td>
</tr>
<tr>
<td></td>
<td>Enters dial peer configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>voice-class sip options-keepalive up-interval value down-interval value</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# voice-class sip options-keepalive up-interval 120 down-interval 120</td>
</tr>
<tr>
<td></td>
<td>Configures OPTIONS keepalive timer interval for DOWN and UP endpoints.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>end</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# end</td>
</tr>
<tr>
<td></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

Configuring Registration Timer

Perform the following task to configure the registration timer in the Cisco UBE rather than on all SIP phones.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. registrar server expires max value min value
6. end
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>- Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> registrar server expires max value min value</td>
<td>Configures the maximum and minimum time (in seconds) for the registration expiry in Cisco UBE.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# registrar server expires max 300 min 200</td>
<td>- If the phone sends expiry time as 600 seconds, then the Cisco UBE will reply with 200 OK message and expiry time 300 seconds, and the phone will resend with expiry 300.</td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>

Configuring the REGISTER Message Throttling in Cisco UBE

Perform the following task to throttle the REGISTER message in Cisco UBE.

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. registration passthrough rate-limit expires value local-fallback tag
## Configuring the Class of Restrictions (COR) List

Class of Restrictions (COR) provides the ability to deny certain call attempts based on the incoming and outgoing class of restrictions that are provisioned on the dial peers.

COR specifies which incoming dial peer can use which outgoing dial peer to make a call. You can provision each dial peer with an incoming and an outgoing COR list. The incoming COR list indicates the capability of the dial peer to initiate certain classes of calls. The outgoing COR list indicates the capability that is required for an incoming dial peer to deliver a call through this outgoing dial peer.

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip</td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> registration passthrough rate-limit expires value local-fallback tag</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Configures the SIP registration passthrough rate-limit expiry value for local-fallback (e2e). Although Cisco UBE receives the REGISTER message every 5 minutes (300 seconds), it will send only one register message every one hour.</td>
</tr>
<tr>
<td>Device(conf-serv-sip)# registration passthrough rate-limit expires 3600 local-fallback 3</td>
<td>• Under dial peer configuration mode, you can use the voice-class sip registration passthrough rate-limit expires value reg-sync dial-peer-tag command.</td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>
Before you begin

You must configure COR Groups. For more information, see Dial Peer Configuration Guide.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice *tag* voip
4. corlist incoming *dial-peer*
5. corlist outgoing *dial-peer*
6. description *string*
7. destination-pattern *number*
8. session protocol sipv2
9. session target registrar
10. voice-class sip registration passthrough local-fallback *tag*
11. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>dial-peer voice <em>tag</em> voip</td>
<td>Enters dial peer configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# dial-peer voice 3 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>corlist incoming <em>dial-peer</em></td>
<td>Specifies the COR to be applied on an incoming dial peer (for incoming calls).</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# corlist incoming FromPhone</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>corlist outgoing <em>dial-peer</em></td>
<td>Specifies the COR to be applied for outgoing dial peer (for outgoing calls).</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# corlist outgoing FromSP</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> description \textit{string}</td>
<td>Adds a description to a dial peer.</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# description registration</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> destination-pattern \textit{number}</td>
<td>Specifies either the prefix or the full E.164 phone number to be used for the dial peer.</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# destination-pattern 1111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> session protocol sipv2</td>
<td>Specifies the session protocol for SIP calls between local and remote devices using the packet network.</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> session target registrar</td>
<td>Specifies to route the call to the registrar endpoint for SIP dial peers.</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# session target registrar</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> voice-class sip registration passthrough local-fallback \textit{tag}</td>
<td>Configures SIP registration passthrough for local fallback mode.</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# voice-class sip registration passthrough local-fallback 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
<td></td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# end</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Verifying Survivability for Hosted and Cloud Services

The \texttt{show} commands can be entered in any order.

**SUMMARY STEPS**

1. enable
2. show dial-peer voice summary
3. show sip-ua registration passthrough status
4. show sip-ua register status
5. show voip rtp connections
6. show call active voice compact
## DETAILED STEPS

### Step 1 enable

Enables privileged EXEC mode.

**Example:**

```
Device> enable
```

### Step 2 show dial-peer voice summary

Displays the summary information for each voice dial peer.

**Example:**

```
Device# show dial-peer voice summary
```

<table>
<thead>
<tr>
<th>dial-peer</th>
<th>hunt</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>TAG</td>
<td>TYPE</td>
</tr>
<tr>
<td>1</td>
<td>voip</td>
<td>up</td>
</tr>
<tr>
<td>2</td>
<td>voip</td>
<td>up</td>
</tr>
<tr>
<td>1000</td>
<td>voip</td>
<td>down</td>
</tr>
<tr>
<td>101</td>
<td>voip</td>
<td>down</td>
</tr>
<tr>
<td>102</td>
<td>voip</td>
<td>down</td>
</tr>
<tr>
<td>300</td>
<td>voip</td>
<td>down</td>
</tr>
<tr>
<td>400</td>
<td>voip</td>
<td>down</td>
</tr>
</tbody>
</table>

### Step 3 show sip-ua registration passthrough status

Displays information about the SIP user agent registration passthrough status. In the sample output shown below, the parameter In-Exp shows the remaining expiry time and the survival field parameters can be regsync, locfall, or normal.

**Example:**

```
Device# show sip-ua registration passthrough status
```

<table>
<thead>
<tr>
<th>CallId</th>
<th>Line</th>
<th>peer</th>
<th>expires(sec)</th>
<th>reg-I</th>
<th>Out-Exp</th>
<th>survival</th>
</tr>
</thead>
<tbody>
<tr>
<td>5300</td>
<td>1111008</td>
<td>1</td>
<td>e2e 1041 /1200</td>
<td>-----</td>
<td>1200 normal *</td>
<td></td>
</tr>
<tr>
<td>5305</td>
<td>1111002</td>
<td>1</td>
<td>e2e 2847 /3000</td>
<td>-----</td>
<td>3000 normal *</td>
<td></td>
</tr>
<tr>
<td>5311</td>
<td>1111020</td>
<td>1</td>
<td>e2e 1070 /1200</td>
<td>-----</td>
<td>1200 normal *</td>
<td></td>
</tr>
</tbody>
</table>

### Step 4 show sip-ua register status

Displays information about the SIP user agent register status.

**Example:**

```
Device# show sip-ua register status
```

<table>
<thead>
<tr>
<th>Line</th>
<th>peer</th>
<th>expires(sec)</th>
<th>reg</th>
<th>survival</th>
<th>P=Associ-URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>11123</td>
<td>23</td>
<td>59</td>
<td>yes</td>
<td>regsync</td>
<td></td>
</tr>
</tbody>
</table>

### Step 5 show voip rtp connections

Displays Real-Time Transport Protocol (RTP) named event packets.
Example:

Device# show voip rtp connections

VoIP RTP Port Usage Information:
Max Ports Available: 8091, Ports Reserved: 101, Ports in Use: 2
Port range not configured, Min: 16384, Max: 32767

<table>
<thead>
<tr>
<th>Ports</th>
<th>Media-Address Range</th>
<th>Available Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>8091</td>
<td>2</td>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No.</th>
<th>CallId</th>
<th>dstCallId</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>LocalIP</th>
<th>RemoteIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5324</td>
<td>5325</td>
<td>16410</td>
<td>16464</td>
<td>9.40.1.168</td>
<td>9.40.1.173</td>
</tr>
<tr>
<td>2</td>
<td>5325</td>
<td>5324</td>
<td>16412</td>
<td>16528</td>
<td>9.40.1.168</td>
<td>9.40.1.174</td>
</tr>
</tbody>
</table>

Found 2 active RTP connections

Step 6: show call active voice compact

Displays the compact version of the call information for voice calls in progress.

Example:

Device# show call active voice compact

<table>
<thead>
<tr>
<th>&lt;callID&gt;</th>
<th>A/O</th>
<th>FAX</th>
<th>T&lt;sec&gt;</th>
<th>Codec</th>
<th>type</th>
<th>Peer Address</th>
<th>IP</th>
<th>R&lt;ip&gt;:&lt;udp&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>5324</td>
<td>ANS</td>
<td>T9</td>
<td>g711ulaw</td>
<td>VOIP</td>
<td>P1111008</td>
<td>9.40.1.173:16464</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5325</td>
<td>ORG</td>
<td>T9</td>
<td>g711ulaw</td>
<td>VOIP</td>
<td>P1111020</td>
<td>9.40.1.174:16528</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total call-legs: 2

Configuration Examples—Survivability for Hosted and Cloud Services

Example: Configuring Local Fallback Globally

In the following example, local fallback is configured at global level:

Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(config-voi-serv)# sip
Device(config-serv-sip)# registration passthrough local-fallback 10
Device(config-serv-sip)# end
Example: Configuring Local Fallback at the Tenant Level

In the following example, local fallback is configured for tenant 1 and is applied for dial-peer 444:

Device>enable
Device#configure terminal
Device(config)#voice class tenant 1
Device(config-class)#registration passthrough local-fallback 10
Device(config-class)#exit
Device(config)#dial-peer voice 444 voip
Device(config-dial-peer)#voice-class sip tenant 1
Device(config-class)#exit

Example: Configuring Local Fallback on a Dial Peer

In the following example, local fallback is configured on dial-peer 2.

Device> enable
Device# configure terminal
Device(config)# dial-peer voice 2 voip
Device(config-dial-peer)# voice-class sip registration passthrough local-fallback 10
Device(config-dial-peer)# end

Example: Configuring Survivability for Phones Sending Single Register Request

In the following example, survivability is configured for phones sending single register request:

Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# survivability single-register
Device(config-serv-sip)# end

Example: Configuring OPTIONS Ping

In the following example, OPTIONS Ping is configured on dial-peer 3:

Device> enable
Device# configure terminal
Device(config)# dial-peer voice 3 voip
Device(config-dial-peer)# voice-class sip options-keepalive up-interval 120 down-interval 120
Device(config-dial-peer)# end

Example: Configuring the Registration Timer

In the following example, registration timer is configured with a expiration value of minimum 200 and maximum 300 seconds.
Example: Configuring REGISTER Message Throttling

In the following example, REGISTER message throttling is configured:

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(config-voi-serv)# sip
Device(config-serv-sip)# registrar server expires max 300 min 200
Device(config-serv-sip)# end
```

Example: Configuring the COR List

In the following example, "FromPhone" and "FromSP" COR groups are configured and applied to dial-peer 2:

```
Device> enable
Device# configure terminal
Device(config)# dial-peer cor list FromPhone
Device(config-dp-corlist)# member 911
Device(config-dp-corlist)# member 1800
Device(config)# dial-peer cor list FromSP
Device(config-dp-corlist)# member 911
Device(config-dp-corlist)# member 1800
Device(config)# exit
Device(config)# dial-peer voice 2 voip
Device(config-dial-peer)# corlist incoming FromPhone
Device(config-dial-peer)# corlist outgoing FromSP
Device(config-dial-peer)# description registration
Device(config-dial-peer)# destination-pattern 1111
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target registrar
Device(config-dial-peer)# voice-class sip registration passthrough local-fallback 5
Device(config-dial-peer)# end
```

Feature Information for Survivability for Hosted and Cloud Services

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

### Table 91: Feature Information for Survivability for Hosted and Cloud Services

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Survivability for Hosted and Cloud Services</td>
<td>Cisco IOS XE Fuji 16.9.1</td>
<td>Supports survivability for Hosted and Cloud Services.</td>
</tr>
</tbody>
</table>
PART XIX

Cisco Unified Communications Manager Line-Side Support

• Cisco Unified Communications Manager Line-Side Support, on page 809
Cisco Unified Communications Manager Line-Side Support

The Cisco Unified Communications Manager (Unified Communications Manager) Lineside feature is no longer supported. The feature is deprecated for Cisco Unified Border Element on Cisco IOS 15.5(2)T Release and later releases. To support this feature, you must configure Cisco Unified Border Element on Cisco IOS 15.4(2)T or prior releases.

Cisco Unified Communications Manager is an enterprise-class IP communications processing system. It extends enterprise telephony features and capabilities to IP phones, media processing devices, VoIP gateways, mobile devices, and multimedia applications. Cisco Unified Border Element (Cisco UBE) provides line-side support for Cisco Unified Communications Manager. This support enables communication between devices (such as phones) used by remote users on different logical networks, in both cloud-based and premise-based deployments.

- Feature Information for Cisco Unified Communications Manager Line-Side Support, on page 809
- Restrictions for Cisco Unified Communications Manager Line-Side Support, on page 810
- Information About Cisco Unified Communications Manager Line-Side Support, on page 811

Feature Information for Cisco Unified Communications Manager Line-Side Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 92: Feature Information for Cisco Unified Communications Manager Line-Side Support

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco Unified Communications Manager Line-Side Support</td>
<td>15.5(2)T</td>
<td>The Cisco Unified Communications Manager (CUCM) Line-Side Support feature was supported until the release 15.4(2)T. This feature has been deprecated from 15.5(2)T release onwards.</td>
</tr>
<tr>
<td>Simplified Line-Side Support of CUCM on CUBE</td>
<td>15.4(2)T Cisco IOS XE Release 3.12S</td>
<td>The Simplified Line-Side Support of CUCM on CUBE feature simplifies the complex CUBE configurations required for registering IP Phones on a CUCM through CUBE using a single CLI that automatically applies all the necessary configurations. The following commands were modified by this feature: <code>extension cucm</code> and <code>voice-class sip extension cucm</code>.</td>
</tr>
<tr>
<td>Cisco Unified Communications Manager Line-Side Support</td>
<td>15.3(3)M Cisco IOS XE Release 3.10S</td>
<td>The Cisco Unified Communications Manager Line-Side Support feature provides line-side support for Cisco Unified Communications Manager and IP phones deployed on different logical networks, in both cloud-based and premise-based deployments. The following commands were introduced or modified: <code>access-secure</code>, <code>capf-address</code>, <code>clear voice phone-proxy all-sessions</code>, <code>complete (ctl file)</code>, <code>ctl-file (phone proxy)</code>, <code>debug voice phone-proxy</code>, <code>description (ctl file)</code>, <code>description (phone proxy)</code>, <code>disable service-settings</code>, <code>max-concurrent-sessions</code>, <code>phone-proxy (dial peer)</code>, <code>port-range</code>, <code>record-entry</code>, <code>show voice class ctl-file</code>, <code>show voice class phone-proxy</code>, <code>service-map</code>, <code>session-timeout</code>, <code>tftp-server address</code>, <code>voice-ctl-file</code>, <code>voice-phone-proxy</code>.</td>
</tr>
</tbody>
</table>

Restrictions for Cisco Unified Communications Manager Line-Side Support

- In Cisco Unified Communications Manager Line-Side Support deployments, Cisco Unified Border Element does not support TFTP encrypted configuration files.
Information About Cisco Unified Communications Manager Line-Side Support

Cisco UBE Line-Side Deployment

In a typical deployment Cisco Unified Border Element (Cisco UBE) is placed between the Cisco Unified Communications Manager and the endpoint. Before invoking a service the phone contacts the CUBE Trivial File Transfer Protocol (TFTP) server to get configuration information such as the Certificate Trust List (CTL) file and phone-specific configuration settings. The phone then registers with Cisco Unified Communications Manager. In the deployment shown below, Cisco Unified Communications Manager and the phone configuration operate in unsecured mode (TCP to Real-Time Transport Protocol). The phone configuration can be changed to operate in a secure mode (Transport Layer Security Secure to Real-Time Transport Protocol) if needed. When the phone registration is completed the phone can invoke all normal call services.

![Cisco UBE Line-Side Deployment Diagram]

Line-Side Deployment Scenarios

Cisco Unified Call Manager Line-Side support can be deployed in the following ways:

- **Line-Side Secure Deployment** -

  CUCM line-side secure deployment, provides secure access between phone and CUBE. CUBE terminates the TLS connection from phone and initiates a TCP connection to CUCM to perform TLS-TCP inter-working. Refer to 'Example: Configuring CUCM Secure Line-Side' section for the steps involved in configuring secure deployment.

- **Line-Side Non-Secure Deployment** -

  CUCM line-side non-secure deployment, provides a non-secure connection between phone and CUBE. Refer to 'Example: Configuring CUCM Non-Secure Line-Side' section for the steps involved in configuring non-secure deployment.
Line-Side Support for CUCM on CUBE

For an IP phone to register on a CUCM through CUBE, CUBE must be configured to do the following requirements.

- TCP must be used for registration.
- The MAC address of the device (device ID) and the device name, present in the CONTACT header of the REGISTER message, need to be copied to the outgoing messages and passed to the CUCM intact.

Table 93: Command for Line-Side Support for CUCM on CUBE

<table>
<thead>
<tr>
<th>Dial-Peer Configuration Mode (config-dial-peer)</th>
<th>Global VoIP Configuration mode (config-voi-serv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>voice-class sip extension cucm</td>
<td>sip</td>
</tr>
<tr>
<td></td>
<td>extension cucm</td>
</tr>
</tbody>
</table>

When Line Side Support for CUCM on CUBE feature is configured, the following supported, nonmandatory headers are passed through automatically without the need for further configuration:

- Call-Info
- Content-ID
- Allow-Events
- Supported
- Remote-Party-ID
- Require
- Referred-By

Figure 77: Predefined Supported NonMandatory Headers

When Line Side Support for CUCM on CUBE is configured, predefined SIP profiles automatically remove the Cisco-Guide header from the outgoing INVITE.

Figure 78: Predefined SIP Profile

If a user explicitly configures the above configurations, ensure that the configurations are merged with the above automatic configurations.
# Configuring a PKI Trustpoint

## SUMMARY STEPS

1. `crypto key generate rsa [label key-label] [modulus modulus-size] general-keys`
2. `crypto pki trustpoint name`
3. `enrollment selfsigned`
4. `subject-name [x.500-name]`
5. `subject-alt-name sip-security-profile-name`
6. `revocation-check method1[method2 [method3]]`
7. `rsakeypair key-label`

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | `crypto key generate rsa [label key-label] [modulus modulus-size] general-keys` | Generates a RSA key pair.  
**Note** A self-signed key can only support a `modulus-size` value of 1024 bits.  
Example:  
Device(config)# crypto key generate rsa label pp_rsa modulus 1024 general-keys |
| Step 2 | `crypto pki trustpoint name` | Declares the trustpoint that the device should use and enters ca-trustpoint configuration mode.  
Example:  
Device(config)# crypto pki trustpoint callmg23 |
| Step 3 | `enrollment selfsigned` | Specifies self-signed enrollment for a trustpoint.  
Example:  
Device(config)# crypto pki trustpoint callmg23 enrollment selfsigned |
| Step 4 | `subject-name [x.500-name]` | Specifies the subject name in the certificate request.  
Example:  
Device(config-ca-trustpoint)# subject-name CN=ASR1006-CCN-4 |
| Step 5 | `subject-alt-name sip-security-profile-name` | Specifies the alternative subject name in the certificate request.  
- Use the `subject-alt-name` command only when Cisco UBE is interacting with CUCM in secure mode.  
- The value of `subject-alt-name` must be the SIP security profile name under CUCM.  
Example:  
Device(config-ca-trustpoint)# subject-alt-name 6961_SEC.cisco.com 8941_SEC.cisco.com 8945_SEC.cisco.com 7975_SEC.cisco.com 7970_SEC.cisco.com |
## Importing the CUCM and CAPF Key

### Before you begin

Download the CUCM key (the CallManager.pem file) from the Cisco Unified Communications Manager Operating System Administration web page.

Login to Cisco Unified OS Administration and Security and Certificate Management, download the CUCM key (the CallManager.pem file), and copy and paste the CUCM key to CUBE.

### SUMMARY STEPS

1. `crypto pki trustpoint name`
2. `revocation-check method1[method2 [method3]]`
3. `enrollment terminal`
4. `crypto pki authenticate name`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>crypto pki trustpoint name</code></td>
<td>Creates a trustpoint for the CUCM key and enters ca-trustpoint configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# crypto pki trustpoint cucm_trustpoint</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>revocation-check method1[method2 [method3]]</code></td>
<td>Checks the revocation status of a certificate.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config-ca-trustpoint)# revocation-check none</code></td>
<td></td>
</tr>
</tbody>
</table>
Creating a CTL File

**SUMMARY STEPS**

1. `voice-ctl-file ctl-filename`
2. `record-entry selfsigned trustpoint trustpoint-name`
3. `record-entry capf trustpoint trustpoint-name`
4. `record-entry cucm-tftp trustpoint trustpoint-name`
5. `complete`

**DETAILS STEPS**

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>voice-ctl-file ctl-filename</code></td>
<td>Creates a CTL file and enters CTL file configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device(config)#voice-ctl-file ct1</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>record-entry selfsigned trustpoint trustpoint-name</code></td>
<td>Configures the trustpoints to be used for creating the CTL file.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device(config-ctl-file)#record-entry selfsigned trustpoint self-trustpoint6s</code></td>
<td></td>
</tr>
</tbody>
</table>
### Configuring a Phone Proxy

**SUMMARY STEPS**

1. `voice-phone-proxy phone-proxy-name`
2. `voice-phone-proxy file-buffer size`
3. `tftp-server-address [ipv4 server-ip-address | domain-name]`
4. `ctl-file ctl-filename`
5. `access-secure`
6. `complete`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><code>voice-phone-proxy phone-proxy-name</code></td>
<td>Configures a phone proxy and enters phone-proxy configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# voice-phone-proxy pp</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Step 2</strong></th>
<th><strong>Purpose</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>voice-phone-proxy file-buffer size</code></td>
<td>Configures the phone-proxy file buffering parameter, in MB.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# voice-phone-proxy file-buffer 30</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Step 3</strong></th>
<th><strong>Purpose</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>`tftp-server-address [ipv4 server-ip-address</td>
<td>domain-name]`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

### Configuring a Phone Proxy

**Purpose Command or Action**

**Step 3**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>record-entry capf trustpoint trustpoint-name</code></td>
<td>Specifies that the trustpoint is created using the CAPF certificate imported from Cisco Unified Communications Manager to the device.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-ctl-file)# record-entry capf trustpoint capf-trustpoint6s</code></td>
<td></td>
</tr>
</tbody>
</table>

**Step 4**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>record-entry cucm-tftp trustpoint trustpoint-name</code></td>
<td>Specifies that the trustpoint is created using the specified TFTP and Cisco Unified Communications Manager certificate imported to the device.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-ctl-file)# record-entry cucm-tftp trustpoint cucm-trustpoint</code></td>
<td></td>
</tr>
</tbody>
</table>

**Step 5**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>complete</code></td>
<td>Completes the CTL-file creation.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-ctl-file)# complete</code></td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

**Device(config-phone-proxy)# tftp-server-address ipv4 172.110.36.2**

**Step 4**

ctl-file *ctl-filename*

*Example:*

Device(config-phone-proxy)# ctl-file ctl

**Step 5**

access-secure

*Example:*

Device(config-phone-proxy)# access-secure

**Step 6**

complete

*Example:*

Device(config-phone-proxy)# complete

---

### Attaching a Phone Proxy to a Dial Peer

**SUMMARY STEPS**

1. dial-peer voice *tag* voip
2. phone-proxy *phone-proxy-name* signal-addr ipv4 *ipv4-address* cucm ipv4 *ipv4-address*
3. session protocol sipv2
4. session target registrar
5. session transport {udp | tcp [tls]}
6. incoming uri {from | request | to | via} *tag*
7. destination uri *tag*
8. voice-class sip call-route url
9. voice-class sip profiles *number*
10. voice-class sip registration passthrough [registrar-index *index*]
11. voice-class sip pass-thru headers
12. voice-class sip copy-list {tag | system}
13. codec transparent

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**

**dial-peer voice *tag* voip**

*Example:*

Device(config)# dial-peer voice 10 voip

**Purpose**

Defines a particular dial peer, specifies the method of voice encapsulation, and enters dial peer configuration mode.
### Attaching a Phone Proxy to a Dial Peer

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong></td>
<td>Configures the phone proxy for the related dial peer.</td>
</tr>
<tr>
<td><code>phone-proxy phone-proxy-name signal-addr ipv4 ipv4-address cucm ipv4 ipv4-address</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# phone-proxy pp1 signal-addr ipv4 10.0.0.8 cucm ipv4 198.51.100.1</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Specifies a session protocol (SIPv2) for calls between local and remote devices.</td>
</tr>
<tr>
<td><code>session protocol sipv2</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Specifies that a call from a VoIP dial peer is routed to the registrar end point.</td>
</tr>
<tr>
<td><code>session target registrar</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session target registrar</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Configures the underlying transport layer protocol for SIP messages to transport layer security over TCP (TLS over TCP).</td>
</tr>
<tr>
<td>`session transport {udp</td>
<td>tcp [tls]}`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session transport tcp tls</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Specifies the voice class used to match the VoIP dial peer to the uniform resource identifier (URI) of an incoming call. Any request matching “uri 11” is destined to this dial peer.</td>
</tr>
<tr>
<td>`incoming uri {from</td>
<td>request</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# incoming uri request 11</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Specifies the voice class used to match a dial peer to the destination URI of an outgoing call. Any request matching “uri 12” is destined to this dial peer.</td>
</tr>
<tr>
<td><code>destination uri tag</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# destination uri 12</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Enables call routing based on the URL.</td>
</tr>
<tr>
<td><code>voice-class sip call-route url</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# voice-class sip call-route url</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>Configures a SIP profile for a voice class.</td>
</tr>
<tr>
<td><code>voice-class sip profiles number</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# voice-class sip profiles 10</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Step 10</td>
<td>voice-class sip registration passthrough [registrar-index index]</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# voice-class sip registration passthrough registrar-index 1</td>
</tr>
<tr>
<td>Step 11</td>
<td>voice-class sip pass-thru headers</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# voice-class sip pass-thru headers 10</td>
</tr>
<tr>
<td>Step 12</td>
<td>voice-class sip copy-list {tag</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# voice-class sip copy-list 10</td>
</tr>
<tr>
<td>Step 13</td>
<td>codec transparent</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# codec transparent</td>
</tr>
</tbody>
</table>

### Verifying CUCM Lineside Support

The `show` commands can be entered in any order.

#### SUMMARY STEPS

1. `enable`
2. `show dial-peer voice dial-peer-id | section voice class sip extension`
3. `show dial-peer voice`
4. `show voice class phone-proxy`
5. `show voice class phone-proxy sessions`

#### DETAILED STEPS

**Step 1**  
`enable`

Enables privileged EXEC mode.

- Enter your password if prompted.

**Example:**

```
Device> enable
```
Step 2  
show dial-peer voice dial-peer-id | section voice class sip extension

Example:
CUBE# show dial-peer voice 5678 | section voice class sip extension
voice class sip extension = system,
Displays if extension cucm has not been configured for the dial peer.

Example:
CUBE# show dial-peer voice 5678 | section voice class sip extension
voice class sip extension = cucm,
Displays if extension cucm has been configured for the dial peer.

Example:
CUBE# show dial-peer voice 5678 | section voice class sip extension
voice class sip extension = none,
Displays if extension cucm has been removed for the dial peer using the no form of the command.

Step 3  
show dial-peer voice

Example:
Device# show dial-peer voice 100

voice class sip extension = system,
voice class sip contact-passing = system,
voice class sip requri-passing = system,
voice class phone proxy name: phone_proxy_secure
voice class phone proxy config: complete

Step 4  
show voice class phone-proxy

Example:
Device# show voice class phone-proxy

Phone-Proxy 'phone_proxy' :
Description:
  Access Secure: non-secure (default)
  Tftp-server address: 20.21.27.146
  Capf server address: 20.21.27.146
  CUCM service settings: preserve (default)
  CTL file name: ctl_file
  Session-timeout: 180 seconds
  Max-concurrent-sessions: 30
  Current sessions: 0
  TFTP sessions: 0
  HTTP download sessions: 0
  HTTP application sessions: 0
  CAPF sessions: 0
  Config status: complete
SIP dial-peers associated:
  Name
  ------------------
  1
  ""
Phone-Proxy 'phone_proxy_secure':
Description:
Access Secure: secure
Tftp-server address: 20.21.27.146
Capf server address: 20.21.27.146
CUCM service settings: preserve (default)
CTL file name: ctl_file
Session-timeout: 180 seconds
Max-concurrent-sessions: 30
Current sessions: 0
TFTP sessions: 0
HTTP download sessions: 0
HTTP application sessions: 0
CAPF sessions: 0
Config status: complete
SIP dial-peers associated:
Name
-------------------
3
dialpeer4
--------------------------------------------

Step 5

show voice class phone-proxy sessions

Example:

Device# show voice class phone-proxy sessions

Phone-Proxy 'phone_proxy_ipad':
<table>
<thead>
<tr>
<th>Source</th>
<th>Sessions of Dial-peer 5</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access: 10.74.9.219 :45232</td>
<td>10.74.9.209 :6970</td>
<td></td>
</tr>
<tr>
<td>Core: 20.21.29.209 :45300</td>
<td>20.21.27.146 :6970</td>
<td></td>
</tr>
</tbody>
</table>
--------------------------------------------

Example: Configuring a PKI Trustpoint

Device(config)# crypto key generate rsa label pp_rsa modulus 1024 general-keys
Device(config)# crypto pki trustpoint callmg23
Device(config-ca-trustpoint)# enrollment selfsigned
Device(config-ca-trustpoint)# subject-name CN=ASR1006-CCN-4
Device(config-ca-trustpoint)# subject-alt-name 6961_SEC.cisco.com 8941_SEC.cisco.com 8945_SEC.cisco.com 7975_SEC.cisco.com 7970_SEC.cisco.com
Device(config-ca-trustpoint)# revocation-check crl
Device(config-ca-trustpoint)# rsakeypair pp1
Example: Importing the CUCM and CAPF Key

The following example shows how to import the CUCM and CAPF key after you have downloaded the CUCM key (the CallManager.pem file) and the CAPF key (the CAPF.pem file) from the Cisco Unified Communications Manager Operating System Administration web page.

```plaintext
Device(config)# crypto pki trustpoint cucm_trustpoint
Device(config-ca-trustpoint)# revocation-check none
Device(config-ca-trustpoint)# enrollment terminal
Device(config-ca-trustpoint)# crypto pki authenticate cucm_trustpoint
```

Example: Creating a CTL File

```plaintext
Device(config)# voice-ctl-file ct1
Device(config-ctl-file)# record-entry selfsigned trustpoint self-trustpoint6s
Device(config-ctl-file)# record-entry capf trustpoint capf-trustpoint6s
Device(config-ctl-file)# record-entry cucm-tftp trustpoint cucm-trustpoint
Device(config-ctl-file)# complete
```

Example: Configuring a Phone Proxy

```plaintext
Device(config)# voice-phone-proxy pp
Device(config-phone-proxy)# voice-phone-proxy pp
Device(config-phone-proxy)# voice-phone-proxy file-buffer size 30
Device(config-phone-proxy)# tftp-server address ipv4 172.110.36.2
Device(config-phone-proxy)# ctl-file ct1
Device(config-phone-proxy)# access-secure
Device(config-phone-proxy)# complete
```

Example: Attaching a Phone Proxy to a Dial Peer

```plaintext
Device(config)# dial-peer voice 10 voip
Device(config-dial-peer)# phone-proxy pp1 signal-addr ipv4 10.0.0.8 cucm ipv4 198.51.100.1
Device(config-dial-peer)# session-protocol sipv2
Device(config-dial-peer)# session target registrar
Device(config-dial-peer)# session transport tcp tls
Device(config-dial-peer)# incoming uri request 11
Device(config-dial-peer)# destination uri 12
Device(config-dial-peer)# voice-class sip call-route url
Device(config-dial-peer)# voice-class sip profiles 10
Device(config-dial-peer)# voice-class sip registration passthrough registrar-index 1
Device(config-dial-peer)# voice-class sip passthrough headers 10
Device(config-dial-peer)# voice-class sip copy-list 10
Device(config-dial-peer)# codec transparent
```
Example: Configuring CUCM Secure Line-Side

The details of the IP address used in the below example are as follows:

- CUBE IP address facing phone : 172.18.110.120
- CUBE IP address facing CUCM : 10.50.209.100
- CUCM IP address : 10.50.209.215

Generate and Import Certificate on CUBE

Device(config)# crypto pki trustpoint selfsign
Device(config)# enrollment selfsigned
Device(config)# subject-name CN=CUBE, O=CISCO
Device(config)# revocation-check none
Device(config)# rsakeypair selfsign

Device(config)# crypto pki trustpoint ccm1
Device(config)# enrollment terminal
Device(config)# revocation-check none

Device(config)# crypto pki trustpoint Cisco_Manufacturing_CA
Device(config)# enrollment terminal
Device(config)# revocation-check none

Device(config)# crypto pki trustpoint selfsignx
Device(config)# enrollment terminal
Device(config)# subject-name cn=3925_pod5
Device(config)# revocation-check none
Device(config)# rsakeypair selfsignx

Device(config)# crypto pki certificate chain ccm1
Device(config)# certificate ca 55C2FCBFBC552B7C6CED497D4AD33F8
[Certificate data omitted]

Device(config)# crypto pki certificate chain Cisco_Manufacturing_CA
Device(config)# certificate ca 6A6967B30000000000000003
[Certificate data omitted]

Device(config)# crypto pki certificate chain selfsignx
Device(config)# certificate self-signed 01
[Certificate data omitted]

Add the Cube Service, Call Flow and Message manipulation configuration.

Device(config)# voice service voip
Device(config)# no ip address trusted authenticate
Device(config)# allow-connections sip to sip
Device(config)# fax protocol t38 version 0 is-redundancy 0 hs-redundancy 0 fallback none
Device(config)# sip
Device(config-sip)# session transport tcp
Device(config-sip)# header-passing
Device(config-sip)# registrar server
Device(config-sip)# nat auto
Device(config-sip)# pass-thru headers unsupp
Device(config-sip)# pass-thru subscribe-notify-events all
Device(config-sip)# pass-thru content unsupp
Device(config-sip)# registration passthrough
Device(config-sip)# extension cucm

Device(config)# voice class uri 1 sip
Device(config)# host ipv4:172.18.110.120
Device(config)# voice class uri 2 sip
Device(config)# host ipv4:10.50.209.100

Device(config)# voice class uri 3 sip
Device(config)# host ipv4:10.50.209.215

Device(config)# interface GigabitEthernet0/0
Device(config-if)# ip address 10.50.209.100 255.255.255.0
Device(config-if)# duplex auto
Device(config-if)# speed auto

Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip address 172.18.110.120 255.255.255.0
Device(config-if)# duplex auto
Device(config-if)# speed auto

Device(config)# dspfarm profile 1 transcode universal security
Device(config-dspfarm-profile)# codec g722-64
Device(config-dspfarm-profile)# codec g711ulaw
Device(config-dspfarm-profile)# codec g711alaw
Device(config-dspfarm-profile)# codec g729ar8
Device(config-dspfarm-profile)# codec g729abr8
Device(config-dspfarm-profile)# maximum sessions 24
Device(config-dspfarm-profile)# associate application CUBE

Configure CTL and Phone Proxy
Device(config)#voice-ctl-file ctl_secure
Device(config-ctl-file)# record-entry capf trustpoint Cisco_Manufacturing_CA
Device(config-ctl-file)# record-entry selfsigned trustpoint selfsignx
Device(config-ctl-file)# complete

Device(config)# voice-phone-proxy phone_proxy
Device(config-phone-proxy)# tftp-server address ipv4 10.50.209.215
Device(config-phone-proxy)# access-secure
Device(config-phone-proxy)# service-map server-addr ipv4 10.50.209.215 port 8443
Device(config-phone-proxy)# service-map server-addr ipv4 10.50.209.215 port 8080
Device(config-phone-proxy)# complete

Device(config)# voice-phone-proxy tftp-address ipv4 10.50.209.100
Device(config-phone-proxy)# port-range 40000 50000
Device(config)# voice-phone-proxy tftp-address ipv4 172.18.110.120
Device(config-phone-proxy)# port-range 40000 50000
Device(config-phone-proxy)# voice-phone-proxy file-buffer size 60

Attaching Phone Proxy to dial Peers
Device(config)# dial-peer voice 1 voip
Device(config-dial-peer)# phone-proxy phone_proxy signal-addr ipv4 172.18.110.120 cucm ipv4 10.50.209.215
 *** Access Dialpeer Facing Outside ***
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target registrar
Device(config-dial-peer)# session transport tcp tls
Example: Configuring CUCM Non-Secure Line-Side

The details of the IP address used in the below example are as follows:

- CUBE IP address facing phone : 172.18.110.120
- CUBE IP address facing CUCM : 10.50.209.100
- CUCM IP address : 10.50.209.215

Generate and Import Certificate on CUBE

Add the Cube Service, Call Flow and Message manipulation configuration.
Device(config)# voice service voip
Device(config)# no ip address trusted authenticate
Device(config)# allow-connections sip to sip
Device(config)# fax protocol t38 version 0 ls-redundancy 0 hs-redundancy 0 fallback none
Device(config)# sip
Device(config-sip)# header-passing
Device(config-sip)# registrar server
Device(config-sip)# nat auto
Device(config-sip)# pass-thru headers unsupp
Device(config-sip)# pass-thru subscribe-notify-events all
Device(config-sip)# pass-thru content unsupp
Device(config-sip)# registration passthrough

Device(config)# voice class uri 1 sip
Device(config)# host ipv4:172.18.110.120

Device(config)# voice class uri 2 sip
Device(config)# host ipv4:10.50.209.100

Device(config)# voice class uri 3 sip
Device(config)# host ipv4:10.50.209.215

Device(config)# interface GigabitEthernet0/0
Device(config-if)# ip address 10.50.209.100 255.255.255.0
Device(config-if)# duplex auto
Device(config-if)# speed auto

Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip address 172.18.110.120 255.255.255.0
Device(config-if)# duplex auto
Device(config-if)# speed auto

Configure CTL and Phone Proxy

Device(config)#voice-ctl-file ctl_secure
Device(config-ctl-file)# record-entry capf trustpoint Cisco_Manufacturing_CA
Device(config-ctl-file)# record-entry selfsigned trustpoint selfsignx
Device(config-ctl-file)# record-entry cucm-tftp trustpoint ccm1
Device(config-ctl-file)# complete

Device(config)# voice-phone-proxy phone_proxy
Device(config-phone-proxy)# tftp-server address ipv4 10.50.209.215 local-addr ipv4 10.50.209.100 acc-addr ipv4 172.18.110.120 port 8443
Device(config-phone-proxy)# ctl-file ctl_secure
Device(config-phone-proxy)# access-secure
Device(config-phone-proxy)# service-map server-addr ipv4 10.50.209.215 port 8443 acc-addr ipv4 172.18.110.120 port 8080
Device(config-phone-proxy)# service-map server-addr ipv4 10.50.209.215 port 3804 acc-addr ipv4 172.18.110.120 port 3804
Device(config-phone-proxy)# complete

Device(config)# voice-phone-proxy tftp-address ipv4 10.50.209.100
Device(config-phone-proxy)# port-range 40000 50000
Device (Config)# voice-phone-proxy tftp-address ipv4 172.18.110.120
Device(config-phone-proxy)# port-range 40000 50000
Device(config-phone-proxy)# voice-phone-proxy file-buffer size 60

Attaching Phone Proxy to dial Peers
Device(config)# dial-peer voice 1 voip
Device(config-dial-peer)# phone-proxy phone_proxy signal-addr ipv4 172.18.110.120 cucm ipv4 10.50.209.215
*** Access Dialpeer Facing Outside ***
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target registrar
Device(config-dial-peer)# session transport tcp tls
Device(config-dial-peer)# destination uri 2
Device(config-dial-peer)# incoming uri request 1
Device(config-dial-peer)# voice-class sip extension cucm
Device(config-dial-peer)# voice-class sip conn-reuse
Device(config-dial-peer)# voice-class sip call-route url
Device(config-dial-peer)# voice-class sip registration passthrough registrar-index 1
Device(config-dial-peer)# dtmf-relay rtp-nte
Device(config-dial-peer)# codec transparent

Device(config)# dial-peer voice 2 voip
*** Core Dialpeer Facing CUCM ***
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.50.209.215
Device(config-dial-peer)# session transport tcp
Device(config-dial-peer)# destination uri 1
Device(config-dial-peer)# incoming uri via 3
Device(config-dial-peer)# voice-class sip call-route url
Device(config-dial-peer)# dtmf-relay rtp-nte
Device(config-dial-peer)# codec transparent

Configuring SIP User Agent

Device(config)# sip-ua
Device(config-sip-ua)# timers connection aging 60
Device(config-sip-ua)# registrar 1 ipv4:10.50.209.215 expires 3600 refresh-ratio 100 tcp
Example: Configuring CUCM Non-Secure Line-Side
PART XX

Security

- SIP TLS Support on CUBE, on page 831
CHAPTER 68

SIP TLS Support on CUBE

The Cisco Unified Border Element (CUBE) supports SIP-to-SIP calls with Transport Layer Security (TLS). TLS provides privacy and data integrity of SIP signaling messages between two applications that communicate. CUBE uses TLS to secure SIP signaling messages. TLS is layered on top of a reliable transport protocol such as TCP. CUBE can be configured at both the global and dial-peer levels for allowing TLS to establish sessions with remote endpoints.

- Feature Information for SIP TLS Support on CUBE, on page 831
- Restrictions, on page 832
- Information About SIP TLS Support on CUBE, on page 832
- How to Configure SIP TLS Support on CUBE, on page 834
- Configuration Examples for SIP TLS Support on CUBE, on page 842

Feature Information for SIP TLS Support on CUBE

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server identity validation through Common Name (CN) and Subject Alternate Name (SAN)</td>
<td>Cisco IOS XE Gibraltar Release 16.11.1a</td>
<td>Support for server identity validation through Common Name (CN) and Subject Alternate Name (SAN) fields in the server certificate.</td>
</tr>
<tr>
<td>Elliptical Curve Ciphers</td>
<td>Cisco IOS XE Gibraltar Release 16.10.1a</td>
<td>Support for configuring Elliptic Curve for a TLS session.</td>
</tr>
</tbody>
</table>
### Feature Information

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change in the default SIP TLS Versions support on CUBE</td>
<td>Cisco IOS XE 16.9.1</td>
<td>Behavior of the command <code>transport tcp tls</code> is modified. In the earlier releases, TLS version v1.0, v1.1 and v1.2 were enabled by default. From this release onwards, only versions v1.1 and v1.2 are enabled by default. TLS version v1.0 is excluded.</td>
</tr>
<tr>
<td>SIP TLS Version 1.2 Support on CUBE</td>
<td>Cisco IOS 15.6(1)T, Cisco IOS XE 3.17S</td>
<td>Support is provided for SIP-to-SIP calls with Transport Layer Security (TLS) version 1.2. The following cipher suites are introduced for release Cisco IOS 15.6(1)T: • TLS_DHE_RSA_WITH_AES_128_CBC_SHA1 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 The following commands were introduced or modified: <code>transport tcp tls</code>, <code>crypto signaling default trustpoint cube</code>, and <code>srtp (voice)</code>.</td>
</tr>
<tr>
<td>SIP TLS Version 1.0 Support on CUBE</td>
<td>Cisco IOS 12.4(6)T</td>
<td>Support is provided for SIP-to-SIP calls with Transport Layer Security (TLS) version 1.0. The following cipher suites are introduced for release Cisco IOS 12.4(6)T: • SSL_RSA_WITH_RC4_128_MD5 • TLS_RSA_WITH_AES_128_CBC_SHA The following commands were introduced or modified: <code>transport tcp tls</code> and <code>crypto signaling default trustpoint cube</code>.</td>
</tr>
</tbody>
</table>

### Restrictions

- ECDSA ciphers are not supported on TLS version 1.0.

### Information About SIP TLS Support on CUBE

**Deployment**

The following figure illustrates an example of CUBE with SIP TLS connections.
In a typical deployment, CUBE is placed between CUCM and the service provider. These devices are authenticated and enrolled with a Certificate Authority (CA) server that issues certificates. It can be Cisco or a third party entity. When a call is made, a TLS handshake is initiated between CUCM and CUBE, and the IOS PKI infrastructure is used to exchange certificates signed by a common trusted CA during the handshake. During the TLS handshake, a dynamically generated symmetric key and cipher algorithms are negotiated between the devices. After the successful TLS handshake, the devices establish a SIP session between the service provider and CUBE. Keys exchanged during the TLS handshake process are used to encrypt or decrypt all SIP signaling messages.

**Note**
The use of PKI on the Cisco IOS software requires that the clock on the devices be synchronized with the network time to ensure proper validation of certificates.

**TLS Cipher Suite Category**

Prior to release Cisco IOS15.6(1)T, CUBE supported TLS v1.0 with the following cipher suites:

- SSL_RSA_WITH_RC4_128_MD5
- TLS_RSA_WITH_AES_128_CBC_SHA

CUBE supports only the mandatory cipher suites for TLS implementation. From Cisco IOS15.6(1)T release onwards, CUBE supports TLS v1.2 which is backward compatible. Following are the cipher suites added:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA1
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

Use the `srtppass-thru` command to globally enable the transparent passthrough of all (supported and unsupported) crypto suites. If SRTP pass-thru feature is enabled, media interworking will not be supported. Ensure that you have symmetric configuration on both the incoming and outgoing dial-peers to avoid media-related issues.
How to Configure SIP TLS Support on CUBE

Configuring SIP TLS on CUBE

Starting from IOS XE version 16.6.1 release, in the Cisco 4000 Series ISR devices the key-pairs information is encrypted.

When you downgrade the Cisco 4000 Series ISR from IOS XE version 16.6.1 or a later release to a pre-16.6.1 release, ensure that you disable the key encryption before the downgrade. Otherwise, the downgrade discards the encrypted keys. To disable the encryption, use the command `no service private-config-encryption` in global configuration mode.

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `crypto key generate rsa {general-keys | usage-keys} label key-label [exportable] [ modulus modulus-size] [storage device:]`
4. `crypto key generate eckey-size {256 | 384} [label] [ec key-label]`
5. `crypto pki trustpoint name`
6. `rsakeypair key-label [key-size [encryption-key-size]]`
7. `eckeypair keyname`  ! Applicable only for TLS version 1.2.
8. `serial-number [none]`
9. `ip-address {ip-address | interface | none]`
10. `subject-name [x.500-name]`
11. `enrollment [mode] [retry period minutes] [retry count number] url [url [pem]]`
12. `crl optional or revocation-check method1 [method2 [method3]]`
13. `password string`
14. `exit`
15. `crypto ca authenticate name or crypto pki authenticate name`
16. `crypto ca enroll name or crypto pki enroll name`
17. `sip ua`
18. `transport tcp tls [v1.0 | v1.1 | v1.2]`
19. `crypto signaling {remote-addr ip address subnet mask | default} trustpoint trustpoint-name`  
   `[{ecdsa-cipher [curve-size 384] | strict-cipher} | cn-san-validate {server [{ecdsa-cipher [curve-size 384] | strict-cipher}] }]`  ! ECDSA ciphers are not supported on TLS version 1.0.
20. `voice service {pots | voatm | vofr | voip}`
21. `transport tcp tls`
22. `url {sip | sips | tel}`
23. `end`
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | enable | Enables privileged EXEC mode.  
- Enter your password if prompted. |
| Example: | Device> enable |
| Step 2 | configure terminal | Enters global configuration mode. |
| Example: | Router# configure terminal |
| Step 3 | crypto key generatersa[general-keys | usage-keys]labelkey-label[exportable ][modulus modulus-size][storage device:] | Generates RSA key pairs. Arguments and keywords are as follows:  
- general-keys—Specifies that the general-purpose key pair should be generated.  
- usage-keys—Specifies that two RSA special-usage key pairs should be generated (that is, one encryption pair and one signature pair) instead of one general-purpose key pair.  
- label key-label—(Optional) Name that is used for an RSA key pair when they are being exported. If a key label is not specified, the fully qualified domain name (FQDN) of the router is used.  
- exportable—(Optional) Specifies that the RSA key pair can be exported to another Cisco device, such as a router.  
- modulus modulus-size—(Optional) IP size of the key modulus in a range from 350 to 2048. If you do not enter the modulus keyword and specify a size, you will be prompted.  
- storage device:—(Optional) Specifies the key storage location. The name of the storage device is followed by a colon (:).  
- kp1— kp1 is a label name that you select. |
<p>| Example: | Router(config)# crypto key generate rsa general-keys label kp1 exportable |
| Step 4 | crypto key generateeckey-size256 | 384][labellabel][ec key-label] | Generates EC key pairs. |
| Example: | Router(config)# crypto key generate rsa general-keys label kp1 exportable |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 5</strong> Required: <code>crypto pki trustpoint name</code></td>
<td>Declares the trustpoint that your router should use. Argument is as follows:</td>
</tr>
<tr>
<td>Example: <code>Router(config)# crypto pki trustpoint cube1</code></td>
<td>- <code>name</code>—Creates a name for the trustpoint that you created.</td>
</tr>
<tr>
<td></td>
<td>- <code>cube1</code>—Represents the trustpoint name that the user specifies.</td>
</tr>
<tr>
<td><strong>Step 6</strong> <code>rsakeypair key-label [key-size [encryption-key-size]]</code></td>
<td>Specifies which key pair to associate with the certificate. Arguments are as follows:</td>
</tr>
<tr>
<td>Example: <code>Router(config)# rsakeypair kp1</code></td>
<td>- <code>key-label</code>—Name of the key pair, which is generated during enrollment if it does not already exists or if the <code>auto-enroll regenerate</code> command is configured.</td>
</tr>
<tr>
<td></td>
<td>- <code>key-size</code>—(Optional) Size of the desired RSA key. If not specified, the existing key size is used.</td>
</tr>
<tr>
<td></td>
<td>- <code>encryption-key-size</code>—(Optional) Size of the second key, which is used to request separate encryption, signature keys, and certificates.</td>
</tr>
<tr>
<td><strong>Step 7</strong> <code>eckeypair keyname</code></td>
<td>Generates EC keys for ECDSA cipher suites.</td>
</tr>
<tr>
<td>Example: <code>Router(config)# eckeypair mykey</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> <code>serial-number [none]</code></td>
<td>Specifies whether the router serial number should be included in the certificate request. Keyword is as follows:</td>
</tr>
<tr>
<td>Example: <code>Router(ca-trustpoint)# serial-number</code></td>
<td>- <code>none</code>—(Optional) Specifies that a serial number will not be included in the certificate request.</td>
</tr>
<tr>
<td><strong>Step 9</strong> `ip-address {ip-address</td>
<td>interface</td>
</tr>
<tr>
<td>Example: <code>Router(ca-trustpoint)# ip-address 172.18.197.154</code></td>
<td>- <code>ip-address</code>—Specifies a dotted IP address that will be included as &quot;unstructuredAddress&quot; in the certificate request.</td>
</tr>
<tr>
<td></td>
<td>- <code>interface</code>—Specifies an interface, from which the router can get an IP address, that will be included as &quot;unstructuredAddress&quot; in the certificate request.</td>
</tr>
<tr>
<td></td>
<td>- <code>none</code>—Specifies that an IP address is not to be included in the certificate request.</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| 10   | **subject-name** [x.500-name] | Specifies the subject name in the certificate request. Argument is as follows:  
  - **x.500-name**—(Optional) Specifies the subject name used in the certificate request. |
|      | **Example:**     |         |
|      | Router (ca-trustpoint) # subject-name  
  CN=172.18.197.154 |         |
| 11   | **enrollment** [mode][retry period minutes][retry count number]url url[pem] | Specifies the enrollment parameters of a certificate authority (CA). Arguments and keywords are as follows:  
  - **mode**—(Optional) Registration authority (RA) mode, if your CA system provides an RA. By default, RA mode is disabled.  
  - **retry period minutes**—(Optional) Specifies the period in which the router will wait before sending the CA another certificate request. The default is 1 minute between retries. (Specify from 1 through 60 minutes.)  
  - **retry count number**—(Optional) Specifies the number of times a router will resend a certificate request when it does not receive a response from the previous request. The default is 10 retries. (Specify from 1 through 100 retries.)  
  - **url url**—URL of the file system where your router should send certificate requests. For enrollment method options, see the enrollment url command.  
  - **pem**—(Optional) Adds privacy-enhanced mail (PEM) boundaries to the certificate request. |
|      | **Example:**     |         |
|      | Router (ca-trustpoint) # enrollment url  
  http://172.18.193.103 |         |
| 12   | **crl optional** or revocation-check method1[method2][method3]] | Allows the certificates of other peers to be accepted without trying to obtain the appropriate CRL or checks the revocation status of a certificate. Arguments are as follows:  
  - **method1 [method2 [method3]]**—Method used by the router to check the revocation status of the certificate. |
|      | **Example:**     |         |
|      | Router (ca-trustpoint) # crl optional  
  or  
  Router (ca-trustpoint) # revocation-check none |         |

**Note** If the second and the third methods are specified, each method will be used only if the previous method returns an error, such as the server being down.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 13** | **password string**  
**Example:**  
Router(ca-trustpoint)# password password |
| Specifies the revocation password for the certificate. Argument is as follows:  
• string—Name of the password |
| **Step 14** | **exit**  
**Example:**  
Router# exit |
| Exists the current mode. |
| **Step 15** | **crypto ca authenticate name** or **crypto pki authenticate name**  
**Example:**  
Router(config)# crypto ca authenticate cube1  
or  
Router(config)# crypto pki authenticate cube1 |
| Authenticates the CA (by getting the certificate of the CA). Argument is as follows:  
• name—Specifies the name of the CA. This is the same name used when the CA was declared with the crypto ca identity command.  
**Note**  
This is where you paste the remote root CA certificate (PEM file format). |
| **Step 16** | **crypto ca enroll name** or **crypto pki enroll name**  
**Example:**  
Router(config)# crypto ca name cube1  
or  
Router(config)# crypto pki name cube1 |
| Obtains the certificates of your router from the certificate authority. The CA server issues two certificates to the trustpoint (CUBE): one to certify the CA server and the other to certify the trustpoint (CUBE). Argument is as follows:  
• name—Specifies the name of the CA. Use the same name when you declared the CA using the crypto pki trustpoint command. |
| **Step 17** | **sip-ua**  
**Example:**  
Router(config)# sip-ua |
| Enters SIP user-agent configuration mode. |
| **Step 18** | **transport tcp tls [v1.0 | v1.1 | v1.2 ]**  
**Example:**  
Router(config-sip-ua)# transport tcp tls v1.2 |
| Configures the specified TLS version.  
**Note**  
TLS v1.1 and TLS v1.2 are the default TLS versions that are configured. TLS v1.0 is also supported. However, to configure TLS v1.0, you must explicitly specify the TLS version.  
For more information on the TLS version configuration, see Transport command. |
| **Step 19** | **crypto signaling {remote-addr ip address subnet mask | default} trustpoint trustpoint-name [ {ecdsa-cipher [curve-size 384] | strict-cipher} | cn-san-validate {server [{ecdsa-cipher [curve-size 384] {strict-cipher}] } / ECDSA ciphers are not supported on TLS version 1.0.} |**  
**Example:**  
Router(config-sip-ua)# transport tcp tls v1.2 |
<p>| Configures the SIP gateway to use its trustpoint when it establishes or accepts TLS connection with a remote device with an IP address. |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Router(config-sip-ua)# crypto signaling default trustpoint cube1</code></td>
<td>The trustpoint label refers to the CUBE’s certificate that is generated with the Cisco IOS PKI commands as part of the enrollment process. <strong>strict-cipher</strong> means that the SIP TLS process uses only those cipher suites that are mandated by the SIP RFC. When you use the <strong>strict-cipher</strong> command argument, avoids changes to the configuration if SIP should mandate newer ciphers. The SSL layer in Cisco IOS does not support TLS_RSA_WITH_3DES_EDE_CBC_SHA. Therefore, CUBE actively uses only the TLS_RSA_WITH_AES_128_CBC_SHA suite in strict mode.</td>
</tr>
</tbody>
</table>

Keywords and arguments are as follows:

- **remote-addr address**—Associates an IP address to a trustpoint.
- **remote-addr subnet mask**—Associates a subnet mask to a trustpoint.
- **default**—Configures a default trustpoint.
- **trustpoint string**—Refers to the SIP gateways certificate generated as part of the enrollment process using Cisco IOS PKI commands.
- **ecdsa-cipher**—Examples are the following: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 and TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.

**Note** ecdsa-cipher is applicable only for the TLS version 1.2.

- **curve-size** - configures the specific size of elliptic curves to be used for a TLS session.
  - **384**- configures 384-bit Elliptic Curve
- **strict-cipher**—Examples are the following: TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA1, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, and TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.
- **cn-san-validate {server}**- Enables the server identity validation through Common Name (CN) and Subject Alternate Name (SAN) fields in the server certificate during client-side SIP/TLS connections. Validation of the CN and SAN fields of the server certificate ensures that the server-side domain is a valid entity. While setting up a TLS connection to a target server,
Verifying SIP TLS Support on CUBE

After a call is made, the **show sip-ua connections tcp tls** command is used to verify whether the transport used for the call is TLS.

Sample output for this command when TLS version is 1.0:

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>voice service {pots</td>
<td>voatm</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>transport tcp tls</td>
<td>Enters this command in SIP configuration mode to enable the TLS port on TCP 5061 to listen.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-voi-sip)# transport tcp tls</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>url {sip</td>
<td>sips</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-serv-sip)# url sips</td>
<td>• <strong>sip</strong>—Generate URLs in SIP format for VoIP calls. This is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>sips</strong>—Generate URLs in SIPS format for VoIP calls.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>tel</strong>—Generate URLs in TEL format for VoIP calls.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> This SIP gateway is now configured to use TLS with endpoints sharing the same CA.</td>
</tr>
<tr>
<td>23</td>
<td>end</td>
<td>Ends the current mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(conf-serv-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>
Verifying SIP TLS Support on CUBE

Detail Output

```
router# show sip-ua connections tcp tls detail
Total active connections: 1
No. of send failures: 0
No. of remote closures: 3
No. of conn. failures: 0
No. of inactive conn. ageouts: 0
Max. tls send msg queue size of 0, recorded for 0.0.0.0:0
TLS client handshake failures: 0
TLS server handshake failures: 0

---------- Printing Detailed Connection Report ----------
Note:
** Tuples with no matching socket entry
  - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port>'
    to overcome this error condition
++ Tuples with mismatched address/port entry
  - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port> id <connid>'
    to overcome this error condition

Remote-Agent: 9.13.46.12, Connections-Count: 1
Remote-Port Conn-Id Conn-State WriteQ-Size Local-Address
-----------------------------------------------
5061  1 Established 0 10.64.86.88

Sample output for the `show sip-ua connections tcp tls` command when TLS version is 1.2:

Detail Output

```
router# show sip-ua connections tcp tls detail
Total active connections: 2
No. of send failures: 0
No. of remote closures: 0
No. of conn. failures: 0
No. of inactive conn. ageouts: 0
Max. tls send msg queue size of 1, recorded for 209.165.201.1:5061
TLS client handshake failures: 0
TLS server handshake failures: 0

---------- Printing Detailed Connection Report ----------
Note:
** Tuples with no matching socket entry
  - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port>'
    to overcome this error condition
++ Tuples with mismatched address/port entry
  - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port> id <connid>'
    to overcome this error condition

Remote-Agent: 209.165.201.1, Connections-Count: 2
Remote-Port Conn-Id Conn-State WriteQ-Size Local-Address TLS-Version
-----------------------------------------------
5061  3 Established 0 - TLSv1.2
36289  2 Established 0 - TLSv1.2

Cipher Curve
------------------------ -------------
ECDHE-ECDSA-AES256-GCM-SHA384 P-384
```
Alternatively, the debug ccsip messages command can be used to verify the “Via:” header for TLS is included. This output is a sample INVITE request of a call that uses SIP TLS and the “sips:” URI scheme:

```
INVITE sips:777@172.18.203.181 SIP/2.0
Via: SIP/2.0/TLS 172.18.201.173:5060;branch=z9hG4bK2C419
From: <sips:333@172.18.201.173>;tag=581BB98-1663
To: <sips:5555555@172.18.197.154>
Date: Wed, 28 Dec 2005 18:31:38 GMT
Call-ID: EB5B1948-770611DA-804F9736-BFA4AC35@172.18.201.173
Remote-Party-ID: "Bob" <sips:+14085559999@1.2.3.4>
Contact: <sips:123@host>
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, CONT, REFER, SUBSCRIBE, NOTIFY, INFO
Max-Forwards: 70
Cseq: 104 INVITE
Expires: 60
Timestamp: 730947404
Content-Length: 298
Content-Type: application/sdp

v=0
o=CiscoSystemsSIP-GW-UserAgent 8437 1929 IN IP4 172.18.201.173
s=SIP Call
i=IN IP4 1.1.1.1
n=0
m=audio 18378 RTP/AVP 0 19
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
a=ptime:20
```

**Configuration Examples for SIP TLS Support on CUBE**

**Example: SIP TLS Support on CUBE**

```shell
show running-config
```

Building configuration...
boot system flash:ctestimg
boot-end-marker
!
aqm-register-fnf
!
logging queue-limit 1000
logging buffered 999999
no logging rate-limit
no logging console
!
oo aaa new-model
ethernet lmi ce
clock timezone IST 5 30
!
!
!
!
ip traffic-export profile 1 mode capture
	bidirectional
	incoming access-list 123
	outgoing access-list 123
!
!
!
!
no ip domain lookup
ip cef
no ipv6 cef
!
multilink bundle-name authenticated
!
!
!
!
!
crypto pki trustpoint ecdsacert1
enrollment terminal pem
subject-name cn=plutododsn
revocation-check none
eckeypair myeckey
!
crypto pki trustpoint selfsign
enrollment selfsigned
subject-name cn=plutododsn
revocation-check none
rsakeypair selfsign
!
crypto pki trustpoint ccm155RSA
enrollment terminal
revocation-check none
!
!
crypto pki certificate chain ecdsacert1
certificate 07
30820248 308201CD A0030201 02020107 0200A060 82A8648CE 3D040303 30593112
3100603 5504030C 09706C75 746F3164 6F64310C 000A0603 55040B0C 03544113
310E300C 06035504 06035453 6973636F 310B3009 06035504 06130242 45311830
16060355 040700CF 74757269 6E672D65 7865632D 6C6E7830 1E170D31 35303831
38313235 3431354A 170D3136 30383137 31323534 31345A30 36311330 11060355
0403130A 706C7574 6F6466F4 736E6564 3F301D0609 2A864886 F70D0109 02161063
656E7472 656C6973 65645F72 74722D30 76310106 072A604E CE3D0201 06052881
04002203 62000446 4E28C728 9A66C344 7D6EB2C7 51CE17F3 D125D12B 7043A98B
Example: SIP TLS Support on CUBE

Security

Cisco Unified Border Element Configuration Guide

844
67616C6F 72653082 0122300D 06092A86 4886F70D 01010105 0038201 0F003082 
01EA02B2 010100CC 39112782 D93A3501 8913EBEA 42522D27 E2CS896D2 
8F38F4A5 7CCC2519 9683142A 6B203E9F C7C92673 85D5A940 99B20FB0 CC8F97D1 
F42C1580 D348B831 3BA74AAE0 79AC0C74 E7BFAFCE 4D23F106 3D4EA333 16BA4768 
66CC5561C 5CE19946 DA731D9E 6E743FA0 5F25E445 BE5B6789 64076291 7E5EB0DA 
C482074E 5D6A6841 245E2B54 FE49C090D 85C5EDEC 32E89675 BC934EC3 8C0FC7D8 
02BBC906 93E3698 A8B44527 93A73391 9C71869D BDE9B6BF 06D68AC0 D47D810E 
FCAB3C8F 13BC3D62 02591976 CD49436E 3E2D5B20 079A031E 3FDDEC1C DFBF8261 
CCC5C6AF 7C6FC79C 0234D266 6C508DD7 CC72C8C6 239372F6 7D7CF5CD B56FFB26 
DB422E2 01E15F02 03010001 A34D304B 300B0603 5S110F04 04030202 B4301006 
03551D25 04163014 060B20B6 01050507 03010608 2B060105 05070302 301D0603 
5S11DE04 160412DF 57484974 38D8E8E8 20B15658 9C17F4A0 0D06092A 
864886F7 0D01010B 05000382 01010308 06GFLAC3 C9396667 8A3A0513 5B2CE16 
0DC6BAB5 5B16D7D7 CE68832 592A4270 5FC7EC97 7AFAF2AA 4FA288DD 66A94AB4 
AA66CA7E F974B9B8 63OFAC2B AB95C3BB ECD7A082 AB0343BE 2F89399D AD94DA5 
6B477B44 88F9B94B FEE2E251 4917D0BB 2A5733B5 4F1F5BBA CCC71F0 64365B39 
3F11F8E8F 81A1B71B 61DB51EB C45A2FAD CA743432 A61C19AB E64C4B5F 1E6673A38 
53421ECE 992505BD 5BAA3FA2 954E37EA FE03B725 283A7F19 37A487E9 891E46E0 
BB399050 0902A25 99FDB2A6 2BD3A2E9 74F01C53 EFB3D4D6 654D064E 56878F6C 
21D80184 88C24AD9 E65B78E 12EB78EE 696B9B77 3E73A3FD 10DEBF2D 3CD2BC9 
606700D1 2D42389C EEE43B56 22977A quit 
voice-card 0 
dspfarm 
dsp services dspfarm 
! 
! 
voice service voip 
no ip address trusted authenticate 
allow-connections sip to sip 
fax protocol t38 version 0 ls-redundancy 0 hs-redundancy 0 fallback none 
sip 
bind control source-interface GigabitEthernet0/1 
bind media source-interface GigabitEthernet0/1 
asymmetric payload full 
srtp negotiate cisco 
! 
! 
voice iec syslog 
! 
! 
mta send mail-from username $s$ 
license udi pid CISCO2921/K9 an FGL1538116L 
hw-module pvdm 0/0 
! 
! 
no memory lite 
! 
redundancy 
! 
! 
! 
interface Embedded-Service-Engine0/0 
no ip address 
shutdown 
!
interface GigabitEthernet0/0
ip address 9.45.38.192 255.255.0.0
shutdown
duplex auto
speed auto
!
interface GigabitEthernet0/1
ip address 10.64.86.177 255.255.255.0
ip traffic-export apply 1 size 5000000
duplex auto
speed auto
no clns route-cache
!
interface GigabitEthernet0/2
no ip address
shutdown
duplex auto
speed auto
!
ip forward-protocol nd
!
ip http server
no ip http secure-server
!
ip rtcp report interval 9000
ip route 0.0.0.0 0.0.0.0 10.64.86.1
ip route 10.0.0.0 255.0.0.0 10.64.86.1
!
!
access-list 123 permit udp any any
access-list 123 permit tcp any any
!
control-plane
!
call treatment on
!
!
mgcp behavior rsip-range tgcp-only
mgcp behavior comedia-role none
mgcp behavior comedia-check-media-src disable
mgcp behavior comedia-sdp-force disable
!
mgcp profile default
!
scp local GigabitEthernet0/1
scp ccm 10.64.86.154 identifier 1 version 7.0
!
!
dial-peer voice 1 voip
destination-pattern 6003
session protocol sipv2
session target ipv4:10.64.86.206:5061
session transport tcp tls
incoming called-number 7003
codec g711ulaw
!
dial-peer voice 2 voip
destination-pattern 7003
session protocol sipv2
session target ipv4:10.64.86.206:5061
session transport tcp tls
incoming called-number 6003
codec g711ulaw
!
sip-ua
transport tcp tls v1.2
connection-reuse
crypto signaling default trustpoint ecdsacert1 ecdsa-cipher
!
!
gatekeeper
shutdown
!
!
line con 0
exec-timeout 0 0
speed 115200
line aux 0
line 2
no activation-character
no exec
transport preferred none
transport output pad telnet rlogin lapb-ta mop udptn v120 ssh
stopbits 1
line vty 0 4
login
transport input none
!
end
Example: SIP TLS Support on CUBE
PART XXI

Voice Quality in CUBE

• CUBE Call Quality Statistics Enhancement, on page 851
• Voice Quality Monitoring, on page 857
CHAPTER 69

CUBE Call Quality Statistics Enhancement

Call quality statistics in CUBE, such as packet loss, jitter, and round trip delay can be added to the call detail record (CDR), and these voice metrics can be calculated in IOS. For more information, refer to Voice Quality Enhancements on Cisco Unified Border Element.

The call quality statistics feature is enhanced to provide the following capabilities:

• Enable or disable Quality of Service (QoS) for CUBE.

• Enable or disable Real-time Transport Protocol (RTP) Control Protocol (RTCP) passthrough.

• Configure call quality criteria parameters.

• Feature Information for Call Quality Statistics Enhancement, on page 851
• Restrictions for Call Quality Statistics Enhancement, on page 852
• Information About Call Quality Statistics Enhancement, on page 852
• How to Configure Call Quality Parameters, on page 853
• Configuration Example for Call Quality Statistics, on page 855

Feature Information for Call Quality Statistics Enhancement

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 94: Feature Information for Call Quality Statistics Enhancement

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Quality Statistics</td>
<td>Cisco IOS XE 3.14S</td>
<td>Call quality statistics in CUBE, such as packet loss, jitter, and round trip delay can be added to the call detail record (CDR), and these voice metrics can be calculated in IOS. For more information, refer to Voice Quality Enhancements on Cisco Unified Border Element. The call quality statistics feature is enhanced to provide the following capabilities: • Enable or disable Quality of Service (QoS) for CUBE. • Enable or disable Real-time Transport Protocol (RTP) Control Protocol (RTCP) passthrough. • Configure call quality criteria parameters.</td>
</tr>
</tbody>
</table>

Restrictions for Call Quality Statistics Enhancement

• Only SIP-to-SIP call quality statistics calculation is supported.
• The RTCP field is not recalculated, as it is end-to-end statistics.
• The round trip delay is only retrieved by RTCP, which means the round trip delay is not calculated if there is no related RTCP.
• Only three codec types are supported for one media flow to calculate the jitter; considering the data path performance, these three codecs would be the maximum number in one cache line.
• Only one RTP synchronization source (SSRC) is supported concurrently per media flow, which is indicated in the m-line of the session description protocol (SDP).
• Round trip delay calculation for transcoding calls is not supported.

Information About Call Quality Statistics Enhancement

Call quality configuration parameters include max_dropout, max_reorder, and clock_rate. A maximum of three codecs (codec_number, payload_type, clock_rate) per media flow is collected by the PI and sent to CPP, which uses these values in statistics calculation. Calculated statistics such as Jitter, Packet Loss, and Delay are then fetched from the CPP to the CDR. These statistics can be viewed in the command line interface.

The CDR has the following data per call leg of the call:

• Packet Loss—Calculated based on methods shown in RFC3550. The RTCP sender/receiver reports are recalculated, and not just copied from the inbound leg to the outbound leg.
• Delay—Calculated based on timestamp received or timestamp of packets sent.
• Jitter—Variation of delay.
For more information on how to calculate the voice quality metrics related to media (voice) quality, such as conversational mean opinion score (MOS), jitter, and so on, see http://www.cisco.com/c/en/us/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-cube-call-monitoring.html.

How to Configure Call Quality Parameters

Configuring Call Quality Criteria Parameters

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. call-quality
5. max-dropout number-of-packets
6. max-reorder number-of-packets
7. clock-rate payload-type-number frequency
8. clock-rate dynamic-default frequency
9. exit
10. rtcp all-pass-through
11. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters global VoIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters call quality configuration mode; this is the global call quality of service setup.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# call-quality</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Configures the acceptable out of sequence future packets to drop. The range is from 2 to 2000 packets. The default value is 100.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-serv-call-quality)# max-dropout 300</td>
<td></td>
</tr>
</tbody>
</table>
### Troubleshooting Call Quality Statistics

Use the following `debug` and `show` commands to enable the logs, which helps in debugging:

- `debug ccsip verbose`
- `debug voip fpi all`
- `debug platform hardware qfp active feature sbc dbe datapath all`
- `debug platform hard qfp act feature sbc dbe client all`
- `debug ccsip message`
- `debug ccsip info`
- `show call active voice`
- `show platform hardware qfp active feature sbc data path call call-id`

The following are some show command outputs that would be useful in troubleshooting:

- `Device# show call active voice | include LostPackets`
LostPackets=0

LostPackets=36

Lost packets detail present in show call active voice output. View the complete command output based on the filters such as call-id to check the packet loss for a particular call leg.//

Device# show call active voice | include PlayDelayJitter

PlayDelayJitter=0

PlayDelayJitter=38

Jitter detail present in show call active voice output. View the complete command output based on the filters such as call-id to check the Jitter for a particular call leg.//

Configuration Example for Call Quality Statistics

voice service voip
no ip address trusted authenticate
callmonitor
rtcp all-pass-through
media statistics
media bulk-stats
allow-connections sip to sip
call-quality
max-dropout 2
max-reorder 2
sip
g729 annexb-all
no call service stop
CHAPTER 70

Voice Quality Monitoring

The Voice Quality Monitoring (VQM) feature gives information on the voice quality metrics related to media (voice) quality, such as conversational mean opinion score (MOS), packet loss rate, and so on. VQM enables you to monitor the quality of calls traversing your VoIP network, and you can diagnose the cause of voice quality issues and troubleshoot them.

The Voice Quality Statistics feature provides information about the quality of the Time-Division Multiplexing Internet Protocol (TDM-IP) voice call.

- Feature Information for Voice Quality Monitoring, on page 857
- Prerequisites for Voice Quality Monitoring, on page 858
- Restrictions for Voice Quality Monitoring and Voice Quality Statistics, on page 858
- Information About Voice Quality Monitoring, on page 859
- How to Configure Voice Quality Monitoring, on page 860
- Configuration Examples for Voice Quality Monitoring, on page 863

Feature Information for Voice Quality Monitoring

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 95: Feature Information for Voice Quality Monitoring and Voice Quality Statistics

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice Quality Statistics</td>
<td>Cisco IOS XE Everest 16.5.1b</td>
<td>Voice quality statistics provides information about the quality of the voice TDM-IP call. This feature is already implemented on ISR-G2, and the feature gap is filled in ISR 4000 series.</td>
</tr>
</tbody>
</table>
### Prerequisites for Voice Quality Monitoring

The following commands must be executed to configure the voice quality metrics:

- `callmonitor`
- `rtcp all-pass-through`
- `media statistics`
- `media bulk-stats`
- `call-quality`
  - `max-dropout 2`
  - `max-reorder 2`

### Restrictions for Voice Quality Monitoring and Voice Quality Statistics

- Only SIP-to-SIP call quality statistics calculation is supported.
- The RTCP field is not recalculated, as it is end-to-end statistics.
- The round trip delay is only retrieved by RTCP, which means the round trip delay is not calculated if there is no related RTCP.
- Only three codec types are supported for one media flow to calculate the jitter; considering the data path performance, these three codecs would be the maximum number in one cache line.
- Only one RTP synchronization source (SSRC) is supported concurrently per media flow, which is indicated in the m-line of the session description protocol (SDP).
- Round trip delay calculation for transcoding calls is not supported.
- VQM MOS values are not calculated for DSP based calls.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice Quality Monitoring</td>
<td>Cisco IOS XE Denali 16.3.1</td>
<td>The Voice Quality Monitoring (VQM) feature provides information on the voice quality metrics related to media (voice) quality, such as conversational mean opinion score (MOS), packet loss rate, and so on. VQM enables you to monitor the quality of calls traversing your VoIP network, and you can diagnose the cause of voice quality issues and troubleshoot them.</td>
</tr>
</tbody>
</table>
Information About Voice Quality Monitoring

The VQM (Voice Quality Monitor) gives information on the voice quality metrics. The VQM on Cisco IOS XE platforms enables statistics gathering based on the received RTCP packets. From these statistics, a voice quality measurement is developed to show the quality of the call. The output is in a simple format, using a system of good, poor, and bad types of ratings.

The following metrics exist in Call Detail Record (CDR) and Management Information Base (MIB) in CUBE, indicating voice quality:

1. MOSQe (conversational quality MOS)
2. Round-trip-delay.
4. Packet-Loss-Rate.

The CDR is sent at the end of a call if AAA accounting is configured.

A CDR example is as follows:

```xml
<MOS-Con>4.4072</MOS-Con>
<round-trip-delay>1 ms</round-trip-delay>
<receive-delay>64 ms</receive-delay>
<voice-quality-total-packet-loss>0.0000 %</voice-quality-total-packet-loss>
```

VQM Metrics

The following are the metrics exported by VQM:
### How to Configure Voice Quality Monitoring

**Enabling Media Statistics Globally**

Perform this task to globally enable media statistics in voice-service configuration mode to estimate the values for packet loss, jitter, and round-trip time.

#### SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. media statistics
5. end

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable&lt;br&gt;Example:&lt;br&gt;Device&gt; enable</td>
<td>Enables privileged EXEC mode.&lt;br&gt;• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal&lt;br&gt;Example:&lt;br&gt;Device# configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Step 3</td>
<td>voice service voip&lt;br&gt;Example:</td>
<td>Enters voice service VoIP configuration mode.</td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config)# voice service voip</td>
<td>Enables media statistics to estimate the values of packet loss, jitter, and Round Trip Time (RTT) statistics.</td>
</tr>
<tr>
<td><strong>Step 4</strong> media statistics</td>
<td><strong>Example:</strong> Device(conf-voi-serv)# media statistics</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td><strong>Example:</strong> Device(conf-voi-serv)# end</td>
</tr>
</tbody>
</table>

---

**Verifying Voice Quality Monitoring**

Perform this task to verify the configuration for voice quality monitoring. The *show* commands can be entered in any order.

#### SUMMARY STEPS

1. enable
2. show call active voice | include LostPackets
3. show call active voice | include ReceiveDelay
4. show call active voice brief | sec RTT
5. show call active voice stats | sec MC

#### DETAILED STEPS

**Step 1** enable

Enables privileged EXEC mode.

**Example:**

Device> enable

**Step 2** show call active voice | include LostPackets

Displays statistics on the CUBE if the Voice Quality Metrics feature is configured.

**Example:**

Device# show call active voice | include LostPackets

LostPackets=0
LostPackets=0

**Step 3** show call active voice | include ReceiveDelay
Verifying Voice Quality Monitoring

Displays statistics on the CUBE if the Voice Quality Metrics feature is configured.

Example:

```
Device# show call active voice | include ReceiveDelay
ReceiveDelay=0
ReceiveDelay=0
```

Step 4 **show call active voice brief | sec RTT**

Displays a truncated version of call information for voice calls in CUBE if the Voice Quality Metrics feature is configured.

Note: This command is not applicable for TDM-IP voice calls.

Example:

```
Device# show call active voice brief | sec RTT
IP 173.39.65.81:7078 SRTP: off rtt:12ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g711ulaw
TextRelay: off Transcoded: No ICE: Off
IP 10.127.17.141:18920 SRTP: off rtt:12ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g711ulaw
TextRelay: off Transcoded: No ICE: Off
```

Step 5 **show call active voice stats | sec MC**

Displays R-Factor Statistics (G.107 MOS) on the CUBE if the Voice Quality Metrics feature is configured. A sample output is provided below for a voice call using G.711ulaw, VAD on, and at 5 percent packet loss rate.

Example:

```
Device# show call active voice stats | sec MC
DSP/RF: ML=, MC=, R1=, R2=, IF=, ID=, IE=, BL=, R0=, VR=
DSP/RF: ML=4.2346, MC=4.2346, R1=92, R2=92, IF=0, ID=0, IE=0, BL=0, R0=93, VR=2.0
The following is an example output for the SNMP MIB:
cmqVoIPCallActiveRxPred107RMosConv.8520964.1 = 423 (MC)
```

For more information on the SNMP MIB "cmqVoIPCallActiveRxPred107RMosConv", see SNMP Object Navigator.

In the sample output, the following can be noted:

- ML for codec G.711ulaw is 4.2346.
- MC for codec G.711ulaw is 4.2346.
- IE for codec G.711 is 0.
- R0 is 93.

The following table defines the abbreviations used in the sample output.

<table>
<thead>
<tr>
<th>Type</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP/RF</td>
<td>ML, MC, R1, R2, IF, ID, IE, BL, R0, VR</td>
<td></td>
</tr>
<tr>
<td>DSP/RF</td>
<td>ML=4.2346, MC=4.2346, R1=92, R2=92, IF=0, ID=0, IE=0, BL=0, R0=93, VR=2.0</td>
<td></td>
</tr>
</tbody>
</table>
### Troubleshooting Tips

Use the following debug commands to troubleshoot the Voice Quality Monitoring feature:

- `debug voip rtp packets`
- `debug performance monitor`
- `debug radius accounting`
- `debug aaa accounting`

### Configuration Examples for Voice Quality Monitoring

**Example: Configuring Media Statistics Globally**

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(config-voi-serv)# media statistics
Device(config-voi-serv)# end
```

**Example: CDR Enabled MOS Output**

At the end of a call, the MOSQe output is displayed in CDR only if the `debug radius accounting` is enabled. The `show log | sec MOS-Con` command displays the MOS-Con value as shown below:

```
Device# show log | sec MOS-Con
```
Example: CDR Enabled MOS Output

PART XXII

Smart Licensing

- Cisco Smart Software Licensing, on page 867
Cisco Smart Software Licensing

Cisco Smart Software Licensing provides a simple cloud-based solution for managing and tracking the use of your licenses and entitlements across your business.

When purchased, licenses are automatically delivered to your company Smart Account, ready for use. Devices report license use by registering to your Smart Account either directly across the internet, through an internet proxy, or by using a Smart Software Manager satellite to mediate reporting from devices across the business.

Cisco Smart Software Manager (CSSM) is the secure, cloud-based portal used to:

• Assign licenses to business entities using Virtual Accounts.

• Register devices to Virtual Accounts.

• Manage license inventory and monitor use.

• Track device authorization for feature use.

For more information on Cisco Smart Licensing, see http://www.cisco.com/go/smartlicensing.

• Feature Information for CUBE Smart Licensing, on page 867
• Registration and License Compliance, on page 868
• Smart Software Licensing Task Flow for CUBE, on page 868
• Verify Smart License Operation for CUBE, on page 871
• CUBE High Availability (HA) Configurations, on page 875

Feature Information for CUBE Smart Licensing

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco Smart Licensing enhancements for CUBE</td>
<td>IOS XE Gibraltar Release 16.11.1a</td>
<td>License requirements for the use of CUBE trunk sessions are reported to Cisco Smart Licensing.</td>
</tr>
</tbody>
</table>
Registration and License Compliance

Devices are required to register with the Cisco Smart Software Manager or satellite to report feature usage and compliance with purchased entitlements. The Smart Agent running on the CUBE platform transitions between the following operational states to control the use of licensed features:

Smart Agent Licensing States

- **Unregistered / Evaluation Mode**—When the platform is first used, the Smart Agent is in the unregistered state. In this state, licensed features may be used freely during a 90-day evaluation period.

- **Unregistered / Evaluation Expired**—If the Smart Agent has not successfully registered by the end of the evaluation period, licensed functionality, including CUBE features, are disabled.

- **Registered**—To enable licensed features, the platform must be registered to CSSM or satellite using a registration ID token. Once registered, the Smart Agent receives an Identity Certificate that is saved securely and used for all ongoing communications. The certificate is valid for one year and will be renewed automatically after six months to ensure continuous operation. While registered, the Smart Agent maintains one of the three authorization states relating to each license type by periodically reporting license usage to the CSSM or satellite:
  
  - **Authorized**—If the associated Smart Account has enough licenses to reserve for the platform, it indicates to the Smart Agent that the features are fully authorized for use. The Smart Agent renews the authorization every 30 days.

  - **Out of Compliance**—If the associated Smart Account does not have enough licenses to reserve for the platform, it will indicate to the Smart Agent that the current feature use is Out of Compliance. Currently, CUBE functionality is not limited in this state.

  - **Authorization Expired**—If the Smart Agent is not able to communicate with the CSSM or satellite for 90 days or more, authorization expires. Currently, CUBE functionality is not limited in this state.

  - **Registration Expired**—If the Smart Agent is not able to contact the CSSM or satellite before the Identity Certificate expires, the platform returns to the unregistered state. The platform resumes the remaining evaluation period, only if available. After registration expires, registration to the CSSM or satellite using a new registration ID token is required.

Note

Each platform provides a single 90-day evaluation period. This period is persistent across the reloads and is based on the router's uptime.

Smart Software Licensing Task Flow for CUBE

The process for using and validating Smart Licensing for CUBE is common to the host platform. Tasks are summarized to highlight requirements specific to the CUBE application.

Task Flow:
1. Obtain a registration ID token from the Virtual Account to which you wish to register the platform. See Obtain the Registration ID Token, on page 869.

2. If using an internet proxy or Smart Software Manager satellite, update the Smart Licensing transport settings. See Configure Smart Licensing Transport Settings, on page 869.

3. Register the Host Platform with CSSM or Satellite, on page 870.

4. Configure CUBE Licensed Features, on page 870.

Obtain the Registration ID Token

Detailed Steps

1. Log in to your Smart Account in either CSSM or satellite.
2. Navigate to the Virtual Account with which you want to register the CUBE.
3. Generate a registration ID token.

Configure Smart Licensing Transport Settings

Step 1  hostname  hostname

Example:
Device(config)# hostname cube
cube(config)#

Ensure that hostname and the PID of the platform are not the same. For example, if the hostname of an ASR1006 router is configured as "ASR1006", registration will be unsuccessful.

Step 2  ip name-server  IP Address

Example:
cube(config)# ip name-server 10.0.0.1 10.0.0.10

Configures valid DNS servers to ensure the correct resolution of the CSSM or satellite hostname.

Step 3  ip http client source-interface  interface name

Example:
cube(config)# ip http client source-interface GigabitEthernet0/0/0

Binds the platform HTTP client to the interface used to access the CSSM or satellite.

Step 4  call-home

Example:
cube(config)# call-home

Access call-home configuration to change the internet proxy and license server URL if necessary.

Step 5  http-proxy  hostname proxy-port  port-number
Example:
```
cube(cfg-call-home)# http-proxy proxy.mybiz.com proxy-port 80
```
If necessary, configure a proxy-server for the platform when a direct HTTP connection to CSSM is not permitted.

**Step 6**

**profile**  
```
cube(cfg-call-home)# profile CiscoTAC-1
```
If registering to a Smart Software Manager satellite, edit the default profile (CiscoTAC-1) to configure the registration URL.

**Step 7**

**destination address http url**  
```
cube(cfg-call-home-profile)# destination address http http://ssat.mybiz.com /Transportgateway/services/DeviceRequestHandler
```
If registering to a satellite, configure the registration URL. The default configuration must be used when registering directly to CSSM.

---

**Register the Host Platform with CSSM or Satellite**

To report license usage, the host platform must be registered to either CSSM or satellite. The platform remains in evaluation mode until registered.

**Before you begin**

1. Obtain the registration ID token from your Smart Account.
2. Configure Smart Licensing transport settings.

```
license smart register id_token id_token
```

Example:
```
Device# license smart register id_token XXXXXXXXXTnVhaUZlRHorQjJERT0%3D
```
 Registers the CUBE with the CSSM or satellite using a registration ID token that you obtain from your Smart Account.

---

**Configure CUBE Licensed Features**

**Step 1**

**voice service voip**

Example:
```
cube(config)# voice service voip
```
Enters global VoIP configuration mode.
Step 2  
mode border-element license capacity sessions

Example:
Device(cfg)# mode border-element license capacity 25

Enables the specified number of licenses for CUBE.

## Verify Smart License Operation for CUBE

Use the following commands to verify the platform registration and license usage:

- **show cube status**—Displays CUBE license capacity and a count of calls that are blocked when the evaluation period has expired.

  ```
cube#show cube status
CUBE-Version : 12.5.0
SW-Version : 16.11.1, Platform CSR1000V
HA-Type : none
Licensed-Capacity : 10
Calls blocked (Smart Licensing Not Configured) : 0
Calls blocked (Smart Licensing Eval Expired) : 0
  ```

- **show license status**—Displays the platform registration and authorization status.

  ```
cube#show license status
Smart Licensing is ENABLED
...
Type: Callhome
Registration:
  Status: REGISTERED
  Smart Account: BU Production Test
  Virtual Account: CUBE Sat Test
  Export-Controlled Functionality: Allowed
  Initial Registration: SUCCEEDED on Feb 18 12:57:04 2019 UTC
  Last Renewal Attempt: None
  Next Renewal Attempt: Aug 17 12:57:03 2019 UTC
  Registration Expires: Feb 18 12:51:49 2020 UTC
License Authorization:
  Status: AUTHORIZED on Mar 04 15:11:54 2019 UTC
  Last Communication Attempt: SUCCEEDED on Mar 04 15:11:54 2019 UTC
  Next Communication Attempt: Apr 03 15:11:53 2019 UTC
  Communication Deadline: Jun 02 15:06:21 2019 UTC
...
  ```

- **show license usage**—Displays the license usage and authorization status.

  ```
cube#show license usage
License Authorization:
  Status: AUTHORIZED on Mar 04 15:11:54 2019 UTC

CSR 1KV APPX 500M (appx_500M):
  Description: CSR 1KV APPX 500M
  Count: 1
  Version: 1.0
  Status: AUTHORIZED
  Export status: NOT RESTRICTED
  ```
CUBE_Trunk_Standard_Session (CUBE_T_STD):
Description: Cisco Unified Border Element (CUBE) Trunk Standard Session License
Count: 10
Version: 1.0
Status: AUTHORIZED
Export status: NOT RESTRICTED

• show license summary—Displays a summary of registration and license usage

Device#show license summary
Smart Licensing is ENABLED

Registration:
  Status: REGISTERED
  Smart Account: BU Production Test
  Virtual Account: CUBE Sat Test
  Export-Controlled Functionality: Allowed
  Last Renewal Attempt: None
  Next Renewal Attempt: Aug 17 12:57:04 2019 UTC

License Authorization:
  Status: AUTHORIZED
  Last Communication Attempt: SUCCEEDED
  Next Communication Attempt: Apr 03 15:11:54 2019 UTC

License Usage:
  License Entitlement tag Count Status
  CUBE_Trunk_Standard_Session (CUBE_T_STD) 10 AUTHORIZED

• show license all—Displays all the information that is related to licensing.

Device#show license all
Smart Licensing Status

Smart Licensing is ENABLED

Registration:
  Status: REGISTERED
  Smart Account: BU Production Test
  Virtual Account: CUBE_VA
  Export-Controlled Functionality: Allowed
  Initial Registration: SUCCEEDED on May 21 07:15:09 2018 IST
  Last Renewal Attempt: None
  Next Renewal Attempt: Nov 17 07:15:08 2018 IST
  Registration Expires: May 21 07:09:26 2019 IST

License Authorization:
  Status: AUTHORIZED on May 21 07:22:09 2018 IST
  Last Communication Attempt: SUCCEEDED on May 21 07:22:09 2018 IST
  Next Communication Attempt: Jun 20 07:22:08 2018 IST
  Communication Deadline: Aug 19 07:16:27 2018 IST

Utility:
  Status: DISABLED

Data Privacy:
  Sending Hostname: yes
  Callhome hostname privacy: DISABLED
  Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED
Transport:
 Type: Callhome

License Usage
 =============

CUBE_Standard_Session (CUBE_T_STD):
 Description: Cisco Unified Border Element (CUBE) Standard Session License
 Count: 5
 Version: 1.0
 Status: AUTHORIZED

Product Information
 ====================

UDI: PID:CSR1000V, SN:9RTYVZ9LKZ4

Agent Version
 =============

Smart Agent for Licensing: 4.4.4_rel/66
Component Versions: SA:(1_3_dev)1.0.15, SI:(dev22)1.2.1, CH:(rel5)1.0.3, PK:(dev 18)1.0.3

Reservation Info
 ================

License reservation: DISABLED

• show license tech support—Displays the license technical support information.

Device#show license tech support
Smart Licensing Tech Support info

Smart Licensing Status
 =========================

Smart Licensing is ENABLED

Registration:
 Status: REGISTERED
 Smart Account: BU Production Test 1
 Virtual Account: CUBE_VA
 Export-Controlled Functionality: Allowed
 Initial Registration: SUCCEEDED on May 21 07:15:09 2018 IST
 Last Renewal Attempt: None
 Next Renewal Attempt: Nov 17 07:15:08 2018 IST
 Registration Expires: May 21 07:09:26 2019 IST

License Authorization:
 Status: AUTHORIZED on May 21 07:22:09 2018 IST
 Last Communication Attempt: SUCCEEDED on May 21 07:22:09 2018 IST
 Next Communication Attempt: Jun 20 07:22:08 2018 IST
 Communication Deadline: Aug 19 07:16:27 2018 IST

Utility:
 Status: DISABLED

Data Privacy:
 Sending Hostname: yes
 Callhome hostname privacy: DISABLED
 Smart Licensing hostname privacy: DISABLED
 Version privacy: DISABLED

Transport:
 Type: Callhome
Evaluation Period:
   Evaluation Mode: EXPIRED
   Evaluation Period Remaining: Expired on May 10 01:44:42 2018 IST

License Usage
--------------
Handle: 1
   License: CUBE_Standard_Session
   Entitlement tag:
   regid.2018-03.com.cisco.CUBE_T_STD,1.0_20e2edcc-6fbd-4b7e-be8c-d6d46f37473f
   Description: Cisco Unified Border Element (CUBE) Standard Session License
   Count: 5
   Version: 1.0
   Status: AUTHORIZED(3)
   Status time: May 21 07:22:03 2018 IST
   Request Time: May 21 07:20:53 2018 IST

Product Information
-------------------
UDI: PID:CSR1000V,SN:9RTYVZ9LKZ4

Agent Version
-------------
Smart Agent for Licensing: 4.4.4_rel/66
Component Versions: SA:(1_3_dev)1.0.15, SI:(d

• show call-home smart-licensing statistics—Displays the details of the data that is exchanged between
  the CUBE and the CSSM.

Device#show call-home smart-licensing statistics
Success: Successfully sent and response received.
Failed : Failed to send or response indicated error occurred.
Inqueue: In queue waiting to be sent.
Dropped: Dropped due to incorrect call-home configuration.

<table>
<thead>
<tr>
<th>Msg Subtype</th>
<th>Success</th>
<th>Failed</th>
<th>Inqueue</th>
<th>Dropped</th>
<th>Last-sent (GMT+05:30)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTITLEMENT</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2018-05-18 15:04:30</td>
</tr>
<tr>
<td>DEREGISTRATION</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2018-05-18 14:51:45</td>
</tr>
<tr>
<td>REGISTRATION</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2018-05-18 14:57:17</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2018-05-18 14:57:22</td>
</tr>
</tbody>
</table>

• show call-home smart-licensing—Displays the destination URL that is configured.

Device#show call-home smart-licensing
Current smart-licensing transport settings:
   Smart-license messages: enabled
   Profile: CiscoTAC-1 (status: ACTIVE)
   Destination URL(s): https://tools.cisco.com/its/service/oddce/services/DDCEService

Use the following commands to debug any issues that are related to your Smart License:

• debug license feature cube all
• debug smart_lic all
• debug call-home smart-licensing all
CUBE High Availability (HA) Configurations

Smart Licensing with CUBE Box-to-Box High Availability (HA)

Box-to-Box redundancy uses the Redundancy Group (RG) Infrastructure to form an Active/Standby pair of routers.

For Smart License configurations on the Active/Standby pair of router platforms, see Smart Software Licensing Task Flow for CUBE, on page 868. When reporting license usage, the Smart Agent includes details of its High Availability group and, if it is in the Active or Standby state. Thus allowing the CSSM or satellite to group license requirements for the High Availability pair.

CUBE Box-to-Box High Availability requires CUBE Trunk Redundant Session licenses.

Before Failover

- Register both the platforms in the High Availability configuration to the same Smart Virtual Account on CSSM or satellite.

- The CSSM or satellite authorizes license usage requests for both Active and Standby platforms.

- Only licenses that are requested by the Active server are reserved from the license pool.

After Failover

- The platform that switches to Active mode reports the license usage to the CSSM or satellite.

- The CSSM or satellite switches the license reservation to the new Active server, ensuring that no additional licenses are used from the Smart Account during failover.

- When the originally Active platform recovers, it is authorized as a Standby server without consuming more licenses from the Smart Account.

Verify Smart License Operation for Box-to-Box High Availability (HA)

You can use all the commands that are given in the section Verify Smart License Operation for CUBE, on page 871 to verify the licensing status in High Availability mode. The following commands specifically reflect Smart License information that is related to Box-to-Box High Availability (HA):

- `show cube status`—Displays CUBE license capacity and High Availability mode.

```
cube-1# show cube status
CUBE-Version : 12.5.0
SW-Version : 16.11.1, Platform CSR1000V
HA-Type : hot-standby-chassis-to-chassis
Licensed-Capacity : 10
Calls blocked (Smart Licensing Eval Expired) : 0
```

- `show license usage`—Displays license usage and authorization status

```
cube-1# show license usage
CUBE_Trunk_Standard_Session (CUBE_T_RED):
    Description: Cisco Unified Border Element (CUBE) Trunk Redundant Session License
    Count: 10
```
• show license summary—Displays the license summary information.

Following is the sample output from the Active instance of CUBE.

Device_CUBE1# show license summary
Smart Licensing is ENABLED

Registration:
Status: REGISTERED
Smart Account: BU Production Test 1
Virtual Account: CUBE_VA
Export-Controlled Functionality: Allowed
Last Renewal Attempt: None
Next Renewal Attempt: Jan 07 22:10:20 2019 IST

License Authorization:
Status: AUTHORIZED
Last Communication Attempt: SUCCEEDED
Next Communication Attempt: Aug 12 10:02:43 2018 IST

License Usage:
<table>
<thead>
<tr>
<th>License</th>
<th>Entitlement tag</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBE_Redundant_Session (CUBE_T_RED)</td>
<td></td>
<td>20</td>
<td>AUTHORIZED</td>
</tr>
</tbody>
</table>

• show license all—Displays Active and Standby states.

cube-1# show license all
CUBE_Redundant_Session (CUBE_T_RED):
Description: Cisco Unified Border Element (CUBE) Redundant Session License
Count: 10
Version: 1.0
Status: AUTHORIZED
Application HA Info:
Application Name: CUBE
Application Id: 1
Application Role: Active

cube-2# show license all
CUBE_Redundant_Session (CUBE_T_RED):
Description: Cisco Unified Border Element (CUBE) Redundant Session License
Count: 10
Version: 1.0
Status: AUTHORIZED
Application HA Info:
Application Name: CUBE
Application Id: 1
Application Role: Standby

• show license tech support—Displays the details of the application HA and the High Availability pair.

cube-1# show license tech support
Smart Licensing Tech Support info
Smart Licensing Status
-----------------------
Smart Licensing is ENABLED
...
License Usage
---------
Handle: 1
License: CUBE_Redundant_Session
Entitlement tag:
regid.2018-03.com.cisco.CUBE_T_RED,1.0_c3de23c1-4e60-4342-a25f-021d709a462c
Description: Cisco Unified Border Element (CUBE) Redundant Session License
Count: 20
Version: 1.0
Status: AUTHORIZED(3)
Status time: Jul 13 10:02:44 2018 IST
Request Time: Jul 13 10:02:11 2018 IST
Application HA Info:
Application Name: CUBE
Application Id: 1
Application Role: Active
Info For Peer: 227{AHa1Ver}{1}{AHa1Name}{cube-1}{AHa1AppID}{1}{AHa1Role}{Active}
{AHa1PHost}{cube-1}{AHa1PUDI}{P:CSR1000V,S:9ECXABN104R}
{AHa1PPIID}{0826fdff-71bf-4af6-a998-c1bc8091b06c}{AHa1PSAcc}
{BU Production Test 1}{AHa1PVAcc}{CUBE_VA}
Peer Info:
Application Name: CUBE
Application Id: 1
Application Role: Standby
Hostname: cube-2
PIID: e8107548-2851-42cf-bb44-33479104ae0
UDI: P:CSR1000V,S:9RTYVZ9LZK4
Smart Account Name: BU Production Test 1
Virtual Account Name: CUBE_VA
Info From Peer: 228{AHa1Ver}{1}{AHa1Name}{CUBE}{AHa1AppID}{1}{AHa1Role}{Standby}
{AHa1PHost}{cube-2}{AHa1PUDI}{P:CSR1000V,S:9RTYVZ9LZK4}
{AHa1PPIID}{e8107548-2851-42cf-bb44-33479104ae0}{AHa1PSAcc}
{BU Production Test 1}{AHa1PVAcc}{CUBE_VA}
...

Smart Licensing with CUBE Inbox High Availability (HA)

You can configure an ASR1000 router platform with two Route Processors for Inbox High Availability using Stateful Switchover (SSO). In this configuration, one Route Processor is active while the other is in standby mode.

For the Smart License configuration, see Smart Software Licensing Task Flow for CUBE, on page 868. Only the Active Route Processor in the SSO configuration reports license usage, so CSSM reserves one set of licenses for the platform.

CUBE Inbox High Availability requires CUBE Trunk Standard Session licenses.

Before Failover

- Smart License configuration is synchronized between the two Route Processors. Only the Active Route Processor registers with CSSM or satellite.

- The CSSM or satellite authorizes license usage requests for the Active Route Processor.
After Failover

- The Route Processor that switches to active mode, reports license usage to the CSSM or satellite.
- As the new report appears to come from the same device, the CSSM or satellite retains the original reservation for the platform.

Verify Smart License Operation for Inbox High Availability (HA)

You can use all the commands that are given in the section Verify Smart License Operation for CUBE, on page 871 to verify the licensing status in the High Availability mode. The following commands specifically reflect Smart License information that is related to Inbox High Availability (HA):

- `show cube status`—Displays CUBE license capacity and High Availability mode.

```plaintext
cube-1# sh cube status
CUBE-Version : 12.5.0
SW-Version : 16.11.1, Platform Platform ASR1006
HA-Type : hot-standby-card-to-card
Licensed-Capacity : 10
Calls blocked (Smart Licensing Not Configured) : 0
Calls blocked (Smart Licensing Eval Expired) : 0
```

- `show redundancy states`—Displays the redundancy state of the route processors.

```plaintext
cube-1# show redundancy states
my state = 13 -ACTIVE
peer state = 8 -STANDBY HOT
Mode = Duplex
Unit = Secondary
Unit ID = 49
Redundancy Mode (Operational) = sso
Redundancy Mode (Configured) = sso
Redundancy State = sso
Maintenance Mode = Disabled
Manual Swact = enabled
Communications = Up
client count = 131
client_notification_TMR = 30000 milliseconds
RF debug mask = 0x0
```

```plaintext
cube-2# show redundancy states
my state = 8 -STANDBY HOT
peer state = 13 -ACTIVE
Mode = Duplex
Unit = Primary
Unit ID = 48
Redundancy Mode (Operational) = sso
Redundancy Mode (Configured) = sso
Redundancy State = sso
Maintenance Mode = Disabled
Manual Swact = cannot be initiated from this the standby unit
Communications = Up
client count = 131
client_notification_TMR = 30000 milliseconds
RF debug mask = 0x0
```

- `show license summary`—Displays license summary information.
cube-1# show license summary
Smart Licensing is ENABLED
Registration:
Status: REGISTERED
Smart Account: BU Production Test 1
Virtual Account: CUBE VA
Export-Controlled Functionality: Allowed
Last Renewal Attempt: None
Next Renewal Attempt: Jan 02 09:06:22 2019 IST
CUBE Smart Licensing
License Authorization:
Status: Authorized
Last Communication Attempt: SUCCEEDED
Next Communication Attempt: Aug 02 00:48:00 2018 IST
License Usage:
License | Entitlement tag | Count | Status
--------------------------------------------
ASR_1000_AdvEnterprise (ASR_1000_AdvEnterprise) | 1 | AUTHORIZED
CUBE_Standard_Session (CUBE_T_STD) | 10 | AUTHORIZED

cube-2# show license summary
Smart Licensing is ENABLED
Registration:
Status: REGISTERED
Smart Account: BU Production Test 1
Virtual Account: CUBE VA
Export-Controlled Functionality: Allowed
Last Renewal Attempt: None
License Authorization:
Status: Authorized
Last Communication Attempt: SUCCEEDED
Next Communication Attempt: None
License Usage:
License | Entitlement tag | Count | Status
--------------------------------------------
ASR_1000_AdvEnterprise (ASR_1000_AdvEnterprise) | 1 | PENDING
Verify Smart License Operation for Inbox High Availability (HA)
PART XXIII

Serviceability

• Support for Session Identifier, on page 883
Support for Session Identifier

Cisco Unified Border Element (CUBE) supports “Session Identifier” for end-to-end tracking of a SIP session in IP-based multimedia communication systems. Support for session identifier is in compliance with RFC 7206 and draft-ietf-insipid-session-id-15.

- Feature Information for Session Identifier Support, on page 883
- Restrictions, on page 884
- Information About Session Identifier, on page 884
- Configuring Support for Session Identifier, on page 885
- Troubleshooting Tips, on page 885

Feature Information for Session Identifier Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 97: Feature Information for Session Identifier Support

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Support for Session Identifier    | Cisco IOS 15.6(2)T Cisco IOS XE Denali 16.3.1 | This feature enables CUBE to support “Session Identifier” for end-to-end tracking of a SIP session in IP-based multimedia communication systems in compliance with RFC 7206 and draft-ietf-insipid-session-id-15. A new keyword session-id is added to the following commands:  
  - show call active voice  
  - show call active video  
  - show call history voice  
  - show call history video  
  - show call active voice brief  
  - show call active video brief |

Restrictions

- Session Identifier is not supported for SIP-H.323, H.323-SIP, and H.323-H.323 calls.

Information About Session Identifier

CUBE supports “Session Identifier” that overcomes the limitations with the existing call-identifiers and allows end-to-end tracking of a SIP session. To support session identifier, “Session-ID” header is added in the SIP request and response messages.

Note "Session Identifier" refers to the value of the identifier, whereas "Session-ID" refers to the header field used to convey the identifier.

The Session-ID comprises of Universally Unique Identifier (UUID) for each user agent participating in a call. Each call consists of two UUID known as local UUID and remote UUID. Local UUID is the UUID generated from the originating user agent and remote UUID is generated from the terminating user agent. The UUID values are presented as strings of lower-case hexadecimal characters, with the most significant octet of the
UUID appearing first. Session Identifier comprises of 32 characters and remains same for the entire session. Refer to RFC 4122 for more information on UUID.

**Example for Session ID header**

Session-ID: ab30317f1a784dc48ff824d0d3715d86; remote=47755a9de7794ba387653f2099600ef2

In the above example:

Local UUID = ab30317f1a784dc48ff824d0d3715d86

Remote UUID = 47755a9de7794ba387653f2099600ef2

### Feature Behavior

- If all the user agents associated with CUBE support session-id, then CUBE allows pass-through of the Session ID header in all SIP request and response messages for the session.

- CUBE looks for the Session ID header present in the SIP messages and validates the SessionID header syntax as defined in draft-ietf-insipid-session-id-15. Session ID format earlier to draft-ietf-insipid-session-id-15 is considered as unsupported.

- If some of the user agents do not support session ID, CUBE generates local UUID on behalf of the user agent and sends the generated UUID in SIP request and response. CUBE generates UUID based on version 5 (SHA-1).

- If a Session ID is received in the format as defined in RFC 7329, CUBE considers it as unsupported. CUBE generates local UUID on behalf of the user agent and sends the generated UUID in SIP request and response.

- In a mid call scenario, where user a session is switched from supporting session identifier to non-supporting session identifier, CUBE saves the previous non-NULL session identifier and sends the saved non-NULL session identifier in re-invite messages as needed.

- For high availability, session ID is check pointed in active and re-created in standby.

### Configuring Support for Session Identifier

Session Identifier support is enabled on CUBE by default. No additional configuration required.

### Troubleshooting Tips

The following show commands helps you to troubleshoot any issues with session identifier.

- **show call active voice session-id** `WORD`
• `show call active voice session-id` *WORD*
• `show call active video session-id` *WORD*
• `show call active video brief session-id` *WORD*

*WORD* can be complete session identifier (local, remote, or both), or wildcard pattern of local or remote UUID. The valid wildcard patterns for search are *, 0-9, a-f, A-F.

The following session identifier is considered in the below examples:

```
SessionIDLocaluuid=db248b6c6cbdc547bbc6c6fdff6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
```

**Valid Search Patterns**

You can search for the session identifier using complete Session ID header as shown below:

```
Device# show call active voice session-id db248b6c6cbdc547bbc6c6fdff6916eeb; remote=4fd24d9121935531a7f8d750ad16e19
```

Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=db248b6c6cbdc547bbc6c6fdff6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 1

You can search for the session identifier using complete local UUID as shown below:

```
Device# show call active voice session-id db248b6c6cbdc547bbc6c6fdff6916eeb
```

Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=db248b6c6cbdc547bbc6c6fdff6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 1

You can search for the session identifier using complete remote UUID as shown below:

```
Device# show call active voice session-id 4fd24d9121935531a7f8d750ad16e19
```

Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
.
.
.
.
.
.
.
.
SessionIDLocaluuid=db248b6cbdc547bcb6c6f6f69d6eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19

SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 1

You can search for session id using wildcard pattern match as shown below:

Device# Device# show call active voice session-id 4fd2*
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0

Device# show call active voice session-id *f*16e*
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0

Device# show call active voice brief session-id *
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# `show call active voice session-id *; remote=`
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0

SessionIDLocaluuid=4fd24d9121935531a7f8d750ad16e19
SessionIDRemoteuuid=db248b6cbdc547bbc66f6db6916eeb

SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# `show call active voice session-id 4fd24d9*; remote=*16eeb`
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0

SessionIDLocaluuid=4fd24d9121935531a7f8d750ad16e19
SessionIDRemoteuuid=db248b6cbdc547bbc66f6db6916eeb

Invalid Search Patterns
The following wildcard search patterns are invalid:

Device# `show call active voice session-id ;remote=`
Invalid Pattern. Pattern can have a string with `^[0-9a-fA-F]*$` only OR a string with `^[0-9a-fA-F]*;remote=[0-9a-fA-F]*$`.

Device# `show call active voice session-id *;remote=*`
Invalid Pattern. Pattern can have a string with `^[0-9a-fA-F]*$` only OR a string with `^[0-9a-fA-F]*;remote=[0-9a-fA-F]*$`.

Device# `show call active video session-id ;remote=*`
Incorrect format for Session-ID Wildcard Pattern regular expression must be of the form `^[0-9A-Fa-f]*$`
Invalid Pattern. Pattern can have a string with `^[0-9a-fA-F]*$` only OR a string with `^[0-9a-fA-F]*;remote=[0-9a-fA-F]*$`.

Device# `show call active voice session-id 4fd24d9*remote=*16eeb`
Incorrect format for Session-ID Wildcard Pattern regular expression must be of the form `^[0-9a-fA-F]*$`
Invalid Pattern. Pattern can have a string with `^[0-9a-fA-F]*$` only OR a string with...
Search using Null session identifier

If one of the session identifier is null, you can search for the session identifiers using 0 as wildcard pattern. The following session identifier is considered in the below example:

```
SessionIDLocaluuid=00000000000000000000000000000000
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e1e9
```

```
Device# show call active voice session-id 0
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
.
.
SessionIDLocaluuid=00000000000000000000000000000000
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e1e9
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2
```

Correlation between Session Identifier and Call Identifier

The following session identifier is considered in the below examples:

```
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdefe69f76eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e1e9
```

You can search for session identifier using the local UUID as shown below:

```
Device# show call active voice session-id db248b6cbdc547bbc6c6fdefe69f76eeb
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
.
.
VOIP:
ConnectionId[0x8CDAC180 0x10000 0x1B7 0x5B56400A]
IncomingConnectionId[0x8CDAC180 0x10000 0x1B7 0x5B56400A]
GlobalCallId=[0xC3DAB665 0x770C11E5 0x80318550 0x5A000ED7]
CallReferenceId=0
CallServiceType=Unknown
RTP Loopback Call=FALSE
RemoteIPAddress=10.64.86.91
RemoteUDPPort=16614
RemoteSignallingIPAddress=10.64.86.91
```

[^0-9a-fA-F*];remote=[0-9a-fA-F*]+S.
RemoteSignallingPort=5060
RemoteMediaIPAddress=10.127.17.142
RemoteMediaPort=16614
CoderTypeRate=g711ulaw

GlobalCallId=[0xC3DAB665 0x770C11E5 0x80318550 0x5A000ED7]
SessionIDLocaluuid=6497636d0b747785241cfbf5aa225064
SessionIDRemoteuuid=d82c680a3eaccd5c29ac6ceeeaa225061
RemoteIPAddress=10.64.86.91
RemoteUDPPort=21978
RemoteSignallingIPAddress=10.64.86.91
RemoteSignallingPort=5060
RemoteMediaIPAddress=10.127.17.188
RemoteMediaPort=21978

From the above output, you get to know that 1022 (highlighted) is the call identifier associated with the local session identifier d82c680a3eaccd5c29ac6ceeeaa225061. You can now use this call identifier to get further details and debugging of the desired call as shown below:

Device# show sip-ua calls callid 1022

SIP CALL INFO of CCAPI callid 1022
Call 1
SIP Call ID : 8cdac180-627159d8-9cd-5b56400a@10.64.86.91
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 4443332212
Called Number : 4443332211
Called URI : sip:4443332211@10.64.86.132:5060
Bit Flags : 0xC0401C 0x10000100 0x80004
CC Call ID : 1022
Source IP Address (Sig) : 10.64.86.132
Destn SIP Req Addr/Port : [10.64.86.91]:5060
Destn SIP Resp Addr/Port: [10.64.86.91]:5060
Destination Name : 10.64.86.91
Number of Media Streams : 1
Number of Active Streams: 1
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 1022
Stream Type : voice-only (0)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr/Port: [10.64.86.132]:16424
Media Dest IP Addr/Port : [10.127.17.142]:16614

Options-Ping ENABLED:NO ACTIVE:NO

SIP CALL INFO of peer leg CCAPI callid 1023

Cisco Unified Border Element Configuration Guide
Example for video Calls

The following session identifier is considered in the below example:

```
SessionIDLocaluid=6f0a93a3a79451aebeb6d83f79a3359f
SessionIDRemoteuid=a55b0f45861551b88f57d1fb5bb23f89
```

All the search patterns listed above for voice calls are also valid for video calls.

You can search for the session identifier using complete UUID (local, remote, or both) or use a wildcard pattern.

```
Device# show call active video session-id 6f*
```

Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

GENERIC:
Index=1
PeerAddress=sipp
PeerSubAddress=
PeerId=1
PeerIfIndex=14
LogicalIfIndex=0
CallDuration=00:00:56 sec
CallState=4
CallOrigin=2
ChargedUnits=0
InfoType=video
TransmitPackets=0
TransmitBytes=0
ReceivePackets=0
ReceiveBytes=0
VOIP:
ConnectionId[0x6083CB92 0x466511E5 0xFFFFFFFF8018F617 0xFFFFFFFFA7C45A02]
IncomingConnectionId[0x6083CB92 0x466511E5 0xFFFFFFFF8018F617 0xFFFFFFFFA7C45A02]
CallID=11
GlobalCallId=[0x6083F24F 0x466511E5 0xFFFFFFFF801BF617 0xFFFFFFFFA7C45A02]
CallReferenceId=0
CallServiceType=Unknown
RTP Loopback Call=FALSE
SessionIDLocaluuid=6f0a93a3a79451aebbb6d83f79a3359f
SessionIDRemoteuuid=a55bf0f45861551b88f57d1fb5bb23f89
RemoteIPAddress=10.64.86.70
RemoteSignallingIPAddress=10.64.86.70
RemoteSignallingIPPort=5061
RemoteMediaIPAddress=10.64.86.70
RemoteMediaIPPort=6003
RoundTripDelay=0 ms
tx_DtmfRelay=inband-voice
FastConnect=FALSE
PART XXIV

Security Compliance

- Common Criteria (CC) and The Federal Information Processing Standards (FIPS) Compliance, on page 895
Common Criteria (CC) and The Federal Information Processing Standards (FIPS) Compliance

Cisco Unified Border Element is Common Criteria (CC) and The Federal Information Processing Standards (FIPS) certified. The certification is applicable to Cisco Unified Border Element on Cisco CSR 1000v Series Cloud Services Router platform only.

Common Criteria (CC)

Common Criteria (CC) is a global security to which security products are evaluated. Common Criteria product certifications are mutually recognized by 28 nations, thus an evaluation that is conducted in one country is recognized by the other countries.

The Common Criteria for Information Technology Security Evaluation is an international standard (ISO/IEC 15408) that guarantees product security. The organizations (Government or Enterprise IT) specify functional and assurance requirements, the vendors claim and develop specific product qualities. The testing facilities examine products to determine whether they meet those vendor claims. Common Criteria guarantees that the process of specification, execution and assessment of a product has been conducted in a stringent and standardized manner.

The Federal Information Processing Standards (FIPS)

The Federal Information Processing Standards (FIPS) Publication 140-2, Security Requirements for Cryptographic Modules, details the U.S. government requirements for cryptographic modules. FIPS 140-2 specifies that a cryptographic module should be a set of hardware, software, firmware, or some combination that implements cryptographic functions or processes, including cryptographic algorithms and, optionally, key generation, and is contained within a defined cryptographic boundary.

FIPS specifies certain crypto algorithms as secure, and it also identifies which algorithms should be used if a cryptographic module is to be called FIPS compliant.

- Feature Information for Common Criteria (CC) and The Federal Information Standards (FIPS) Compliance, on page 896
- Supported Hardware and Software for Virtual CUBE, on page 896
- Common Criteria Configuration on Cisco CSR 1000v, on page 896
- FIPS Configuration on Cisco CSR 1000v, on page 908
Feature Information for Common Criteria (CC) and The Federal Information Standards (FIPS) Compliance

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>

Supported Hardware and Software for Virtual CUBE

For details on prerequisites for Virtual CUBE, see Supported Hardware and Software for Virtual CUBE.

Common Criteria Configuration on Cisco CSR 1000v

Enable Common Criteria Mode

Before you begin

- Delete existing certificates.
- Remove existing crypto keys.
- Remove existing TLS configuration (TLS version and Cipher Suites).

Step 1

```
enable
```

**Example:**

```
Router# enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2

```
configure terminal
```
Example:
Router# configure terminal
Enters global configuration mode.

Step 3  cc-mode
Example:
Router(config)# cc-mode
Enables common criteria configuration mode.

What to do next
Common Criteria (CC) mode enforces certain security checks for cryptographic protocols such as Transport Layer Security (TLS). CUBE uses TLS to secure signaling over SIP and HTTP client for XCC providers. Configure SIP TLS and HTTP TLS in the Common Criteria (CC) mode.

SIP TLS Configuration

SIP TLS Configuration Task Flow

Following are the steps to configure SIP TLS on your Cisco CSR 1000v router in Common Criteria mode.

1. Generate RSA Public Key, on page 897
2. Configure Certificate Authority Server, on page 898
3. Configure CSR Trustpoint, on page 899
4. Configure Peer Trustpoint, on page 900
5. Add Client Verification Trustpoint, on page 901
6. Enforce Strict SRTP, on page 902

Generate RSA Public Key

Step 1  enable
Example:
Router#enable
Enables privileged EXEC mode.
   • Enter your password if prompted.

Step 2  configure terminal
Example:
Router#configure terminal
Enters global configuration mode.
Step 3  
**crypto key generate rsa label** *key-label modulus modulus-size*

**Example:**
Router(config)# crypto key generate rsa general-keys label CUBE modulus 3072

Generates a public RSA key that is used with your CSR certificate.

- The *key-label* specifies the name that is used for an RSA key pair when they are exported.
- The *modulus-size* specifies the size of the key modulus. By default, the modulus of a Certification Authority (CA) key is 1024 bits. The size of the key modulus must be 2048 bits or higher, for it to be Common Criteria compliant.

Step 4  
**exit**

**Example:**
Router(config)# exit

Exits global configuration mode.

---

**Configure Certificate Authority Server**

Step 1  
**enable**

**Example:**
Router# enable

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2  
**configure terminal**

**Example:**
Router# configure terminal

Enters global configuration mode.

Step 3  
**crypto pki server** *cs-label*

**Example:**
Router(config)# crypto pki server CUBE

Defines a label for the Certificate Server and enters the certificate server configuration mode.

**Note**  
If you have generated the RSA key pair manually using the command **crypto key generate rsa label** *key-label modulus modulus-size*, the *cs-label* must match with the *key-label*, otherwise a certificate with the default key size of 1024 bits is generated.

Step 4  
**database level complete**

**Example:**
Router(cs-server)# database level complete

Writes each issued certificate to the certificate enrollment database.
Step 5  
**grant auto**  
*Example:*

Router(cs-server)# grant auto  
Automatically grants reenrollment requests for subordinate Certificate Authority (CA) server or Registration Authority (RA) mode Certificate Authority (CA).

Step 6  
**hash sha384**  
*Example:*

Router(cs-server)# hash sha384  
Sets the hash function SHA-384 for the signature that the Cisco IOS Certificate Authority (CA) uses to sign all the certificates that are issued by the server.

Step 7  
**no shut**  
*Example:*

Router(cs-server)# no shut  
% Some server settings cannot be changed after CA certificate generation.  
% Please enter a passphrase to protect the private key  
% or type Return to exit  
Password:  
Re-enter password:  
% Generating 3072 bit RSA keys, keys will be non-exportable...  
[OK] (elapsed time was 0 seconds)  
% Certificate Server enabled.  
Enables or reenables the certificate server. If the subordinate certificate server is enabled for the first time, the certificate server generates the key and receives its signing certificate from the root certificate server.  
After entering the passphrase (when prompted), the certificate server is enabled. This passphrase protects the private key.

Step 8  
**exit**  
*Example:*

Router(cs-server)# exit  
Exits certificate server configuration mode.

---

**Configure CSR Trustpoint**

Step 1  
**enable**  
*Example:*

Router#enable  
Enables privileged EXEC mode.  
  * Enter your password if prompted.
**Configure Peer Trustpoint**

**Step 1**  
**enable**  
**Example:**  
`Router#enable`  
Enables privileged EXEC mode.  
- Enter your password if prompted.

**Step 2**  
**configure terminal**  
**Example:**

```
Router#configure terminal
```
Enters global configuration mode.

**Configure Peer Trustpoint**

**Step 2**  
**configure terminal**  
**Example:**

```
Router#configure terminal
```
Enters global configuration mode.

**Step 3**  
**crypto pki trustpoint name**  
**Example:**  
`Router(config)#crypto pki trustpoint CUBE-TLS`
Declares the trustpoint with the name specified and enters trustpoint configuration mode. This trustpoint is used by your Router application for the TLS communication.

**Step 4**  
**hash sha384**  
**Example:**  
`Router(ca-trustpoint)#hash sha384`
Sets the hash function SHA-384 for the signature that the Cisco IOS Certificate Authority (CA) uses to sign all the certificates that are issued by the server.

A trustpoint with sample CSR certificate with subject-name "CN=Secure-Router" and "rsakeypair Router" is given below. The "rsakeypair label" must match with the label of the RSA keys that are generated in the earlier steps.

```
crypto pki trustpoint CUBE-TLS
  enrollment url http://X.X.X.X:80
  serial-number none
  fqdn none
  ip-address none
  subject-name CN=Secure-CUBE
  revocation-check none
  rsakeypair Router
```

**Step 5**  
**exit**  
**Example:**  
`Router(ca-trustpoint)# exit`
Exits trustpoint configuration mode.
Add Client Verification Trustpoint

Step 1 enable
Example:
Router#enable
Enables privileged EXEC mode.
  • Enter your password if prompted.

Step 2 configure terminal
Example:
Router#configure terminal
Enters global configuration mode.

Step 3 sip-ua
Example:
Router(config)#sip-ua
Enters SIP User Agent configuration mode to configure SIP-UA related commands.

**Step 4**

`crypto signaling remote-addr remote ip address remote ip mask trustpoint CUBEs trustpoint label client-vtp verification trustpoint`

**Example:**

Router(config-sip-ua)#crypto signaling remote-addr X.X.X.X 255.255.255.255 trustpoint CUBE-TLS client-vtp CUBE-VERIFY

Assigns a client verification trustpoint to SIP-UA. This client verification trustpoint is used to send Distinguished Name (DN) of the Certificate Authority (CA) server in the CUBE's client certificate request.

**Step 5**

`exit`

**Example:**

Router(config-sip-ua)#exit

Exits sip-ua configuration mode.

---

**Enforce Strict SRTP**

**Step 1**

`enable`

**Example:**

Router#enable

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2**

`configure terminal`

**Example:**

Router#configure terminal

Enters global configuration mode.

**Step 3**

`voice service voip`

**Example:**

Router(config)#voice service voip

Enters voice service configuration mode and specifies the encapsulation method as VoIP.

**Step 4**

`srtp`

**Example:**

Router(conf-voi-ser)#srtp

Enforces SRTP to secure the call flow through CUBE.

**Step 5**

`exit`

**Example:**

Router(conf-voi-ser)#exit
Exits voice service configuration mode.

HTTPS TLS Configuration

HTTPS TLS Configuration Task Flow

Following are the steps to configure HTTPS TLS on your Cisco CSR 1000v router in Common Criteria mode.

1. Prepare Cisco CSR 1000v Router's HTTP Server to Run in CC Mode, on page 903
2. Create Certificate Map for HTTPS Peer Trustpoint, on page 904
3. Configure HTTPS TLS Version, on page 905
4. Configure Supported Cipher Suites, on page 906
5. Apply Certificate Map to HTTPS Peer Trustpoint, on page 906

Prepare Cisco CSR 1000v Router's HTTP Server to Run in CC Mode

Step 1 enable
Example:
Router#enable
Enables privileged EXEC mode.
• Enter your password if prompted.

Step 2 configure terminal
Example:
Router#configure terminal
Enters global configuration mode.

Step 3 ip http server name
Example:
Router(config)#ip http server
Enables the HTTP server on the Cisco CSR 1000v router, allowing the use of Cisco web browser UI to monitor the router and issue commands to it.

Step 4 ip http authentication local
Example:
Router(config)#ip http authentication local
Specifies the authentication method for HTTP server users. The keyword local indicates that the username, password, and privilege level access combination that is specified in the local system configuration should be used for authentication and authorization.
Step 5  ip http secure-server  
Example:  
Router(config)#ip http secure-server  
Enables a secure HTTP server on the Cisco CSR 1000v router.

Step 6  ip http secure-trustpoint trustpoint-name  
Example:  
Router(config)#ip http secure-trustpoint CUBE-TLS  
Specifies the trustpoint that is used for obtaining signed certificates for a secure HTTP server on the Cisco CSR 1000v router.

Step 7  ip http secure-client-auth  
Example:  
Router(config)#ip http secure-client-auth  
Configures the HTTP server to request an X.509v3 certificate from the client to authenticate the client during the connection process.

Step 8  ip http secure-peer-verify-trustpoint client's issuer  
Example:  
Router(config)#ip http secure-peer-verify-trustpoint secure-clientissuer  
Configures the client verification trustpoint for the HTTP server on the Cisco CSR 1000v router. This peer verification trustpoint is used to send Distinguished Name (DN) of Certificate Authority (CA) in the client certificate request during the TLS handshake of HTTP.

Step 9  exit  
Example:  
Router(config)#exit  
Exits the global configuration mode.

Create Certificate Map for HTTPS Peer Trustpoint

Step 1  enable  
Example:  
Router#enable  
Enables privileged EXEC mode.

Step 2  configure terminal  
Example:  
Router#configure terminal
Enters global configuration mode.

**Step 3**  
**crypto pki certificate map label sequence-number**

**Example:**

```
Router(config)#crypto pki certificate map cubemap 10
```

Creates a certificate map that defines certificate-based Access Control Lists (ACLs) and enters the certificate map configuration mode. The `sequence-number` orders the ACLs with the same label. ACLs with the same label are processed from the lowest to the highest sequence number. When an ACL is matched, the processing stops with a successful result.

**Step 4**  
**alt-subject-name eq match-value**

**Example:**

```
Router(ca-certificate-map)#alt-subject-name peername
```

Specifies the certificate fields with their matching criteria in the certificate map configuration mode. The alternate subject name that is specified in the map must be present in SAN extension of the peer id certificate.

**Step 5**  
**exit**

**Example:**

```
Router(ca-certificate-map)#exit
```

Exits certificate map configuration mode.

---

**Configure HTTPS TLS Version**

**Step 1**  
**enable**

**Example:**

```
Router#enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2**  
**configure terminal**

**Example:**

```
Router#configure terminal
```

Enters global configuration mode.

**Step 3**  
**ip http tls-version version**

**Example:**

```
Router(config)#ip http tls-version TLSv1.2
```

Configures the specified TLS version for HTTPS. Configure TLSv1.1 or TLSv1.2 to be Common Criteria compliant.

**Step 4**  
**exit**

**Example:**

```
Router(config)#exit
```
Configure Supported Cipher Suites

Step 1 enable
Example:
Router#enable
Enables privileged EXEC mode.
• Enter your password if prompted.

Step 2 configure terminal
Example:
Router#configure terminal
Enters global configuration mode.

Step 3 ip http secure-ciphersuite supported cipher suites
Example:
Router(config)#ip http secure-ciphersuite aes-128-cbc-sha aes-256-cbc-sha dhe-aes-128-cbc-sha
ecdhe-rsa-aes-cbc-sha2 ecdhe-ecdsa-aes-gcm-sha2
Specifies the cipher suites that are used for encryption over the secure HTTP connection between the client and the HTTP server. Common Criteria supports the cipher suites that are given in the preceding example. Configure all the cipher suites if you are not aware of the client cipher support.

Step 4 exit
Example:
Router(config)#exit
Exits global configuration mode.

Apply Certificate Map to HTTPS Peer Trustpoint

Step 1 enable
Example:
Router#enable
Enables privileged EXEC mode.
• Enter your password if prompted.

Step 2 configure terminal

Cisco Unified Border Element Configuration Guide
Example:
Router#configure terminal
Enters global configuration mode.

Step 3  crypto pki trustpoint name
Example:
Router(config)#crypto pki trustpoint CUBE-HTTPS
Declares the HTTPS peer trustpoint for the Cisco CSR 1000v router.

Step 4  match certificate map name
Example:
Router(ca-trustpoint)#match certificate cubemap
Associates the certificate map that is defined by using the crypto pki certificate map command with the HTTPS trustpoint. The map name argument in the match certificate command must match the label argument that is specified in the previously defined crypto pki certificate map command.

Step 5  match eku attribute
Example:
Router(ca-trustpoint)#match eku client-auth server-auth
Allows the HTTPS peer which acts as a client and a server to validate a peer certificate only if the specified Extended Key Usage (EKU) attribute is present in the certificate. If the Cisco CSR 1000v router is a client, then you must configure server-auth. If Cisco CSR 1000v router is a server, then you must configure client-auth.

Step 6  exit
Example:
Router(ca-trustpoint)#exit
Exits trustpoint configuration mode.

NTP Configuration Restrictions in Common Criteria Mode

In Common Criteria mode, the following restrictions are applicable to NTP configuration.

- Do not configure NTP version 1 and 2. Following are the NTP version commands.
  - ntp server ip-address prefer source interface version version
  - ntp peer ip-address version version

- Do not configure NTP broadcast. Following are the NTP broadcast commands.
  - ntp broadcast delay delay-timer
  - ntp broadcast client
  - ntp broadcast destination ip-address
  - ntp broadcast destination ip-address key key
FIPS Configuration on Cisco CSR 1000v

Configuration Requirements for FIPS Compliance

There is no specific command to enable FIPS mode. For the Virtual CUBE on the Cisco CSR 1000v router to be FIPS-compliant, the following commands must be configured.

- **crypto key generate rsa modulus modulus-size**
  
The *modulus-size* varies from 360 bits to 4096 bits. The size of the RSA key must be 2048 bit or higher for FIPS compliance.

- **The Hash Algorithms that are configured using the command hash sha384 under the configured trustpoint and the crypto pki server on the CSR must use sha384 or greater, namely sha512.**
PART XXV

Appendixes

• Additional References, on page 911
• Glossary, on page 915
Additional References

The following sections provide references related to the CUBE Configuration Guide.

- Related References, on page 911
- Standards, on page 912
- MIBs, on page 912
- RFCs, on page 912
- Technical Assistance, on page 914

### Related References

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Navigator</td>
<td>For information about platforms supported, and Cisco IOS software image support., search by Feature Name listed in Feature Information Table in <a href="http://www.cisco.com/go/cfn">www.cisco.com/go/cfn</a></td>
</tr>
<tr>
<td>Bug Search Tool Kit</td>
<td>For information about latest caveats and feature information, see Bug Search Tool</td>
</tr>
<tr>
<td>Cisco IOS commands</td>
<td>Cisco IOS Master Commands List, All Releases</td>
</tr>
<tr>
<td>Cisco IOS Voice commands</td>
<td>Cisco IOS Voice Command Reference</td>
</tr>
<tr>
<td>Related Application Guides</td>
<td>• Cisco Unified Communications Manager and Cisco IOS Interoperability Guide</td>
</tr>
<tr>
<td></td>
<td>• Cisco IOS SIP Configuration Guide</td>
</tr>
<tr>
<td></td>
<td>• Cisco Unified Communications Manager (CallManager) Programming Guides</td>
</tr>
</tbody>
</table>
## Standards

<table>
<thead>
<tr>
<th>Standard</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITU-T G.711</td>
<td>—</td>
</tr>
</tbody>
</table>

## MIBs

<table>
<thead>
<tr>
<th>MIB</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>• CISCO-PROCESS MIB</td>
<td></td>
</tr>
<tr>
<td>• CISCO-MEMORY-POOL-MIB</td>
<td></td>
</tr>
<tr>
<td>• CISCO-SIP-UA-MIB</td>
<td></td>
</tr>
<tr>
<td>• DIAL-CONTROL-MIB</td>
<td></td>
</tr>
<tr>
<td>• CISCO-VOICE-DIAL-CONTROL-MIB</td>
<td></td>
</tr>
<tr>
<td>• CISCO-DSP-MGMT-MIB</td>
<td></td>
</tr>
<tr>
<td>• IF-MIB</td>
<td></td>
</tr>
<tr>
<td>• IP-TAP-MIB</td>
<td></td>
</tr>
<tr>
<td>• TAP2-MIB</td>
<td></td>
</tr>
<tr>
<td>• USER-CONNECTION-TAP-MIB</td>
<td></td>
</tr>
</tbody>
</table>

To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL:

http://www.cisco.com/go/mibs

## RFCs

<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 1889</td>
<td><em>RTP: A Transport Protocol for Real-Time Applications</em></td>
</tr>
<tr>
<td>RFC 2131</td>
<td><em>Dynamic Host Configuration Protocol</em></td>
</tr>
<tr>
<td>RFC</td>
<td>Title</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>RFC 2132</td>
<td>DHCP Options and BOOTP Vendor Extensions</td>
</tr>
<tr>
<td>RFC 2198</td>
<td>RTP Payload for Redundant Audio Data</td>
</tr>
<tr>
<td>RFC 2327</td>
<td>SDP: Session Description Protocol</td>
</tr>
<tr>
<td>RFC 2543</td>
<td>SIP: Session Initiation Protocol</td>
</tr>
<tr>
<td>RFC 2543-bis-04</td>
<td>SIP: Session Initiation Protocol, draft-ietf-sip-rfc2543bis-04.txt</td>
</tr>
<tr>
<td>RFC 2782</td>
<td>A DNS RR for Specifying the Location of Services (DNS SRV)</td>
</tr>
<tr>
<td>RFC 2806</td>
<td>URLs for Telephone Calls</td>
</tr>
<tr>
<td>RFC 2833</td>
<td>RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals</td>
</tr>
<tr>
<td>RFC 3203</td>
<td>DHCP reconfigure extension</td>
</tr>
<tr>
<td>RFC 3261</td>
<td>SIP: Session Initiation Protocol</td>
</tr>
<tr>
<td>RFC 3262</td>
<td>Reliability of Provisional Responses in Session Initiation Protocol (SIP)</td>
</tr>
<tr>
<td>RFC 3323</td>
<td>A Privacy Mechanism for the Session Initiation Protocol (SIP)</td>
</tr>
<tr>
<td>RFC 3325</td>
<td>Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks</td>
</tr>
<tr>
<td>RFC 3515</td>
<td>The Session Initiation Protocol (SIP) Refer Method</td>
</tr>
<tr>
<td>RFC 3361</td>
<td>Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers</td>
</tr>
<tr>
<td>RFC 3455</td>
<td>Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)</td>
</tr>
<tr>
<td>RFC 3608</td>
<td>Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration</td>
</tr>
<tr>
<td>RFC 3711</td>
<td>The Secure Real-time Transport Protocol (SRTP)</td>
</tr>
<tr>
<td>RFC 3925</td>
<td>Vendor-Identifying Vendor Options for Dynamic Host Configuration Protocol version 4 (DHCPv4)</td>
</tr>
</tbody>
</table>
Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support website provides extensive online resources, including</td>
<td><a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a></td>
</tr>
<tr>
<td>documentation and tools for troubleshooting and resolving technical issues</td>
<td></td>
</tr>
<tr>
<td>with Cisco products and technologies.</td>
<td></td>
</tr>
<tr>
<td>To receive security and technical information about your products, you can</td>
<td></td>
</tr>
<tr>
<td>subscribe to various services, such as the Product Alert Tool (accessed from</td>
<td></td>
</tr>
<tr>
<td>Field Notices), the Cisco Technical Services Newsletter, and Really Simple</td>
<td></td>
</tr>
<tr>
<td>Syndication (RSS) Feeds.</td>
<td></td>
</tr>
<tr>
<td>Access to most tools on the Cisco Support website requires a Cisco.com user</td>
<td></td>
</tr>
<tr>
<td>ID and password.</td>
<td></td>
</tr>
</tbody>
</table>
AMR-NB — Adaptive Multi Rate codec - Narrow Band.
Allow header — Lists the set of methods supported by the UA generating the message.
bind — In SIP, configuring the source address for signaling and media packets to the IP address of a specific interface.
call — In SIP, a call consists of all participants in a conference invited by a common source. A SIP call is identified by a globally unique call identifier. A point-to-point IP telephony conversation maps into a single SIP call.
call leg — A logical connection between the router and another endpoint.
CLI — command-line interface.
Content-Type header — Specifies the media type of the message body.
CSeq header — Serves as a way to identify and order transactions. It consists of a sequence number and a method. It uniquely identifies transactions and differentiates between new requests and request retransmissions.
delta — An incremental value. In this case, the delta is the difference between the current time and the time when the response occurred.
dial peer — An addressable call endpoint.
DNS — Domain Name System. Used to translate H.323 IDs, URLs, or e-mail IDs to IP addresses. DNS is also used to assist in locating remote gatekeepers and to reverse-map raw IP addresses to host names of administrative domains.
DNS SRV — Domain Name System Server. Used to locate servers for a given service.
DSP — Digital Signal Processor.
DTMF — dual-tone multifrequency. Use of two simultaneous voice-band tones for dialing (such as touch-tone).
EFXS — IP phone virtual voice ports.
FQDN — fully qualified domain name. Complete domain name including the host portion; for example, serverA.companyA.com .
FXS—analog telephone voice ports.

gateway—A gateway allows SIP or H.323 terminals to communicate with terminals configured to other protocols by converting protocols. A gateway is the point where a circuit-switched call is encoded and repackaged into IP packets.

H.323—An International Telecommunication Union (ITU-T) standard that describes packet-based video, audio, and data conferencing. H.323 is an umbrella standard that describes the architecture of the conferencing system and refers to a set of other standards (H.245, H.225.0, and Q.931) to describe its actual protocol.

iLBC—internet Low Bitrate Codec.

INVITE—A SIP message that initiates a SIP session. It indicates that a user is invited to participate, provides a session description, indicates the type of media, and provides insight regarding the capabilities of the called and calling parties.

IP—Internet Protocol. A connectionless protocol that operates at the network layer (Layer 3) of the OSI model. IP provides features for addressing, type-of-service specification, fragmentation and reassemble, and security. Defined in RFC 791. This protocol works with TCP and is usually identified as TCP/IP. See TCP/IP.

ISDN—Integrated Services Digital Network.

Minimum Timer—Configured minimum value for session interval accepted by SIP elements (proxy, UAC, UAS). This value helps minimize the processing load from numerous INVITE requests.

Min-SE—Minimum Session Expiration. The minimum value for session expiration.

multicast—A process of transmitting PDUs from one source to many destinations. The actual mechanism (that is, IP multicast, multi-unicast, and so forth) for this process might be different for LAN technologies.

originator—User agent that initiates the transfer or Refer request with the recipient.

PDU—protocol data units. Used by bridges to transfer connectivity information.

PER—Packed Encoding Rule.

proxy—A SIP UAC or UAS that forwards requests and responses on behalf of another SIP UAC or UAS.

proxy server—An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy interprets and, if necessary, rewrites a request message before forwarding it.

recipient—User agent that receives the Refer request from the originator and is transferred to the final recipient.

redirect server—A server that accepts a SIP request, maps the address into zero or more new addresses, and returns these addresses to the client. It does not initiate its own SIP request or accept calls.

re-INVITE—An INVITE request sent during an active call leg.

Request URI—Request Uniform Resource Identifier. It can be a SIP or general URL and indicates the user or service to which the request is being addressed.

RFC—Request For Comments.

RTP—Real-Time Transport Protocol (RFC 1889)

SCCP—Skinny Client Control Protocol.

SDP—Session Description Protocol. Messages containing capabilities information that are exchanged between gateways.
**session** — A SIP session is a set of multimedia senders and receivers and the data streams flowing between the senders and receivers. A SIP multimedia conference is an example of a session. The called party can be invited several times by different calls to the same session.

**session expiration** — The time at which an element considers the call timed out if no successful INVITE transaction occurs first.

**session interval** — The largest amount of time that can occur between INVITE requests in a call before a call is timed out. The session interval is conveyed in the Session-Expires header. The UAS obtains this value from the Session-Expires header of a 2xx INVITE response that it sends. Proxies and UACs determine this value from the Session-Expires header in a 2xx INVITE response they receive.

**SIP** — Session Initiation Protocol. An application-layer protocol originally developed by the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force (IETF). Their goal was to equip platforms to signal the setup of voice and multimedia calls over IP networks. SIP features are compliant with IETF RFC 2543, published in March 1999.

**SIP URL** — Session Initiation Protocol Uniform Resource Locator. Used in SIP messages to indicate the originator, recipient, and destination of the SIP request. Takes the basic form of user@host, where user is a name or telephone number, and host is a domain name or network address.

**SPI** — service provider interface.

**socket listener** — Software provided by a socket client to receives datagrams addressed to the socket.

**stateful proxy** — A proxy in keepalive mode that remembers incoming and outgoing requests.

**TCP** — Transmission Control Protocol. Connection-oriented transport layer protocol that provides reliable full-duplex data transmissions. TCP is part of the TCP/IP protocol stack. See also TCP/IP and IP.

**TDM** — time-division multiplexing.

**UA** — user agent. A combination of UAS and UAC that initiates and receives calls. See UAS and UAC.

**UAC** — user agent client. A client application that initiates a SIP request.

**UAS** — user agent server. A server application that contacts the user when a SIP request is received and then returns a response on behalf of the user. The response accepts, rejects, or redirects the request.

**UDP** — User Datagram Protocol. Connectionless transport layer protocol in the TCP/IP protocol stack. UDP is a simple protocol that exchanges datagrams without acknowledgments or guaranteed delivery, requiring that error processing and retransmission be handled by other protocols. UDP is defined in RFC-768.

**URI** — Uniform Resource Identifier. Takes a form similar to an e-mail address. It indicates the user’s SIP identity and is used for redirection of SIP messages.

**URL** — Universal Resource Locator. Standard address of any resource on the Internet that is part of the World Wide Web (WWW).

**User Agent** — A combination of UAS and UAC that initiates and receives calls. See UAS and UAC.

**VFC** — Voice Feature Card.

**VoIP** — Voice over IP. The ability to carry normal telephone-style voice over an IP-based Internet with POTS-like functionality, reliability, and voice quality. VoIP is a blanket term that generally refers to the Cisco standards-based approach (for example, H.323) to IP voice traffic.