CONTENTS

CHAPTER 1

Read Me First 1
Short Description 2

CHAPTER 2

New and Changed Information 3

CHAPTER 3

Supported Platforms 5
Feature Comparison on Supported Platforms 7

PART I

Overview of Cisco Unified Border Element 11

CHAPTER 4

Information About Cisco Unified Border Element 13
SIP/H.323 Trunking 16
Typical Deployment Scenarios for CUBE 17
How to Configure Basic CUBE Features 18
Enabling the CUBE Application on a Device 19
Verifying the CUBE Application on the Device 21
Configuring a Trusted IP Address List for Toll-Fraud Prevention 22

CHAPTER 5

Virtual CUBE 25
Feature Information for Virtual CUBE 25
Prerequisites for Virtual CUBE 26
Hardware 26
Software 26
Features Supported with Virtual CUBE 27
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restrictions</td>
<td>27</td>
</tr>
<tr>
<td>Information about Virtual CUBE</td>
<td>27</td>
</tr>
<tr>
<td>Media</td>
<td>27</td>
</tr>
<tr>
<td>Virtual CUBE Licensing Requirements</td>
<td>28</td>
</tr>
<tr>
<td>Virtual CUBE with CSR1000V</td>
<td>28</td>
</tr>
<tr>
<td>Virtual CUBE with Catalyst 8000V</td>
<td>28</td>
</tr>
<tr>
<td>Install Virtual CUBE on ESXi</td>
<td>28</td>
</tr>
<tr>
<td>How to Enable Virtual CUBE</td>
<td>29</td>
</tr>
<tr>
<td>Troubleshooting Virtual CUBE</td>
<td>29</td>
</tr>
<tr>
<td><strong>CHAPTER 6</strong></td>
<td><strong>Dial-Peer Matching</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Dial Peers in CUBE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Configuring Inbound and Outbound Dial-Peer Matching for CUBE</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Preference for Dial-Peer Matching</strong></td>
</tr>
<tr>
<td><strong>CHAPTER 7</strong></td>
<td><strong>DTMF Relay</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Feature Information for DTMF Relay</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Information About DTMF Relay</strong></td>
</tr>
<tr>
<td></td>
<td><strong>DTMF Tones</strong></td>
</tr>
<tr>
<td></td>
<td><strong>DTMF Relay</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Configuring DTMF Relays</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Interoperability and Priority with Multiple DTMF Relay Methods</strong></td>
</tr>
<tr>
<td></td>
<td><strong>DTMF Interoperability Table</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Verifying DTMF Relay</strong></td>
</tr>
<tr>
<td><strong>CHAPTER 8</strong></td>
<td><strong>Introduction to Codecs</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Why CUBE Needs Codecs</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Restrictions for Voice-Class Codec Transparent</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Voice Media Transmission</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Voice Activity Detection</strong></td>
</tr>
<tr>
<td></td>
<td><strong>VoIP Bandwidth Requirements</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Supported Audio and Video Codecs</strong></td>
</tr>
<tr>
<td></td>
<td><strong>How to Configure Codecs</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Configuring Audio and Video Codecs at the Dial Peer Level</strong></td>
</tr>
</tbody>
</table>
Feature Information for Media Path 127
Media Flow-Through 128
   Restrictions for Media Flow-Through 128
   Configure Media Flow-Through 129
Media Flow-Around 130
   Configure Media Flow-Around 130
Media Anti-Trombone 131
   Prerequisites 132
   Restrictions for Media Anti-Tromboning 132
   Configuring Media Anti-Tromboning 132

CHAPTER 13 SIP Profiles 135
   Feature Information for SIP Profiles 135
   Information About SIP Profiles 136
      Important Characteristics of SIP Profiles 137
   Restrictions for SIP Profiles 139
   How to Configure SIP Profiles 139
      Configuring a SIP Profile to Manipulate SIP Request or Response Headers 140
      Configuring SIP Profiles for Copying Unsupported SDP Headers 141
         Example: Configuring SIP Profile Rules (Attribute Passing) 143
         Example: Configuring SIP Profile Rules (Parameter Passing) 143
         Example: Configuration to Remove an Attribute 143
      Configuring SIP Profile Using Rule Tag 143
      Configuring a SIP Profile for Non-standard SIP Header 145
      Upgrading or Downgrading SIP Profile Configurations 147
      Configuring a SIP Profile as an Outbound Profile 148
      Configuring a SIP Profile as an Inbound Profile 149
   Verifying SIP Profiles 150
   Troubleshooting SIP Profiles 151
      Examples: Adding, Modifying, Removing SIP Profiles 152
         Example: Adding a SIP, SDP, or Peer Header 152
         Example: Modifying a SIP, SDP, or Peer Header 153
         Example: Remove a SIP, SDP, or Peer Header 156
         Example: Inserting SIP Profile Rules 157
## Chapter 19
### Matching Inbound Dial Peers by URI
- Configuring an Inbound Dial Peer to Match on URI
- Examples for Configuring an Inbound Dial Peer to Match on a URI

## Chapter 20
### URI-Based Dialing Enhancements
- Feature Information for URI-Based Dialing Enhancements
- Information About URI-Based Dialing Enhancements
- Call Flows for URI-Based Dialing Enhancements
- How to Configure URI-Based Dialing Enhancements
  - Configuring Pass Through of SIP URI Headers
  - Configuring Pass Though of Request URI and To Header URI (Global Level)
  - Configuring Pass Though of Request URI and To Header URI (Dial Peer Level)
  - Configuring Pass Through of 302 Contact Header
  - Configuring Pass Through of 302 Contact Header (Global Level)
  - Configuring Pass Through of 302 Contact Header (Dial Peer Level)
- Deriving of Session Target from URI
- 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)
  - Example: Configuring Pass Though of Request URI and To Header URI (Dial Peer Level)
  - Example: Configuring Pass Through of 302 Contact Header
  - Example: Configuring Pass Through of 302 Contact Header (Global Level)
  - Example: Configuring Pass Through of 302 Contact Header (Dial Peer Level)
  - Example: Deriving Session Target from URI
- Additional References for URI-Based Dialing Enhancements

## Chapter 21
### Multiple Pattern Support on a Voice Dial Peer
- Feature Information for Multiple Pattern Support on a Voice Dial Peer
- Restrictions for Multiple Pattern Support on a Voice Dial Peer
- Information About Multiple Pattern Support on a Voice Dial Peer
- Configuring Multiple Pattern Support on a Voice Dial Peer
- Verifying Multiple Pattern Support on a Voice Dial Peer
- Configuration Examples for Multiple Pattern Support on a Voice Dial Peer
CHAPTER 22
Outbound Dial-Peer Group as an Inbound Dial-Peer Destination 293
Feature Information for Outbound Dial-Peer Group as an Inbound Dial-Peer Destination 293
Restrictions 294
Information About Outbound Dial-Peer Group as an Inbound Dial-Peer Destination 294
Configuring Outbound Dial-Peer Group as an Inbound Dial-Peer Destination 295
Verifying Outbound Dial-Peer Groups as an Inbound Dial-Peer Destination 297
Troubleshooting Tips 298
Configuration Examples for Outbound Dial Peer Group as an Inbound Dial-Peer Destination 299

CHAPTER 23
Inbound Leg Headers for Outbound Dial-Peer Matching 303
Feature Information for Inbound Leg Headers for Outbound Dial-Peer Matching 303
Prerequisites for Inbound Leg Headers for Outbound Dial-Peer Matching 304
Restrictions for Inbound Leg Headers for Outbound Dial-Peer Matching 304
Information About Inbound Leg Headers for Outbound Dial-Peer Matching 305
Configuring Inbound Leg Headers for Outbound Dial-Peer Matching 305
Verifying Inbound Leg Headers for Outbound Dial-Peer Matching 308
Configuration Example: Inbound Leg Headers for Outbound Dial-Peer Matching 310

CHAPTER 24
Server Groups in Outbound Dial Peers 313
Feature Information for Configuring Server Groups in Outbound Dial Peers 313
Information About Server Groups in Outbound Dial Peers 314
How to Configure Server Groups in Outbound Dial Peers 315
Configuring Server Groups in Outbound Dial Peers 315
Verifying Server Groups in Outbound Dial Peers 318
Configuration Examples for Server Groups in Outbound Dial Peers 319

CHAPTER 25
Domain-Based Routing Support on the Cisco UBE 323
Feature Information for Domain-Based Routing Support on the Cisco UBE 323
Restrictions for Domain-Based Routing Support on the Cisco UBE 324
Information About Domain-Based Routing Support on the Cisco UBE 324
How to Configure Domain-Based Routing Support on the Cisco UBE 325
Configuring Domain-Based Routing at Global Level 325
Configuring Domain-Based Routing at Dial Peer Level 326
Verifying and Troubleshooting Domain-Based Routing Support on the Cisco UBE 327
Configuration Examples for Domain-Based Routing Support on the Cisco UBE 330
Example Configuring Domain-Based Routing Support on the Cisco UBE 330

CHAPTER 26

ENUM Enhancement per Kaplan Draft RFC 331
Feature Information for ENUM Enhancement per Kaplan Draft RFC 331
Restrictions for ENUM Enhancement per Kaplan Draft RFC 332
Information About ENUM Enhancement per Kaplan Draft RFC 333
How to Configure ENUM Enhancement per Kaplan Draft RFC 333
  Enabling Source-Based Routing 333
  Testing the ENUM Request 334
  Verifying the ENUM Request 334
Troubleshooting Tips 336
Configuration Examples for ENUM Enhancement per Kaplan Draft RFC 336

PART III

Multi-Tenancy 339

CHAPTER 27

Support for Multi-VRF 341
Feature Information for VRF 341
Information About Voice-VRF 343
Information About Multi-VRF 343
VRF Preference Order 344
Restrictions 344
Recommendations 345
Configuring VRF 345
  Create a VRF 346
  Assign Interface to VRF 347
  Create Dial-peers 348
  Bind Dial-peers 349
Configure VRF-Specific RTP Port Ranges 351
  Example: VRF with overlapping and non-overlapping RTP Port Range 353
Directory Number (DN) Overlap across Multiple-VRFs 354
  Example: Associating Dial-peer Groups to Overcome DN Overlap 355
IP Overlap with VRF 356
Codecs Configured Using Preference Lists 412
Prerequisites for Codec Preference Lists 412
Restrictions for Codecs Preference Lists 413
How to Configure Codec Preference Lists 413

Configuring Audio Codecs Using a Codec Voice Class and Preference Lists 413
Disabling Codec Filtering 415

Troubleshooting Negotiation of an Audio Codec from a List of Codecs 416
Verifying Negotiation of an Audio Codec from a List of Codecs 417

PART V  DSP Services 421

CHAPTER 31  Transcoding 423
Configure LTI-Based Transcoding 424
Configuration Examples for LTI Based Transcoding 426
Configuring SCCP-based Transcoding (ISR-G2 devices only) 428
TLS for SCCP Connection for DSP Services 431
Configuring Secure Transcoding 431
Configuring the Certificate Authority 431
Configuring a Trustpoint for the Secure Universal Transcoder 432
Configuring DSPFARM Services 434
Associating SCCP to the Secure DSPFARM Profile 434
Registering the Secure Universal Transcoder to the CUBE 437
Configuration Examples for SCCP Based Transcoding 439

CHAPTER 32  Transrating 441
Configuring Transrating for a Codec 441

CHAPTER 33  Call Progress Analysis Over IP-to-IP Media Session 443
Feature Information for Call Progress Analysis Over IP-IP Media Session 443
Restrictions for Call Progress Analysis Over IP-to-IP Media Session 444
Information About Call Progress Analysis Over IP-IP Media Session 445
Call Progress Analysis 445
CPA Events 445
How to Configure Call Progress Analysis Over IP-to-IP Media Session 446
### Contents

- Enabling CPA and Setting the CPA Parameters 446
- Verifying the Call Progress Analysis Over IP-to-IP Media Session 448
- Troubleshooting Tips 449
- Configuration Examples for the Call Progress Analysis Over IP-to-IP Media Session 449

#### CHAPTER 34

**Voice Packetization** 451

- Configuring Transrating for a Codec 451

#### CHAPTER 35

**Fax Detection for SIP Call and Transfer** 453

- Restrictions for Fax Detection for SIP Call and Transfer On Cisco IOS XE 453
- Information About Fax Detection for SIP Call and Transfer 453
- Local Redirect Mode 454
- Refer Redirect Mode 455
- Fax Detection with Cisco IOS XE High Availability 456
- How to Configure Fax Detection for SIP Calls 456
- Configure DSP Resource to Detect Fax Tone 456
- Dial-peer Configuration to Redirect Fax Call 457
- Verifying Fax Detection for SIP Calls 459
- Troubleshooting Fax Detection for SIP Calls 460
- Configuration Examples for Fax Detection for SIP Calls 460
- Example: Configuring Local Redirect 460
- Example: Configuring Refer Redirect 461
- Feature Information for Fax Detection for SIP Call and Transfer 461

#### PART VI

**Video** 463

#### CHAPTER 36

**Video Suppression** 465

- Feature Information for Video Suppression 465
- Restrictions 465
- Information About Video Suppression 466
- Feature Behavior 466
- Configuring Video Suppression 466
- Troubleshooting Tips 467
### PART VII  Media Services  469

#### CHAPTER 37  Configuring RTCP Report Generation  471

- **Prerequisites**  471
- **Restrictions**  471
- Configuring RTCP Report Generation on Cisco UBE  472
- **Troubleshooting Tips**  473
- Feature Information for Configuring RTCP Report Generation  474

### PART VIII  Media Recording  477

#### CHAPTER 38  Network-Based Recording  479

- Feature Information for Network-Based Recording  479
- Restrictions for Network-Based Recording  480
- Information About Network-Based Recording Using CUBE  481
  - Deployment Scenarios for CUBE-based Recording  481
  - Open Recording Architecture  482
    - Network Layer  483
    - Capture and Media Processing Layer  483
    - Application Layer  483
  - MediaForking Topologies  484
    - MediaForking with Cisco UCM  484
    - MediaForking without Cisco UCM  484
  - SIP Recorder Interface  484
    - Metadata  484
  - How to Configure Network-Based Recording  485
    - Configuring Network-Based Recording (with Media Profile Recorder)  485
    - Configuring Network-Based Recording (without Media Profile Recorder)  488
  - Verifying the Network-Based Recording Using CUBE  490
- Additional References for Network-Based Recording  505

#### CHAPTER 39  SIPREC (SIP Recording)  507

- Feature Information for SIPREC-based Recording  507
Secure Recording of Secure Calls and Nonsecure Calls 598
Support for High Availability 599
Media Latch 599

PART X

SIP Header Manipulation 601

CHAPTER 44

Passing Headers Unsupported by CUBE 603
Feature Information for Copying with SIP Profiles 603
Example: Passing a Header Not Supported by CUBE 603

CHAPTER 45

Copying SIP Headers 605
Feature Information for Copying with SIP Profiles 605
How to Copy SIP Header Fields to Another 606
Copying From an Incoming Header and Modifying an Outgoing Header 606
Copying From One Outgoing Header to Another 608
Example: Copying the To Header into the SIP-Req-URI 609

CHAPTER 46

Manipulate SIP Status-Line Header of SIP Responses 611
Feature Information for Manipulating SIP Responses 611
Copying Incoming SIP Response Status Line to Outgoing SIP Response 612
Modifying Status-Line Header of Outgoing SIP Response with User Defined Values 615

PART XI

Payload Type Interoperability 617

CHAPTER 47

Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls 619
Feature Information for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls 619
Restrictions for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls 620
Symmetric and Asymmetric Calls 620
High Availability Checkpointing Support for Asymmetric Payload 621
How to Configure Dynamic Payload Type Passthrough for DTMF and Codec Packets for SIP-to-SIP Calls 622
Configuring Dynamic Payload Type Passthrough at the Global Level 622
### Protocol Interworking

#### PART XII

#### CHAPTER 48

**Delayed-Offer to Early-Offer**

- Feature Information for Delayed-Offer to Early-Offer 629
- Prerequisites for Delayed-Offer to Early-Offer 630
- Restrictions for Delayed-Offer to Early-Offer Media Flow-Around 630
- Delayed-Offer to Early-Offer in Media Flow-Around Calls 630
  - Configuring Delayed Offer to Early Offer 631
  - Configuring Delayed Offer to Early Offer for Video Calls 632
  - Configuring Delayed Offer to Early Offer Medial Flow-Around 633
- MidCall Renegotiation Support for Delayed-Offer to Early-Offer Calls 634
  - Restrictions for MidCall Renegotiation Support for DO-EO Calls 635
  - Configuring Mid Call Renegotiation Support for Delayed-Offer to Early-Offer Calls 635
- High-Density Transcoding Calls in Delayed-Offer to Early-Offer 636
  - Restrictions for High-Density Transcoding DO-EO Calls 637
  - Configuring High-Density Transcoding 637

#### CHAPTER 49

**H.323-to-SIP Interworking on CUBE**

- Prerequisites 639
- Restrictions 639
- H.323-to-SIP Basic Call Interworking 640
- H.323-to-SIP Supplementary Features Interworking 642
- H.323-to-SIP Codec Progress Indicator Interworking for Media Cut-Through 643
- Configuring H.323-to-SIP Interworking 643

#### CHAPTER 50

**H.323-to-H.323 Interworking on CUBE**

- Feature Information for H.323-to-H.323 Interworking 645
CHAPTER 53

SRTP-RTP Interworking 683

Feature Information for SRTP-RTP Interworking 683
Prerequisites for SRTP-RTP Interworking 684
Restrictions for SRTP-RTP Interworking 684
Information About SRTP-RTP Interworking 684
Support for SRTP-RTP Interworking 684

Using SRTP-RTP Chain for Interworking Between AES_CM_128_HMAC_SHA1_32 and AES_CM_128_HMAC_SHA1_80 Crypto Suites 686

Supplementary Services Support 687

How to Configure Support for SRTP-RTP Interworking 688
Configuring SRTP-RTP Interworking Support 688
Configuring Crypto Authentication 690
Enabling SRTP Fallback 692
Troubleshooting Tips 694

Verifying SRTP-RTP Supplementary Services Support 694

Configuration Examples for SRTP-RTP Interworking 695

Example: SRTP-RTP Interworking 695
Example: Configuring Crypto Authentication 696
Example: Configuring Crypto Authentication (Dial Peer Level) 696
Example: Configuring Crypto Authentication (Global Level) 696

CHAPTER 54

SRTP-SRTP Pass-Through 697

Feature Information for Support of SRTP-SRTP Pass-Through Calls 697
Information About SRTP-SRTP Pass-Through 698
Pass-Through of Unsupported Crypto Suites 698

Configure Pass-Through of Unsupported Crypto Suites for a Specific Dial Peer 699
Configure Pass-Through of Unsupported Crypto Suites Globally 701

Configuration Examples for SRTP-SRTP Pass-Through 702
PART XIV

CHAPTER 55

High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms

About CUBE High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms

Box-to-Box Redundancy

Redundancy Group (RG) Infrastructure

Network Topology

Considerations and Restrictions

How to Configure CUBE High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms

Before You Begin

Configure High Availability

Configuration Examples

Example: Control Interface Protocol Configuration

Example: Redundancy Group Protocol Configuration

Example: Redundant Traffic Interface Configuration

Verify Your Configuration

Troubleshoot High Availability Issues

CHAPTER 56

High Availability on Cisco ASR 1000 Series Aggregation Services Routers

About CUBE High Availability on Cisco ASR 1000 Series Routers

Inbox Redundancy

Box-to-Box Redundancy

Redundancy Group (RG) Infrastructure

PROTECTED Mode

Network Topology

Considerations and Restrictions

How to Configure CUBE High Availability on Cisco ASR 1000 Series Router
Feature Information for Consumption of Multiple Forked 18x Responses with SDP During Early Dialog 865
Prerequisites 866
Restrictions 866
Information About Consumption of Forked 18x Responses with SDP During Early Dialog 866
Characteristics of Forked 18x Responses with SDP during Early Dialog 866
Configuring Consumption of Forked 18x Responses with SDP During Early Dialog 867
Configuring Consumption of Forked 18x Responses with SDP During Early Dialog Renegotiate 868
Troubleshooting Tips 870

CHAPTER 66
Support for Pass-Through of Unsupported Content Types in SIP INFO Messages 871
Feature Information 871
Configure SIP INFO Message with Unsupported Content Type 871
Information About Pass-Through of Unsupported Content Types in SIP INFO Messages 872

CHAPTER 67
Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element 873
Feature Information for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element 883
Prerequisites for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element 884
Restrictions for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element 885
Configuring P-Header and Random-Contact Support on the Cisco Unified Border Element 885
Configuring P-Header Translation on a Cisco Unified Border Element 885
Configuring P-Header Translation on an Individual Dial Peer 886
Configuring P-Called-Party-Id Support on a Cisco Unified Border Element 887
Configuring P-Called-Party-Id Support on an Individual Dial Peer 888
Configuring Privacy Support on a Cisco Unified Border Element 889
Configuring Privacy Support on an Individual Dial Peer 890
Configuring Random-Contact Support on a Cisco Unified Border Element 891
Configuring Random-Contact Support for an Individual Dial Peer 893

PART XVII
SIP Supplementary Services 895
## CHAPTER 68  Dynamic Refer Handling  897
- Feature Information for Dynamic REFER Handling  897
- Prerequisites  898
- Restrictions  898
- Configuring REFER Passthrough with Unmodified Refer-to  898
- Configuring REFER Consumption  900
- Troubleshooting Tips  902

## CHAPTER 69  Cause Code Mapping  903
- Feature Information for Cause Code Mapping  903
- Cause Code Mapping  904
- Configuring Cause Code Mapping  905
- Verifying Cause Code Mapping  906

## PART XVIII  Hosted and Cloud Services  909

## CHAPTER 70  Hosted and Cloud Services Delivery with CUBE  911

## CHAPTER 71  CUBE SIP Registration Proxy  913
- Registration Pass-Through Modes  913
  - End-to-End Mode  913
  - Peer-to-Peer Mode  914
- Registration in Different Registrar Modes  915
- Registration Overload Protection  916
  - Registration Overload Protection--Call Flow  916
- Registration Rate-limiting  916
  - Registration Rate-limiting Success--Call Flow  917
- Prerequisites for SIP Registration Proxy on Cisco UBE  917
- Restrictions  917
- Configuring CUBE SIP Registration Proxy  917
  - Enabling Local SIP Registrar  917
  - Configuring SIP Registration Proxy at the Global Level  919
  - Configuring SIP Registration Proxy at the Tenant Level  920
CHAPTER 72 Survivability for Hosted and Cloud Services 929

Information About Survivability for Hosted and Cloud Services 929

Advantages of Using CUBE Survivability Feature 929

Local Fallback 929

Registration Synchronization 930

Registration Through Alias Mapping 930

CUBE when WAN is UP 931

CUBE Survivability When WAN Is Down 932

How to Configure Survivability for Hosted and Cloud Services 934

Configuring Local Fallback or Registration Synchronization Globally 934

Configuring Local Fallback or Registration Synchronization at the Tenant Level 935

Configuring Local Fallback or Registration Synchronization on a Dial Peer 936

Configuring Survivability for Phones Sending Single Register Request 937

Configuring OPTIONS Ping 938

Configuring Registration Timer 939

Configuring the REGISTER Message Throttling in CUBE 940

Configuring the Class of Restrictions (COR) List 941

Verifying Survivability for Hosted and Cloud Services 943

Configuration Examples—Survivability for Hosted and Cloud Services 945

Example: Configuring Local Fallback Globally 945

Example: Configuring Local Fallback at the Tenant Level 946

Example: Configuring Local Fallback on a Dial Peer 946

Example: Configuring Survivability for Phones Sending Single Register Request 946

Example: Configuring OPTIONS Ping 946

Example: Configuring the Registration Timer 946

Example: Configuring REGISTER Message Throttling 947

Example: Configuring the COR List 947
CHAPTER 73

SUBSCRIBE-NOTIFY Passthrough 949
Restrictions for SUBSCRIBE-NOTIFY Passthrough 949
Information About SUBSCRIBE-NOTIFY Passthrough 950
SUBSCRIBE-NOTIFY Passthrough Request Routing 950
SUBSCRIBE-NOTIFY Passthrough Survivability Mode 951
Configure SUBSCRIBE-NOTIFY Passthrough 951
Configuring an Event List 951
Configuring SUBSCRIBE-NOTIFY Event Passthrough Globally 952
Configuring SUBSCRIBE-NOTIFY Event Passthrough at the Dial-Peer Level 953
Verifying SUBSCRIBE-NOTIFY Passthrough 954
Troubleshooting Tips 956
Configuration Examples for SUBSCRIBE-NOTIFY Passthrough 956
Example: Configuring an Event List 956
Example: Configuring SUBSCRIBE-NOTIFY Event Passthrough Globally 956
Example: Configuring SUBSCRIBE-NOTIFY Event Passthrough under a Dial Peer 957
Feature Information for SUBSCRIBE-NOTIFY Passthrough 957

PART XIX

Cisco Unified Communications Manager Line-Side Support 959

CHAPTER 74

Cisco Unified Communications Manager Line-Side Support 961
Feature Information for Cisco Unified Communications Manager Line-Side Support 961
Restrictions for Cisco Unified Communications Manager Line-Side Support 962
Information About Cisco Unified Communications Manager Line-Side Support 963
Cisco UBE Line-Side Deployment 963
Line-Side Deployment Scenarios 963
Line-Side Support for CUCM on CUBE 964
Configuring a PKI Trustpoint 965
Importing the CUCM and CAPF Key 966
Creating a CTL File 967
Configuring a Phone Proxy 968
Attaching a Phone Proxy to a Dial Peer 969
Verifying CUCM Lineside Support 971
CHAPTER 80

Support for Session Identifier 1041
Feature Information for Session Identifier Support 1041
Restrictions 1042
Information About Session Identifier 1042
Feature Behavior 1043
Configuring Support for Session Identifier 1043
Troubleshooting Tips 1043

PART XXIV

Security Compliance 1051

CHAPTER 81

Common Criteria (CC) and The Federal Information Processing Standards (FIPS) Compliance 1053
Feature Information for Common Criteria (CC) and the Federal Information Standards (FIPS) Compliance 1054
Supported Hardware and Software for Virtual CUBE 1054
Common Criteria Configuration on Cisco CSR 1000v 1054
Enable Common Criteria Mode 1054
SIP TLS Configuration 1055
SIP TLS Configuration Task Flow 1055
Generate RSA Public Key 1055
Configure Certificate Authority Server 1056
Configure CSR Trustpoint 1057
Configure Peer Trustpoint 1058
Add Client Verification Trustpoint 1059
Enforce Strict SRTP 1060
HTTPS TLS Configuration 1061
HTTPS TLS Configuration Task Flow 1061
Prepare Cisco CSR 1000v Router's HTTP Server to Run in CC Mode 1061
Create Certificate Map for HTTPS Peer Trustpoint 1062
Read Me First

Important Information

Note
For CUBE feature support information in Cisco IOS XE Bengaluru 17.6.1a and later releases, see Cisco Unified Border Element IOS-XE Configuration Guide.

Note
The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.

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.
Short Description

The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.

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: https://www.cisco.com/c/en/us/about/legal/trademarks.html. 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)
New and Changed Information

- New and Changed Information, on page 3

**Note**

- For detailed information on CUBE features supported on Cisco IOS Releases, Cisco IOS XE 3S Releases, and Cisco IOS XE Denali 16.3.1 and later Releases, refer to CUBE Cisco IOS Feature Roadmap, CUBE Cisco IOS XE 3S Feature Roadmap, and CUBE Cisco IOS XE Releases Feature Roadmap respectively.

- For CUBE feature support information for Cisco IOS XE Bengaluru 17.6.1a and later releases, see Cisco Unified Border Element IOS-XE Configuration Guide.

- H.323 protocol is no longer supported from Cisco IOS XE Bengaluru 17.6.1a onwards. Consider using SIP for multimedia applications.

- The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on RFP documentation, or language that is used by a referenced third-party product.

<table>
<thead>
<tr>
<th>Description</th>
<th>Documented at</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure forking of nonsecure calls through Media Proxy</td>
<td>CUBE Media Proxy, on page 569</td>
</tr>
<tr>
<td>Support for Cisco 8200L Catalyst Edge Series Platforms</td>
<td>Supported Platforms, on page 5</td>
</tr>
<tr>
<td>Support for VoIP Trace Serviceability Framework</td>
<td>VoIP Trace for CUBE, on page 1035</td>
</tr>
</tbody>
</table>
CHAPTER 3

Supported Platforms

Note Cisco Cloud Services Router 1000V Series (CSR 1000V) is no longer supported from Cisco IOS XE Bengaluru 17.4.1a onwards. If you are using CSR 1000V, you have to upgrade to Cisco Catalyst 8000V Edge Software (Catalyst 8000V). For End-of-Life information on CSR 1000V, see End-of-Sale and End-of-Life Announcement for the Select Cisco CSR 1000v Licenses.

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

Note For information on migrating from existing Cisco IOS XE 3S releases to the Cisco IOS XE Denali 16.3 release, seeCisco 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 Services Generation 2 Routers (ISR G2)</td>
<td>Cisco 2900 Series Integrated Services Routers&lt;br&gt;Cisco 3900 Series Integrated Services Routers</td>
<td>Cisco IOS 12 M and T&lt;br&gt;Cisco IOS 15 M and T&lt;br&gt;1</td>
</tr>
<tr>
<td>Cisco 4000 Series Integrated Services Routers (ISR G3)</td>
<td>Cisco 4321 Integrated Services Routers&lt;br&gt;Cisco 4331 Integrated Services Routers&lt;br&gt;Cisco 4351 Integrated Services Routers&lt;br&gt;Cisco 4431 Integrated Services Routers&lt;br&gt;Cisco 4451 Integrated Services Routers&lt;br&gt;Cisco 4461 Integrated Services Routers</td>
<td>Cisco IOS XE 3S&lt;br&gt;Cisco IOS XE Denali 16.3.1 onwards&lt;br&gt;2</td>
</tr>
<tr>
<td>Cisco 1000 Series Integrated Services Routers (ISR)</td>
<td>All router models belonging to Cisco 1100 Integrated Services Routers</td>
<td>Cisco IOS XE Gibraltar 16.12.1a onwards</td>
</tr>
</tbody>
</table>
### Supported Platforms

<table>
<thead>
<tr>
<th><strong>Cisco Router Platforms</strong></th>
<th><strong>Cisco Router Models</strong></th>
<th><strong>Cisco IOS Software Releases</strong></th>
</tr>
</thead>
</table>
| Cisco Aggregated Services Routers (ASR) | Cisco ASR1001-X Aggregated Services Routers  
Cisco ASR1002-X Aggregated Services Routers  
Cisco ASR1004 Aggregated Services Routers with RP2  
Cisco ASR1006 Aggregated Services Routers with RP2 and ESP40 | Cisco IOS XE 3S  
Cisco IOS XE Denali 16.3.1 onwards |
|  | Cisco ASR1006-X Aggregated Services Routers with RP2 and ESP40 | Cisco IOS XE Everest 16.6.1 onwards |
|  | Cisco ASR1006-X Aggregated Services Routers with RP3 and ESP40/ESP100 | Cisco IOS XE Everest 16.6.1 onwards |
|  | Cisco ASR1006-X Aggregated Services Routers with RP3 and ESP100X | Cisco IOS XE Amsterdam 17.3.2 onwards |
| 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 |
| Cisco Catalyst 8000V Edge Software (Catalyst 8000V) | Cisco Catalyst 8000V Edge Software (Catalyst 8000V) | Cisco IOS XE Bengaluru 17.4.1a onwards |
| Cisco 8300 Catalyst Edge Series Platforms | C8300-1N1S-6T  
C8300-1N1S-4T2X  
C8300-2N2S-6T  
C8300-2N2S-4T2X | Cisco IOS XE Amsterdam 17.3.2 |
| Cisco 8200 Catalyst Edge Series Platform | C8200-1N-4T | Cisco IOS XE Bengaluru 17.4.1a |
| Cisco 8200L Catalyst Edge Series Platform | C8200L-1N-4T | Cisco IOS XE Bengaluru 17.5.1a |

1. Support for CUBE on Cisco 2900 Series Integrated Services Routers and Cisco 3900 Series Integrated Services Routers are only up to release 15.7 M.
2. 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.

- Feature Comparison on Supported Platforms, on page 7
# Feature Comparison on Supported Platforms

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

## Note
Collaboration feature support on Cisco ISR 4000 Series Routers is available from Cisco IOS XE Release 3.13.1S onwards. Cisco Cloud Services Routers 1000V Series support is available from Cisco IOS XE Release 3.15S onwards.

## Table 1: Feature Comparisons for 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 ISR 1000 Series Routers</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Availability Implementation</td>
<td>Redundancy Group Infrastructure</td>
<td>Hot Standby Protocol (HSRP) Based</td>
<td>Redundancy Group Infrastructure</td>
<td>No</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>
</tr>
<tr>
<td>DSP Card Type</td>
<td>SPA-DSP</td>
<td>PVDM2/PVDM3</td>
<td>PVDM4 SM-X-PVDM</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>
</tr>
<tr>
<td>Transcoder—LTI</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</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>Yes</td>
</tr>
<tr>
<td>Noise Reduction and ASP</td>
<td>Yes</td>
<td>Yes (Cisco IOS Release 15.2(3)T onwards)</td>
<td>Yes</td>
<td>No</td>
</tr>
<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>
</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 ISR 1000 Series Routers</th>
</tr>
</thead>
<tbody>
<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>
</tr>
<tr>
<td>CUBE for SP Managed and Hosted Services</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 SIP SRST is not supported</td>
<td>Yes (Cisco IOS XE Fuji 16.7.1 Release onwards)</td>
<td>Yes. From Cisco IOS XE Bengaluru 17.5.1a</td>
</tr>
<tr>
<td>IPv6</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2: Feature Comparisons for Supported Platforms (Contd...)

<table>
<thead>
<tr>
<th>Features</th>
<th>Cisco CSR 1000V Series Routers</th>
<th>Cisco 8000V Catalyst Series Edge Platforms</th>
<th>Cisco 8300 Catalyst Edge Series Platforms</th>
<th>Cisco 8200 Catalyst Edge Series Platforms</th>
<th>Cisco 8200L Catalyst Edge Series Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA Implementation</td>
<td>RG Infrastructure</td>
<td>RG Infrastructure</td>
<td>RG Infrastructure</td>
<td>RG Infrastructure</td>
<td>RG Infrastructure</td>
</tr>
<tr>
<td>Media Forking</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DSP Card Type</td>
<td>No</td>
<td>No</td>
<td>NIM-PVDM</td>
<td>NIM-PVDM</td>
<td>NIM-PVDM</td>
</tr>
<tr>
<td>Transcoder registered to CUCM</td>
<td>No</td>
<td>No</td>
<td>Yes (via SCCP)</td>
<td>Yes (via SCCP)</td>
<td>Yes (via SCCP)</td>
</tr>
<tr>
<td>Transcoder—LTI</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cisco UC Gateway Services API</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Noise Reduction &amp; ASP</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Call Progress Analysis</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Supported Platforms

#### Feature Comparison on Supported Platforms

<table>
<thead>
<tr>
<th>Features</th>
<th>Cisco CSR 1000V Series Routers</th>
<th>Cisco 8000V Catalyst Series Edge Platforms</th>
<th>Cisco 8300 Catalyst Edge Series Platforms</th>
<th>Cisco 8200 Catalyst Edge Series Platforms</th>
<th>Cisco 8200L Catalyst Edge Series Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRTP-RTP Interworking</td>
<td>Yes - No DSP resources required</td>
<td>Yes - No DSP resources required</td>
<td>Yes - No DSP resources required</td>
<td>Yes - No DSP resources required</td>
<td>Yes - No DSP resources required</td>
</tr>
<tr>
<td></td>
<td>(Cisco IOS XE Release 3.15S onwards)</td>
<td></td>
<td></td>
<td></td>
<td></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>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Note**

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 25
• Dial-Peer Matching, on page 31
• DTMF Relay, on page 37
• Introduction to Codecs, on page 51
• Call Admission Control, on page 65
• Basic SIP Configuration, on page 83
• SIP Binding, on page 111
• Media Path, on page 127
• SIP Profiles, on page 135
• SIP Out-of-Dialog OPTIONS Ping Group, on page 163
• Configure TCL IVR Applications, on page 171
• VoIP for IPv6, on page 191
• Monitoring of Phantom Packets, on page 247
• Configurable SIP Parameters via DHCP, on page 253
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.

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

**Note**

H.323 protocol is no longer supported from Cisco IOS XE Bengaluru 17.6.1a onwards. Consider using SIP for multimedia applications.

The Session Initiation Protocol (SIP) is a signaling communications protocol, widely used for controlling multimedia communication sessions such as voice and video calls over IP networks. SIP (or H.323) trunking is the use of VoIP to facilitate the connection of 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 is configured:

- Voice dial-peer with session protocol as SIP.
- voice register global
- sip-ua

In the releases before Cisco IOS XE Gibraltar 16.11.1a, the following commands 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.
How to Configure Basic CUBE Features

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 | periodicity [mins value | hours value | days value] ]
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><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# 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> Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> mode border-element license [capacity sessions</td>
<td>Enables CUBE configuration and configures the number of licenses (capacity).</td>
</tr>
<tr>
<td>periodicity {mins value</td>
<td>hours value</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(conf-voi-serv)# mode border-element license capacity 200</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# mode border-element license periodicity days 15</td>
<td></td>
</tr>
</tbody>
</table>

#### Effective from Cisco IOS XE Amsterdam 17.2.1r, the **capacity** keyword and **sessions** argument are deprecated. However, the keyword and argument are available in the Command Line Interface (CLI). If you try to configure license capacity using CLI, the following error message is displayed:

```
Error: CUBE SIP trunk licensing is now based on dynamic session counting. Static license capacity configuration has been deprecated.
```

#### Effective from Cisco IOS XE Amsterdam 17.2.1r, the **periodicity** keyword and [mins hours days] argument are introduced. The **periodicity** keyword configures periodicity interval for license entitlement requests for CUBE. If you do not configure license periodicity, the default license period of 7 days is enabled.
### Purpose

Command or Action | Purpose
---|---

**Note**  
We recommend you to configure interval in days. Configuring interval in minutes or hours increases the frequency of entitlement requests and thereby increases the processing load on Cisco Smart Software Manager (CSSM). License periodicity configuration of minutes or hours is recommended to be used only with Cisco Smart Software Manager On-Prem (formerly known as Cisco Smart Software Manager satellite) mode.

---

### Step 5

**allow-connections from-type to to-type**  
**Example:**

Device(conf-voi-serv)# allow-connections sip to sip

Allows connections between specific types of endpoints in a VoIP network.

- The two protocols (endpoints) refer to the VoIP protocols (SIP or H.323) on the two call legs.

---

### Step 6

**end**  
**Example:**

Device(conf-voi-serv)# end

Returns to privileged EXEC mode.

---

### 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. In releases before Cisco IOS XE Amsterdam 17.2.1r, CUBE status display is enabled only if **mode border-element** command is configured with call license capacity. Effective from Cisco IOS XE Amsterdam 17.2.1r, this dependency is removed and Licensed-Capacity information is excluded from output.  
**Example:**
Before Cisco IOS XE Amsterdam 17.2.1r:

Device# 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

Effective from Cisco IOS XE Amsterdam 17.2.1r:

Device# show cube status

CUBE-Version: 12.8.0
SW-Version: 17.2.1, Platform CSR1000V
HA-Type: none

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>Step 1</td>
<td></td>
</tr>
<tr>
<td><strong>enable</strong></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</td>
<td></td>
</tr>
<tr>
<td><strong>configure terminal</strong></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>Step 3</td>
<td></td>
</tr>
<tr>
<td><strong>voice service voip</strong></td>
<td>Enters global 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>Step 4</td>
<td></td>
</tr>
<tr>
<td><strong>ip address trusted list</strong></td>
<td>Enters IP address trusted list mode and enables</td>
</tr>
<tr>
<td>Example:</td>
<td>the addition of valid IP addresses.</td>
</tr>
<tr>
<td>Device(conf-voi-serv)# ip address trusted list</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step 5</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ipv4 ipv4-address [network-mask]</strong></td>
<td>Allows you to add up to 100 IPv4 addresses in the IP address trusted list. Duplicate IP addresses are not allowed.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(cfg-iptrust-list)# ipv4 192.0.2.1 255.255.255.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 6</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ipv6 ipv6-address</strong></td>
<td>Allows you to add IPv6 addresses to the trusted IP address list.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(cfg-iptrust-list)# ipv6 2001:DB8:0:ABCD::1/48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 7</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>end</strong></td>
<td>Returns to privileged EXEC mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(cfg-iptrust-list)# end</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 5

Virtual CUBE

The Cisco Unified Border Element (CUBE) feature set has traditionally been delivered with hardware router platforms, such as the Cisco Integrated Services Router (ISR) series. A subset of CUBE features (vCUBE) may be used in virtualized environments with the Cisco CSR 1000v Series Cloud Services Router or Cisco Catalyst 8000V Edge Software (Catalyst 8000V).

Note

When upgrading to Catalyst 8000V software from a CSR1000V release, an existing throughput configuration will be reset to a maximum of 250 Mbps. Install an HSEC authorization code, which you can obtain from your Smart License account, before reconfiguring your required throughput level.

- Feature Information for Virtual CUBE, on page 25
- Prerequisites for Virtual CUBE, on page 26
- Features Supported with Virtual CUBE, on page 27
- Restrictions, on page 27
- Information about Virtual CUBE, on page 27
- Install Virtual CUBE on ESXi, on page 28
- How to Enable Virtual CUBE, on page 29
- Troubleshooting Virtual CUBE, 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>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual CUBE in Cisco Catalyst 8000V Edge Software (Catalyst 8000V)</td>
<td>Cisco IOS XE Bengaluru 17.4.1a</td>
<td>Virtual CUBE introduced for Cisco Catalyst 8000V Edge Software (Catalyst 8000V) in VMware ESXi and AWS environments.</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>vCUBE in Amazon Web Services (AWS)</td>
<td>Cisco IOS XE Gibraltar 16.12.4a</td>
<td>vCUBE offer introduced in AWS for Cisco CSR 1000v Series Cloud Services Router.</td>
</tr>
<tr>
<td>Virtual CUBE</td>
<td>Cisco IOS XE 3.15S</td>
<td>Virtual CUBE introduced for Cisco CSR 1000v Series Cloud Services Router in VMware ESXi environments.</td>
</tr>
</tbody>
</table>

Prerequisites for Virtual CUBE

Hardware

- The vCUBE feature set is bundled as part of the Cisco virtual router software and is used when deployed in VMware ESXi virtualized environments. For more information on how to deploy Cisco virtualized routers in VMware ESXi environments, see [Installing the Cisco CSR 1000V in VMware ESXi Environments](#) and [Installing in VMware ESXi Environment](#).
- For information on the best practices for setting ESXi host BIOS parameters for performance, see [BIOS Settings](#).
- Virtual CUBE is supported on the CSR 1000V and C8000V platforms.
- Virtual CUBE is also supported in AWS. You must use the AWS Marketplace product listing for virtual CUBE.
- For more information about the Cisco CSR 1000V in AWS, see [Cisco CSR 1000V Series Cloud Services Router Deployment Guide for Amazon Web Services](#).

Note

- The CSR1000V and Catalyst 8000V product may be used in several different public and private cloud environments. However, vCUBE is only supported when deployed on VMware ESXi and AWS platforms currently.
- When you use a consolidated (.bin) image to upgrade a CSR 1000V medium configuration (2 vCPU, 4 GB RAM) to Catalyst 8000V, you must change the virtual machine vRAM allocation to at least 5 GB to ensure advertised performance. Alternatively and when deploying in AWS environments, boot the router using individual packages rather than a consolidated image without the need for additional memory. Refer to [Installing Subpackages from a Consolidated Package](#) for details.

Software

- Obtain the relevant license for the router platform. See [Virtual CUBE Licensing Requirements](#), on page 28 for more information.
- In AWS, only Bring Your Own License (BYOL) is supported for vCUBE. Pay as You Go (Subscription) versions of the CSR 1000V and C8000V are not supported. Make sure you choose the vCUBE AWS Marketplace product listing. Refer to [Cisco Virtual CUBE-BYOL](#).
For more information about Cisco virtual routers, see CSR 1000V Data Sheet and Catalyst 8000V Data Sheet.

## Features Supported with Virtual CUBE

vCUBE supports most of the CUBE features available in IOS XE releases. vCUBE does not support the following:

- DSP-based features
  - Codec Transcoding, Transrating
  - Raw Inband to RTP-NTE DTMF Interworking
  - Call progress Analysis (CPA)
  - Noise Reduction (NR), Acoustic Shock Protection (ASP), and Audio Gain

- H.323 Interworking
- IOS-based Hardware Media Termination Point (MTP)

---

**Note**

CUBE high availability is not currently supported on vCUBE when deployed in AWS.

## Restrictions

- Software MTP is not supported.
- CSR1000V used as MTP/TRP for CUCM is not supported.

---

**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

### Media

vCUBE media performance depends on the underlying host platform consistently providing packet switching latency of less than 5 milliseconds. The recommended hardware and virtual machine configurations ensure this performance when followed closely.

For more information on how to monitor media performance, see Voice Quality Monitoring.
Virtual CUBE Licensing Requirements

For information about licensing of virtual CUBE with CSR1000V and C8000V, refer to CUBE Smart Licensing.

Virtual CUBE with CSR1000V

vCUBE is enabled for the CSR1000V with the APPX and AX platform licenses. vCUBE processes and CLI commands are enabled when either of these licenses are enabled. Secure call features require the AX license. In common with all CUBE instances, L-CUBE Smart License options are required for each active session.

The following table details the license requirements for Virtual CUBE on the CSR1000V.

<table>
<thead>
<tr>
<th>Virtual CUBE Session License</th>
<th>Platform License</th>
<th>Features</th>
<th>Throughput License</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-CUBE Smart License options</td>
<td>APPX</td>
<td>No TLS / SRTP support</td>
<td>Session count * (signaling + bidirectional media bandwidth)</td>
</tr>
<tr>
<td></td>
<td>AX</td>
<td>All vCUBE features</td>
<td></td>
</tr>
</tbody>
</table>

For detailed information about licensing, see Cisco CSR 1000v Software Configuration Guide.

Virtual CUBE with Catalyst 8000V

vCUBE is enabled for the Catalyst 8000V with the DNA Network Essentials license.

<table>
<thead>
<tr>
<th>Virtual CUBE Session License</th>
<th>DNA Subscription</th>
<th>Features</th>
<th>DNA Bandwidth License</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-CUBE Smart License options</td>
<td>Essentials or above</td>
<td>All vCUBE features</td>
<td>Session count * (signaling + bidirectional media bandwidth)/2</td>
</tr>
</tbody>
</table>

For detailed information on licensing, see Licensing.

Install Virtual CUBE on ESXi

SUMMARY STEPS

1. Use the CSR1000V or the Catalyst 8000V OVA application file (available from software.cisco.com) to deploy a new virtual instance directly in VMware ESXi.

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> Use the CSR1000V or the Catalyst 8000V OVA application file (available from software.cisco.com) to deploy a new virtual instance directly in VMware ESXi.</td>
<td>Note: Select the required instance size during the OVA deployment. For further details on how to perform the deployment, see Cisco CSR 1000V Series Cloud Services Router Software Configuration Guide or Cisco Catalyst 8000V Edge Software Installation And Configuration Guide.</td>
</tr>
</tbody>
</table>
How to Enable Virtual CUBE

SUMMARY STEPS

1. Power on the virtual machine.
2. Enable platform and throughput licenses and register to a Cisco licensing server.
3. Enable virtual CUBE using the steps in Enabling the CUBE Application on a Device.

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>Power on the virtual machine.</td>
<td>Powers on the vCUBE.</td>
</tr>
<tr>
<td>Step 2</td>
<td>Enable platform and throughput licenses and register to a Cisco licensing server.</td>
<td>Enables platform and throughput licenses and registers that virtual CUBE to a licensing server.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Enable virtual CUBE using the steps in Enabling the CUBE Application on a Device.</td>
<td>Enables vCUBE on a device.</td>
</tr>
</tbody>
</table>

Troubleshooting Virtual CUBE

To troubleshoot vCUBE, follow the same procedure for Cisco ASR routers. This procedure includes crash file decoding, decoding traceback, and so on. For more details, see Troubleshoot Cisco ASR 1000 Series Aggregation Services Routers Crashes.

To troubleshoot virtual machine issues, see Cisco CSR 1000V Series Cloud Services Router Software Configuration Guide and Cisco Catalyst 8000V Edge 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.

**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.

- 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 359.

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-nt

CUCM sending 9 + All digits dialed (Outgoing calls)

Incoming call number used to match the inbound LAN dial peer

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-nt

SP will be sending 10 digits inbound (Incoming Calls)

Destination pattern used to match the outbound LAN dial peer

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.
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 4: 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 <strong>DNIS-string</strong></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 <strong>ANI-string</strong></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 <strong>ANI-string</strong></td>
<td>This command uses the inbound call leg to the inbound dial peer.</td>
<td>ANI string for inbound</td>
</tr>
</tbody>
</table>
### Command in Dial-Peer Configuration

<table>
<thead>
<tr>
<th>Description</th>
<th>Call Setup Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.164 Patterns</td>
<td>E.164 Patterns</td>
</tr>
</tbody>
</table>

#### DNIS

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.

#### Directory URI

This command uses the directory URI (Uniform Resource Identifier) number of an incoming INVITE from a SIP entity to match an inbound dial peer. This directory URI is part of the SIP address of a device.

The command calls a globally defined voice class identifier where the directory URI is configured. It requires the configuration of session protocol sipv2.

#### Preference for Dial-Peer Matching

The following is the order in which inbound dial-peer is matched for SIP call-legs:

1. **E.164 Patterns**
2. **Directory URI**
3. **Session Protocol SIPv2**
4. **Session Protocol SIPv1**
5. **Voice Class**

### Table 5: 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>destination-pattern DNIS-string</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>destination URI-class-identifier</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>destination e164-pattern-map pattern-map-group-id</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>
• voice class uri $URI-class-identifier$ with incoming uri $\{via\} URI-class-identifier$

• voice class uri $URI-class-identifier$ with incoming uri $\{request\} URI-class-identifier$

• voice class uri $URI-class-identifier$ with incoming uri $\{to\} URI-class-identifier$

• voice class uri $URI-class-identifier$ with incoming uri $\{from\} URI-class-identifier$

• incoming called-number $DNIS-string$

• answer-address $ANI-string$

The following is the order in which inbound dial-peer is matched for H.323 call-legs:

• incoming uri $\{called\} URI-class-identifier$

• incoming uri $\{calling\} URI-class-identifier$

• incoming called-number $DNIS-string$

• answer-address $ANI-string$

The following is the order in which outbound dial-peer is matched for SIP call-legs:

• destination route-string

• destination $URI-class-identifier$ with target carrier-id $string$

• destination-pattern with target carrier-id $string$

• destination $URI-class-identifier$

• destination-pattern

• target carrier-id $string$

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.
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](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

**Table 6: 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 Release 12.1(2)T</td>
<td>The DTMF relay feature allows CUBE to send DTMF digits over IP.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 2.1</td>
<td>The <code>dtmf-relay</code> command was added.</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 the DTMF relay by default and to enable, requires DTMF relay capabilities.

Note

Note: DTMF tones that are sent by phones do not traverse the CUBE.

DTMF Relay

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

In-band DTMF-Relay passes the DTMF digits using the RTP media stream. It 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

Note: The main advantage of DTMF relay is that in-band DTMF relay sends low-bandwidth codecs such as the G.729 and G.723 with greater fidelity. Without the use of DTMF relay, calls established with low-bandwidth codecs has trouble accessing automated DTMF-based systems. For example, voicemail, 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.

The VoIP compressed code causes the loss of integrity of the DTMF digits. However, the DTMF relay prevents the loss of integrity of DTMF digits. The relayed DTMF regenerates transparently on the peer side.

Figure 10: DTMF Relay Mechanism
The following lists the DTMF relay mechanisms that support the VoIP dial-peers based on the configured keywords. The DTMF relay mechanism can be either out-of-band (H.323 or SIP) or in-band (RTP).

- **h245-alphanumeric and h245-signal**—These two methods are available only on H.323 dial peers. It 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. It addresses a potential problem with the alphanumeric method when interworking with other vendors' systems.

- **sip-notify**—This method is available on the SIP dial peers only. It is a Cisco proprietary out-of-band DTMF relay mechanism that transports DTMF signals using SIP-Notify message. The SIP Call-Info header indicates the use of the SIP-Notify DTMF relay mechanism. Acknowledging the message with a 18x or 200 response message containing a similar SIP Call-Info header.

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

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

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

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

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

- **sip-kpml**—This method is available only on SIP dial peers. The RFC 4730 defines the out-of-band DTMF relay mechanism to register the DTMF signals using the SIP-Subscribe messages. It transports the DTMF signals using the SIP-Notify messages containing an XML-encoded body. This method is called Key Press Markup Language.

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

  A registered SIP endpoint to Cisco Unified Communications Manager or Cisco Unified Communications Manager Express uses this method. 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 a rtp-nte payload, a SIP Call-Info header, and an Allow-Events header with KPML.

  The following SIP-Notify message displays after the subscription. The endpoints transmit digits using SIP-Notify messages with KPML events through XML. The following example transmits, the digit “1”:

  ```
  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. It 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 enables for SIP dial peers, and invokes on receiving a SIP INFO message with DTMF relay content.

  The gateway receives the following sample SIP INFO message with specifics about the DTMF tone. 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). The RFC2833 defines the in-band DTMF relay mechanism. RFC2833 defines the formats of NTE-RTP packets to transport DTMF digits, hookflash, and other telephony events between two peer endpoints. Using the RTP stream, sends the DTMF tones as packet data after establishing call media. It is differentiated from the audio by the RTP payload type field, preventing compression of DTMF-based RTP packets. For example, sending the audio of a call on a session with an RTP payload type identifies it as G.711 data. Similarly sending the DTMF packets with an RTP payload type 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 a reliable digit relay between Cisco VoIP gateways on using a low-bandwidth codec.

---

**Note** By default, Cisco device uses Payload type 96 and 97 for fax. 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 negotiate between the two ends at call setup. They use the Session Description Protocol (SDP) within the body section of the SIP message.

---

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

- cisco-rtp—It 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. Cisco Unified Communications Manager does not support this method.

**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**—It 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 A-law or mu-law can carry in-band DTMF reliably. Female voices sometimes trigger the recognition of a DTMF tone.

The DTMF digits pass like the rest of your voice as normal audio tones with no special coding or markers. It uses the same codec as your voice, generated by your phone.

### Configuring DTMF Relays

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

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

Use any of the following variables `method`:

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

Configure multiple DTMF methods 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 configured DTMF relay mechanisms on CUBE, an MTP or transcoder is required.

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

<table>
<thead>
<tr>
<th>Table 7: Supported H.323 and SIP DTMF Relay Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.323 Gateway</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>In-band</td>
</tr>
<tr>
<td>Out-of-band</td>
</tr>
</tbody>
</table>
Interoperability and Priority with Multiple DTMF Relay Methods

• CUBE negotiates both rtp-nte and sip-kmpl if both support and advertise in the incoming INVITE. However, if CUBE does not initiate sip-kmpl, CUBE relies on the rtp-nte DTMF method to receive digits and a SUBSCRIBE. CUBE still accepts SUBSCRIBE for KPML. It 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 SDP 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, refer table 3 if you must configure sip-kpml on an inbound dial peer and h245-signaling on an outbound dial peer in an RTP-RTP Flow through configuration. The table shows that the combination supports (as image information is present) the required image IOS 12.4(15)T or IOS XE or above. The following are the call scenarios provided:

• 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 8: RTP-RTP Flow-Through

<table>
<thead>
<tr>
<th>Inbound dial-peer protocol</th>
<th>DTMF Relay Type</th>
<th>H.323</th>
<th>SIP</th>
<th>In-band</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>h245-alphanumeric</td>
<td>h245-signal</td>
<td>Rtp-nte</td>
</tr>
<tr>
<td>H.323</td>
<td></td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h245-signal</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<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>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-kpml</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-notify</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-info</td>
<td></td>
<td>Supported</td>
<td>Supported*</td>
</tr>
<tr>
<td>In-band</td>
<td>Voice in-band (G.711)</td>
<td></td>
<td></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.

### Table 9: RTP-RTP with DSP Involved Flow-Through Calls

<table>
<thead>
<tr>
<th>Inbound dial-peer protocol</th>
<th>DTMF Relay Type</th>
<th>H.323</th>
<th>SIP</th>
<th>In-band</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>h245-alphanumeric</td>
<td>h245-signal</td>
<td>Rtp-nte</td>
</tr>
<tr>
<td>H.323</td>
<td></td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h245-signal</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rtp-nte</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 10: RTP-RTP Flow Around

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

* media resource is required (Transcoder) for IOS versions. CUBE falls back to flow-through mode if the media resource is unavailable.
### Table 11: RTP-RTP with High-Density Transcoder Flow Through

<table>
<thead>
<tr>
<th>Inbound dial-peer relay type</th>
<th>DTMF Relay Type</th>
<th>H.323</th>
<th>SIP</th>
<th>In-band</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.323</td>
<td>h245-alphanumeric</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>h245-signal</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>rtp-nte</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>SIP</td>
<td>rtp-nte</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-kpml</td>
<td>Supported</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-notify</td>
<td>Supported</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-info</td>
<td></td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>In-band</td>
<td>Voice in-band (G.711)</td>
<td>Supported</td>
<td>Supported</td>
<td></td>
</tr>
</tbody>
</table>

### Table 12: SRTP-RTP Flow Through

<table>
<thead>
<tr>
<th>Inbound dial-peer relay type</th>
<th>DTMF Relay Type</th>
<th>H.323</th>
<th>SIP</th>
<th>In-band</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.323</td>
<td>h245-alphanumeric</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>h245-signal</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>rtp-nte</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>SIP</td>
<td>rtp-nte</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>sip-kpml</td>
<td>Supported</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-notify</td>
<td>Supported</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sip-info</td>
<td></td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>In-band</td>
<td>Voice in-band (G.711)</td>
<td>Supported</td>
<td>Supported</td>
<td></td>
</tr>
</tbody>
</table>
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-B4F9B5F60172.18.193.251
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 8888
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
```
Step 2  

**show sip-ua calls dtmf-relay sip-info**

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

**Example:**

```
Device# show sip-ua calls 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 : 9598A547-5513111E2-8008F709-2470C996@172.27.161.122
State of the call : STATE_ACTIVE (7)
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 (7)
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 (7)
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 : 9598A547-5513111E2-8008F709-2470C996@172.27.161.122
State of the call : STATE_ACTIVE (7)
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
```
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-78C438FF@10.86.176.119
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 257
No. Timestamp Digit Duration
Call 2
SIP Call ID : 22BC36A5-F01411E3-81808A6A-5FE95113@10.86.176.142
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 256
Number of SIP User Agent Client(UAC) calls: 2
```

```
SIP UAS CALL INFO
Call 1
SIP Call ID : 22BC36A5-F01411E3-81808A6A-5FE95113@10.86.176.142
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 256
No. Timestamp Digit Duration
Call 2
SIP Call ID : D0498774-F01311E3-82A0DE9F-78C438FF@10.86.176.119
  State of the call : STATE_ACTIVE (7)
  Calling Number : 2017
  Called Number : 1011
  CC Call ID : 257
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
```

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

```
Number of SIP User Agent Server(UAS) calls: 2
```

```
```
```
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-78C438FF8010.86.176.119
  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-78C438FF8010.86.176.119
  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
Introduction to Codecs

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, on page 51
• Voice Media Transmission, on page 52
• Voice Activity Detection, on page 53
• VoIP Bandwidth Requirements, on page 54
• Supported Audio and Video Codecs, on page 56
• How to Configure Codecs, on page 57
• Configuration Examples for Codecs, on page 62

Why CUBE Needs Codecs

CUBE uses codecs to compress digital voice samples to reduce bandwidth usage per call. Refer to Table 14: 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.

**Restrictions for Voice-Class Codec Transparent**

- While using the voice-class codec transparent, only the offer is passed transparently (without filtering). Codec filtering is done on the SDP present in answer and the first codec is passed to other side.
- CUBE does not support Early-Offer to Delayed-Offer (EO-DO) call flows.

---

**Note**

You can use 'pass-thru content sdp', if you do not want to involve CUBE in the codec negotiation.

---

**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.

![Figure 12: Compressed RTP](image)

- 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.

Note
VAD to NO-VAD calls are not supported by CUBE.

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 13: 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 14: Codec and Bandwidth Information

<table>
<thead>
<tr>
<th>Codec &amp; Bit Rate (kbps)</th>
<th>Codec Simple Size (Bytes)</th>
<th>Codec Simple 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</td>
<td>20</td>
<td>50</td>
<td>42.8</td>
<td>27.6</td>
<td>47.2</td>
<td></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>Codec &amp; Bit Rate (kbps)</td>
<td>Codec Sample Size (Bytes)</td>
<td>Codec Sample Interval (ms)</td>
<td>Mean Opinion Score (MOS)</td>
<td>Voice Payload Size (Bytes)</td>
<td>Voice Payload Size (ms)</td>
<td>Payload Size (ms)</td>
<td>Packets Per Second (PPS)</td>
<td>Bandwidth MP or FRF.12 (kbps)</td>
<td>Bandwidth w/cRTP MP or FRF.12 (kbps)</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------</td>
<td>----------------------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>---------------------------</td>
<td>-------------------------------</td>
<td>----------------------------------</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></td>
<td>82.8</td>
<td>67.6</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></td>
<td>34.0</td>
<td>18.8</td>
</tr>
<tr>
<td>ilbc_mode_30 (1333 kbps)</td>
<td>50</td>
<td>30</td>
<td>NA</td>
<td>50</td>
<td>30</td>
<td>33.3</td>
<td></td>
<td>25.867</td>
<td>15.73</td>
</tr>
</tbody>
</table>

**Table 15: 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>
<tbody>
<tr>
<td>Codec Sample Size (Bytes)</td>
<td>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).</td>
</tr>
<tr>
<td>Codec Sample Interval (ms)</td>
<td>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).</td>
</tr>
<tr>
<td>MOS</td>
<td>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.</td>
</tr>
<tr>
<td>Voice Payload Size (Bytes)</td>
<td>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.</td>
</tr>
<tr>
<td>Voice Payload Size (ms)</td>
<td>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 bytes * 8) / (20 ms) = 8 kbps ]</td>
</tr>
</tbody>
</table>
**PPS**

PPS represents the number of packets that need to be transmitted every second in order to deliver the codec bitrate. 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 pps = (8 kbps) / (160 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 16: Audio Codecs Supported on CUBE*

<table>
<thead>
<tr>
<th>Codec Keyword</th>
<th>Codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>acld</td>
<td>AACLD 90000 bps</td>
</tr>
<tr>
<td>clear-channel</td>
<td>Clear Channel 64000 bps (No voice capabilities: data transport only)</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>
</tbody>
</table>
Codec Keyword | Codec
--- | ---
g728 | G.728 16000 bps
g729br8 | G.729 ANNEX-B 8000 bps (contains built-in VAD that cannot be disabled)
g729r8 | G.729 8000 bps
gsmamr-nb | GSM AMR-NB 4750 to 12200 bps (contains built-in VAD that cannot be disabled)
ilbc | iLBC 13330 or 15200 bps
isac | iSAC 10 to 32 kbps (variable bit-rate)
mp4a-latm | MP4A-LATM up to 128 kbps
transparent | Transparent; uses the endpoint codec
opus | Opus up to 510 kbps

Table 17: Video Codecs Supported on CUBE

| Codec Keyword | Codec |
--- | --- |
h261 | Video Codec H261 |
h263 | Video Codec H263 |
h263+ | Video Codec H263+ |
h264 | Video Codec H264 |
mpeg4 | Video Codec MPEG-4 ISO/IES 14496-2 |

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]]
5. Do the following to configure a video codec:

   - video codec codec

6. (Optional) Do one of the following to configure RTP payload type:

   - rtp payload-type cisco-codec-isac number
   - rtp payload-type cisco-codec-ilbc number
   - rtp payload-type cisco-codec-video-h263+ number
   - rtp payload-type cisco-codec-video-h264 number
   - rtp payload-type opus number

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>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&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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 1 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> Enter one of the following to configure an audio codec:</td>
<td>Configures an audio codec at the dial peer level.</td>
</tr>
<tr>
<td>codec codec [bytes payload-size fixed-bytes ]</td>
<td>• g729r8, 20-byte payload is configured by default.</td>
</tr>
<tr>
<td>codec isac [mode {adaptive</td>
<td>independent} [bit-rate value framesize { 30</td>
</tr>
<tr>
<td>codec ilbc [mode frame-size [bytes payload-size]]</td>
<td></td>
</tr>
<tr>
<td>codec mp4-latm [profile tag]</td>
<td></td>
</tr>
<tr>
<td>codec opus [profile tag]</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>For g711alaw Codec</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# codec g711alaw</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>For ISAC Codec</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# codec isac mode independent</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

Command or Action | Purpose
--- | ---
**Step 5** | Do the following to configure a video codec:
• **video codec codec**

**Example:**
For Video Codec
Device(config-dial-peer)# video codec h261

Configures a video codec at the dial peer level.

**Step 6** | (Optional) Do one of the following to configure RTP payload type:
• **rtp payload-type cisco-codec-isac number**
• **rtp payload-type cisco-codec-ilbc number**
• **rtp payload-type cisco-codec-video-h263+ number**
• **rtp payload-type cisco-codec-video-h264 number**
• **rtp payload-type opus number**

**Example:**
Device(config-dial-peer)# rtp payload-type opus

Configures the RTP payload type.

**Step 7** | end
**Example:**
Device(config-dial-peer)# end

Returns to privileged EXEC mode.

---

### 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[profile profile-tag]**
   • **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**
## 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><strong>enable</strong>&lt;br&gt;Example: <code>Device&gt; enable</code></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;Example: <code>Device&gt; configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>voice class codec tag</strong>&lt;br&gt;Example: <code>Device(config)# voice class codec 10</code></td>
<td>Enters voice-class configuration mode for the specified codec voice class.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Do the following for each audio codec you want to configure in the voice class:&lt;br&gt;• codec preference value codec-type [profile profile-tag ]&lt;br&gt;• codec preference value codec-type [bytes payload-size fixed-bytes ]&lt;br&gt;• codec preference value isac [mode {adaptive</td>
<td>independent}] [bit-rate value framesize { 30</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>exit</strong>&lt;br&gt;Example: <code>Device(config-class)# exit</code></td>
<td>Exits the current mode.&lt;br&gt;• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>dial-peer voice number voip</strong>&lt;br&gt;Example: <code>Device(config)# dial-peer voice 1 voip</code></td>
<td>Enters dial peer configuration mode for the specified VoIP dial peer.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>voice-class codec tag offer-all</strong>&lt;br&gt;Example: <code>Device(config-dial-peer)# voice-class codec 10</code></td>
<td>Applies the previously configured voice class and associated codecs to a dial peer.&lt;br&gt;• The <strong>offer-all</strong> keyword allows the device to offer all codecs configured in a codec voice class.</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>end</strong>&lt;br&gt;Example: <code>Device(config-dial-peer)# end</code></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
## 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>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 1 | enable                | Enables privileged EXEC mode.  
Example: `enable`  
Device> enable |  
• Enter your password if prompted. |
| Step 2 | `configure terminal`  | Enters global configuration mode.  
Example: `configure terminal`  
Device> configure terminal | |
| Step 3 | `voice class codec` *tag*  
Example: `voice class codec` 10 | Enters voice-class configuration mode for the specified codec voice class.  
Device(config)# voice class codec 10 | |
| Step 4 | `video codec` *codec*  
Example: `video codec` h261 | Configures a video codec within the voice class.  
video codec h261 | |
| Step 5 | `exit`  
Example: `exit`  
Device(config-class)# exit | Exits the current mode.  
• Enter your password if prompted. |
| Step 6 | `dial-peer voice` *number voip*  
Example: `dial-peer voice` 1 voip | Enters dial peer configuration mode for the specified VoIP dial peer.  
Device(config)# dial-peer voice 1 voip | |
| Step 7 | `voice-class codec` *tag offer-all*  
Example: `voice-class codec` 10 | Applies the previously configured codec voice class and associated codecs to a dial peer.  
• The offer-all keyword allows the device to offer all codecs configured in the codec voice class.  
Device(config-dial-peer)# voice-class codec 10 | |
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 Psipp 9.45.33.11:57210
 24 ORG T3 mp4a-latm VOIP P123 9.45.33.11:57210

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
 58 ANS T11 g711ulaw VOIP Psipp 2001:......:230A:6080
 59 ORG T11 g711ulaw VOIP P5000110011 10.13.37.150:6090
```

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

Example: Configuring a Codec Profile, Codec Preference List and Applying it to a Dial Peer for Opus Codec

router(config)# codec profile 79 opus
router(conf-codec-profile)# fmtp "fmtp:114 maxplaybackrate=16000; sprop-maxcapturerate=16000;
  maxaveragebitrate=20000; stereo=1; sprop-stereo=0; useinbandfec=0; usedtx=0"
router(conf-codec-profile)#exit

router(config)# voice class codec 80
router(config-class)# codec preference 1 opus profile 79
router(config-class)# exit

router(config)# dial-peer voice 604 voip
router(config-dial-peer)# rtp payload-type opus 126
router(config-dial-peer)# voice-class codec 80 offer-all
router(config-dial-peer)# exit
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 65
- Configuring CAC Based on Call Spike Detection, on page 67
- Configuring CAC Based on Maximum Calls per Destination, on page 68
- Bandwidth-Based Call Admission Control, on page 69

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>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:** | `Device# configure terminal` | |
| **Step 3** | `call threshold global [cpu-5sec | cpu-avg | io-mem | proc-mem | total-calls | total-mem] low low-threshold high high-threshold` | Configures the Call Admission Control feature based on the total calls, cpu, and memory usage at the interface level to reject SIP calls when the bandwidth that is required for the calls exceed the aggregate bandwidth threshold.  
**Note**  
By default, the system rejects incoming calls if the 5 second CPU utilization on the gateway exceeds 95%, and if the in-use process memory on the gateway exceeds 98%. |
| **Example:** | `Device(config)# call threshold global total-calls low 1 high 1`  
`or`  
`Device(config)# call threshold global cpu-avg low 75 high 85`  
`or`  
`Device(config)# call threshold global total-mem low 75 high 85` | |
| **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` | |
Example: Internal Error Code (IEC) for Default Call Rejection Based on CPU Utilization and Memory

Following is the sample Internal Error Code (IEC) that explains default call rejection based on CPU utilization and memory:

```
%VOICE_IEC-3-GW: C SCRIPTS: Internal Error (Low memory): IEC=1.1.181.11.4.0 on callID
1GUID=00000000000000000000000000000000
%IVR-3-LOW_MEMORYRESOURCE: IVR: System running low on memory (99/100 in use). Call (callID=1)
is rejected.

%VOICE_IEC-3-GW: C SCRIPTS: Internal Error (CPU high): IEC=1.1.181.11.3.0 on callID 2
%IVR-3-LOW_CPURESOURCE: IVR: System experiencing high cpu utilization (97/100). Call (callID=2) is rejected.

%VOICE_IEC-3-GW: CCAPI: Internal Error (Call spike threshold): IEC=1.1.181.1.29.0 on callID 3
%SIP-3-MEMCAC: Call rejected due to CAC based on Memory usage, sent response 503
```

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>
<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: 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: Device# configure terminal</td>
<td></td>
</tr>
</tbody>
</table>
### 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>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
<td>Device&gt;enable</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
<td>Device# configure terminal</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 3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>dial-peer voice</code></td>
<td>Enters dial peer voice configuration mode.</td>
<td></td>
</tr>
<tr>
<td><code>tag voip</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# dial-peer voice 10 voip</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>session protocol sipv2</code></td>
<td>Configures SIP as the session protocol type.</td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# session protocol sipv2</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><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>
<td></td>
</tr>
<tr>
<td><code>Device(config)# max-conn &lt;1-214748364&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>end</code></td>
<td>Exits global configuration mode and enters privileged EXEC mode.</td>
<td></td>
</tr>
<tr>
<td><code>Device# end</code></td>
<td></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.
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.
  - Filtering of codecs to accommodate calls within the available bandwidth.
  - Media flow-around, Session Description Protocol (SDP) pass-through, out-of-box low-density transcoding, high-density transcoding, video transcoding, and midcall consumption functionalities.
  - Non-SIP call legs.
  - Subinterfaces for bandwidth-based CAC on an interface.

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.

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

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 18: 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>G.726 (24 Kbps)</td>
<td>15</td>
<td>60</td>
<td>20</td>
<td>50</td>
<td>40</td>
<td>48</td>
</tr>
<tr>
<td>G.726 (16 Kbps)</td>
<td>10</td>
<td>40</td>
<td>20</td>
<td>50</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>G.728 (16 Kbps)</td>
<td>10</td>
<td>40</td>
<td>20</td>
<td>50</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>G722_64k (64 Kbps)</td>
<td>80</td>
<td>160</td>
<td>20</td>
<td>50</td>
<td>80</td>
<td>88</td>
</tr>
<tr>
<td>ilbc_mode_20 (15.2 Kbps)</td>
<td>38</td>
<td>38</td>
<td>20</td>
<td>50</td>
<td>31</td>
<td>39</td>
</tr>
<tr>
<td>ilbc_mode_30 (13.33 Kbps)</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>33.3</td>
<td>24</td>
<td>29</td>
</tr>
<tr>
<td>gsm (13 Kbps)</td>
<td>33</td>
<td>33</td>
<td>20</td>
<td>50</td>
<td>30</td>
<td>37</td>
</tr>
<tr>
<td>gsm (12 Kbps)</td>
<td>32</td>
<td>32</td>
<td>20</td>
<td>50</td>
<td>29</td>
<td>37</td>
</tr>
<tr>
<td>G.Clear (64 Kbps)</td>
<td>80</td>
<td>160</td>
<td>20</td>
<td>50</td>
<td>80</td>
<td>88</td>
</tr>
<tr>
<td>GSM AMR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>ISAC (32 Kbps)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>37</td>
<td>37</td>
</tr>
</tbody>
</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

---

#### Table 19: Fax Bandwidth Table

<table>
<thead>
<tr>
<th>T.38 Fax Bit Rate</th>
<th>Redundancy</th>
<th>Maximum Bandwidth in Kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>2400</td>
<td>None</td>
<td>8</td>
</tr>
<tr>
<td>2400</td>
<td>Redundancy</td>
<td>17</td>
</tr>
<tr>
<td>9600 (default)</td>
<td>None</td>
<td>16</td>
</tr>
<tr>
<td>9600 (default)</td>
<td>Redundancy</td>
<td>46</td>
</tr>
<tr>
<td>14400</td>
<td>None</td>
<td>20</td>
</tr>
<tr>
<td>14400</td>
<td>Redundancy</td>
<td>65</td>
</tr>
<tr>
<td>33600</td>
<td>None</td>
<td>40</td>
</tr>
<tr>
<td>33600</td>
<td>Redundancy</td>
<td>142</td>
</tr>
</tbody>
</table>
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><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> call threshold interface type number int-bandwidth {class-map name [l2-overhead percentage]</td>
<td>low low-threshold high high-threshold} [midcall-exceed]</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# call threshold interface GigabitEthernet 0/0 int-bandwidth low 1000 high 20000 midcall-exceed</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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]**
6. **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.  
  Example:  
  Device> enable |
| **Step 2** | `configure terminal` | Enters global configuration mode.  
  Example:  
  Device# configure terminal |
| **Step 3** | `dial-peer voice tag voip` | Enters dial peer voice configuration mode.  
  Example:  
  Device(config)# dial-peer voice 44 voip |
| **Step 4** | `session protocol sipv2` | Configures the Bandwidth-Based Call Admission Control feature for SIP dial peers only.  
  Example: |
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>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>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device# configure terminal</td>
<td>Enters privileged EXEC mode.</td>
</tr>
</tbody>
</table>

### Step 3

**voice service voip**

**Example:**

Device(config)# voice service voip

**Enters voice-service configuration mode.**

### Step 4

**sip**

**Example:**

Device(conf-voi-serv)# sip

**Enters service SIP configuration mode.**

### Step 5

**error-code-override cac-bandwidth failure**

**sip-status-code-number**

**Example:**

Device(conf-serv-sip)# error-code-override cac-bandwidth failure

**Confures bandwidth-based CAC SIP error response code mapping at the global level.**

### Step 6

**end**

**Example:**

Device(conf-serv-sip)# end

**Exits service SIP configuration mode and enters privileged EXEC mode.**

---

**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-bandwidth 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></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>enable</td>
<td></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>Enters global configuration mode.</td>
</tr>
<tr>
<td>configure terminal</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

#### Step 3
- **dial-peer voice** `tag {pots | voatm | vofr | voip}`
  - **Example:**
    ```text
    Device(config)# dial-peer voice 88 voip
    ```
  - **Purpose:** Enters dial peer voice configuration mode.

#### Step 4
- **voice-class sip error-code-override cac-bandwidth failure** `{sip-status-code-number | system}`
  - **Example:**
    ```text
    Device(config-dial-peer)# voice-class sip error-code-override cac-bandwidth failure 500
    ```
  - **Purpose:** Configures bandwidth-based CAC SIP error response code mapping at the dial peer level.

#### Step 5
- **end**
  - **Example:**
    ```text
    Device(config-dial-peer)# end
    ```
  - **Purpose:** Exits dial peer configuration mode and enters privileged EXEC mode.

### 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:**
    ```text
    Device>enable
    ```
  - Enables privileged EXEC mode.

**Step 2**
- **show call threshold config**
  - **Example:**
    ```text
    Device# show call threshold config
    Some resource polling interval:
    CPU_AVG interval: 60
    Memory interval: 5
    ```
GigabitEthernet0/0 int-bandwidth 0 100 400 N/A

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
   All resources are available for this check.
   Recorded time: 04:29:39 UTC Wed Dec 8 2010

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 ccsip 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(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(conf-voi-serv)# sip
Device(conf-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 20: Feature Information for Bandwidth-Based Call Admission Control

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Bandwidth-Based Call Admission Control     | 15.2(2)T               | 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 |
| Bandwidth-Based Call Admission Control     | Cisco IOS XE Release 3.7S | 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 |
Basic SIP Configuration

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](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

- Prerequisites for Basic SIP Configuration, on page 83
- Restrictions for Basic SIP Configuration, on page 83
- Information About Basic SIP Configuration, on page 84
- How to Perform Basic SIP Configuration, on page 85
- Configuration Examples for Basic SIP Configuration, on page 101
- Toll Fraud Prevention, on page 108

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 registrar on behalf of analog telephone voice ports (FXS), IP phone virtual voice ports (EFSX), 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 (EFSX), 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 21: 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 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**

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>Router&gt; enable</td>
</tr>
<tr>
<td></td>
<td>Example: Router# configure terminal</td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Router# configure terminal</td>
</tr>
<tr>
<td>Step 3 voice service voip</td>
<td>Enters voice-service VoIP configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Router(config)# voice service voip</td>
</tr>
<tr>
<td>Step 4 [no] shutdown [forced]</td>
<td>Shuts down or enables VoIP call services.</td>
</tr>
<tr>
<td></td>
<td>Example: Router(config-voi-serv)# shutdown forced</td>
</tr>
<tr>
<td>Step 5 exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Router(config-voi-serv)# exit</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>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<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></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router&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></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>voice service voip</code></td>
<td>Enters voice-service VoIP configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# voice service voip</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>sip</code></td>
<td>Enters SIP configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config-voi-serv)# sip</code></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>[no] call service stop [forced] [maintain-registration]</code></td>
<td>Shuts down or enables VoIP call services for the selected submode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(conf-serv-sip)# call service stop maintain-registration</code></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>exit</code></td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(conf-serv-sip)# exit</code></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`
### Configuring SIP Register Support

#### 6. **timers register** milliseconds

#### 7. **exit**

---

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> en<strong>able</strong></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> sip-ua</td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> registrar {dns: address</td>
<td>ipv4: destination-address} expires seconds [tcp] [secondary]</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-sip-ua)# registrar ipv4:10.8.17.40 expires 3600 secondary</td>
<td></td>
</tr>
<tr>
<td>• dns: address --Domain-name server that resolves the name of the dial peer to receive calls.</td>
<td></td>
</tr>
<tr>
<td>• ipv4: destination-address --IP address of the dial peer to receive calls.</td>
<td></td>
</tr>
<tr>
<td>• expires seconds --Default registration time, in seconds.</td>
<td></td>
</tr>
<tr>
<td>• tcp --Sets transport layer protocol to TCP. UDP is the default.</td>
<td></td>
</tr>
<tr>
<td>• secondary --Specifies registration with a secondary SIP proxy or registrar for redundancy purposes. Optional.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> retry register <strong>number</strong></td>
<td>Use this command to set the total number of SIP Register messages that the gateway should send. The argument is as follows:</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-sip-ua)# retry register 6</td>
<td></td>
</tr>
<tr>
<td>• number --Number of Register message retries. Range: 1 to 10. Default: 6.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> timers register <strong>milliseconds</strong></td>
<td>Use this command to set how long the SIP user agent waits before sending register requests. The argument is as follows:</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
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> 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> sip-ua</td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# sip-ua</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.

#### 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> 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>
</tbody>
</table>

**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.
### Purpose

#### Command or Action

**Step 3**

- **voice service voip**
- **Example:**
  
  Router(config)# voice service voip

**Step 4**

- **redirect ip2ip**
- **Example:**
  
  Router(conf-voi-serv)# redirect ip2ip

**Step 5**

- **exit**
- **Example:**
  
  Router(conf-voi-serv)# exit

### 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

#### Command or Action

**Step 1**

- **enable**
- **Example:**
  
  Router> enable

**Step 2**

- **configure terminal**
- **Example:**
  
  Router# configure terminal

- **Purpose:**
  
  Enters privileged EXEC mode or any other security level set by a system administrator. Enter your password if prompted.

- **Purpose:**
  
  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>Use this command to enter dial-peer configuration mode. The argument is as follows:</td>
</tr>
<tr>
<td><strong>dial-peer voice tag voip</strong></td>
<td>• <em>tag</em> --Digits that define a particular dial peer. Range: 1 to 2,147,483,647 (enter without commas).</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# dial-peer voice 29 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enables a specific application on a dial peer. The argument is as follows:</td>
</tr>
<tr>
<td><strong>application application-name</strong></td>
<td>• <em>application-name</em> --Name of the predefined application you wish to enable on the dial peer. For SIP, the default Tcl 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</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config-dial-peer)# application session</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Redirects SIP phone calls to SIP phone calls on a specific VoIP dial peer using the Cisco IOS voice gateway.</td>
</tr>
<tr>
<td><strong>redirect ip2ip</strong></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-dial-peer)# redirect ip2ip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td><strong>exit</strong></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>

**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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></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>enable</code></td>
<td>Router&gt; enable</td>
</tr>
<tr>
<td><strong>Example:</strong></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>Router# configure terminal</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<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>Router(config)# voice service voip</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters SIP configuration mode.</td>
</tr>
<tr>
<td><code>sip</code></td>
<td>Router(config-voi-serv)# sip</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Sets the order of contacts in the 300 Multiple Choice Message. Keywords are as follows:</td>
</tr>
</tbody>
</table>
| `redirect contact order [best-match | longest-match]` | - **best-match** --Use the current system configuration to set the order of contacts.  
- **longest-match** --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. |
| **Example:** | Router(conf-serv-sip)# redirect contact order best-match |
| **Step 6** | Exits the current mode. |
| `exit` | Router(conf-serv-sip)# exit |
| **Example:** | |

**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 13: 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 14: 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.
dial-peer voice 53001 pots
    preference 2
    destination-pattern 5300001
    prefix 5300001
!
dial-peer voice 53002 pots
    preference 2
    destination-pattern 5300002
    prefix 5300002
!
dial-peer voice 530011 voip
    preference 1
    destination-pattern 5300001
    session protocol sipv2
    session target ipv4:10.1.1.41
    playout-delay maximum 300
    codec g711alaw
!
dial-peer voice 530022 voip
    preference 1
    destination-pattern 5300002
    session protocol sipv2
    session target ipv4:10.1.1.41
    playout-delay maximum 300
    codec g711alaw

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:
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,
    OkPreack 0/0, OkPreconditionMet 0/0,
    OkSubscribe 0/0, OkNOTIFY 0/0,
    OkInfo 0/0, 202Accepted 0/0
Redirect (Inbound only except for MovedTemp(Inbound/Outbound)):

- MultipleChoice 0
- MovedPermanently 0
- MovedTemporarily 0/0
- UseProxy 0
- AlternateService 0

Client Error:
- BadRequest 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
- NullableTooLarge 0/0
- ReqURITooLarge 0/0
- UnsupportedMediaType 0/0
- BadExtension 0/0
- UseProxy 0
- AlternateService 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
- Cancel 0/0
- Options 0/0
- Prack 0/0
- Comet 0/0
- Subscribe 0/0
- NOTIFY 0/0
- Register 49/16

Retry Statistics:
- Invite 0
- Ack 0
- Cancel 0
- Response 0
- Prack 0
- Comet 0
- Reliable1xx 0
- NOTIFY 0

Example:

Router# show sip ua statistics
SIP Response Statistics (Inbound/Outbound):
- Informational:
  - Trying 0/0
  - Forwarded 0/0
  - Queued 0/0
  - SessionProgress 0/0
- Success:
  - OKInvite 0/0
  - OKAck 0/0
  - OKCancel 0/0
  - OKOptions 0/0
  - OKPrack 0/0
  - OKPreconditionMet 0/0
  - OKSubscribe 0/0
  - OKNotify 0/0
  - 202Accepted 0/0

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.
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 (millisecs)
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 commands 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
```
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
!
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:FOO5B302.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 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
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
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
!
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
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 1719
!
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 111
- Information About SIP Binding, on page 112
- Configuring SIP Binding, on page 118
- Verifying SIP Binding, on page 120

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 22: Feature Information for SIP Binding

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Gateway Support for the bind Command</td>
<td>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</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 IPAddress 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>
<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. 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. Only one SIP listen socket with global bind address.</td>
</tr>
</tbody>
</table>
The **bind** command performs different functions based on the state of the interface (see the table below).

**Table 24: State of the Interface for the bind Command**

<table>
<thead>
<tr>
<th>Interface State</th>
<th>Result Using Bind Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down</td>
<td>TCP, TLS, and User Datagram Protocol (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></td>
</tr>
<tr>
<td>No 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 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>No active calls</td>
<td></td>
</tr>
<tr>
<td>No shut down</td>
<td>TCP, TLS, and UDP socket listeners are initially closed. 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>Active calls</td>
<td></td>
</tr>
<tr>
<td>Interface State</td>
<td>Result Using Bind Command</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
</tr>
<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 <code>no shutdown</code> 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 can be modified when the dial-peer do not have any active calls.
- 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 25: Interaction Between Previously Set and New bind Commands

<table>
<thead>
<tr>
<th>Interface State</th>
<th>bind Command</th>
<th>Result Using bind Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without active calls</td>
<td>bind all</td>
<td>Generated bind control and bind media commands to override existing bind control and bind media commands.</td>
</tr>
<tr>
<td></td>
<td>bind control</td>
<td>Overrides existing bind control command.</td>
</tr>
<tr>
<td></td>
<td>bind media</td>
<td>Overrides existing bind media command.</td>
</tr>
<tr>
<td>With active calls</td>
<td>bind all or bind control bind media</td>
<td>Global Configuration: Blocks the command, and the following error message appears: • Error: You cannot change the interface binding for a dial-peer that is processing live traffic.</td>
</tr>
<tr>
<td></td>
<td>bind all or bind control or bind media</td>
<td>Dial-peer Configuration: You cannot apply bind or no bind command to a dial-peer that is processing active calls. Blocks the command, and the following error message appears: • Error: You cannot change the interface binding for a dial-peer that is processing live traffic.</td>
</tr>
</tbody>
</table>

Consider the following scenarios for attaching a tenant to a dial-peer that is processing active calls:

- You can attach a tenant to a dial-peer, when the the dial-peer has bind (bind control or bind all) command enabled.

- You cannot attach a tenant to a dial-peer, when the dial-peer has no bind or bind media command enabled and the tenant has bind control or bind all command enabled.

Consider the following scenarios for changing bind configuration on a tenant, when the tenant is attached to a dial-peer that is processing active calls:

- You can change the bind configuration on tenant, when the associated dial-peer has bind (bind control or bind all) command enabled. Because, the dial-peer bind configuration takes precedence over the tenant bind configuration.
You cannot change the bind configuration on tenant, when the associated dial-peer has `no bind` or `bind media` command enabled and the tenant has `bind control` or `bind all` command enabled.

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

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>bind all</code> command applies to global and dial peer. The table</td>
</tr>
<tr>
<td>below applies to <code>bind media</code> only if the media interface is the</td>
</tr>
<tr>
<td>same as the <code>bind control</code> interface. If the two interfaces are</td>
</tr>
<tr>
<td>different, media behavior is independent of the interface state.</td>
</tr>
</tbody>
</table>

**Table 26: 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 socke)</td>
</tr>
<tr>
<td>With or without active calls</td>
<td>Then the sockets are opened to listen to any IP address.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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. (Socket listeners receive datagrams addressed to the socke)</td>
</tr>
<tr>
<td>Without active calls</td>
<td>Then the sockets are opened and bound to the IP address set by the <code>bind</code> command.</td>
</tr>
<tr>
<td></td>
<td>The sockets accept packets destined for the bound address only.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>With active calls</td>
<td>Then the sockets are opened to listen to any IP address.</td>
</tr>
<tr>
<td></td>
<td>The dial peer bind socket listeners of the interface are reopened 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.

| Interface State                                                                 | Result Using bind all or bind control Commands                                                                                                                                                                                                 |
|--------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Bound interface’s IP address is removed.                                       | 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 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.                                                                                          |
| The physical cable is pulled on the bound port, or the interface layer goes down. | 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.                                                                                          |
| A bind interface is shut down, or its IP address is changed, or the physical cable is pulled while SIP calls are active. | 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. |
**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>Router&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>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>interface type number</td>
<td>Configures an interface type and enters the interface configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>- type number — Type of interface to be configured and the port, connector, or interface card number.</td>
</tr>
<tr>
<td></td>
<td>Router(config)# interface fastethernet0/0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ip address ip-addressmask [secondary]</td>
<td>Configures a primary or secondary IP address for an interface.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>Note: Secondary IP address on an interface with SIP binding is not supported for CUBE.</td>
</tr>
<tr>
<td></td>
<td>Router(config-if)# ip address 192.168.200.33 255.255.255.0</td>
<td></td>
</tr>
<tr>
<td>5</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(config-if)# exit</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Use one of the following commands to configure SIP binding:</td>
<td>Sets a source interface for signaling and media packets. The binding applies to the specified interfaces only. SIP must be configured globally or at a dial peer level.</td>
</tr>
<tr>
<td></td>
<td>• bind {control</td>
<td>all} source-interface interface-id [ipv6-address ipv6-address] in SIP configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• bind media {source-address ipv4 ipv4-address</td>
<td>source-interface interface-id [ipv6-address ipv6-address]} in SIP configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• all</td>
<td>Binds signaling and media packets.</td>
</tr>
<tr>
<td></td>
<td>voice-class sip bind media {source-address ipv4 ipv4-address</td>
<td>source-interface interface-id [ipv6-address ipv6-address]} in dial-peer configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• ipv4 ipv4-address — Configures the IPv4 address.</td>
<td></td>
</tr>
</tbody>
</table>
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:
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)
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 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: IP4 IP6
Transport types supported: RTP/AVP udp tl

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 req 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: ''
DNS-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 media rsvp-fail DSCP = ef, ip signaling DSCP = af31,
ip video rsvp-none DSCP = ef, ip video rsvp-pass DSCP = af41
ip video rsvp-fail 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 = best-effort, acc-qos = best-effort,
req-qos video = best-effort, acc-qos video = best-effort,
req-qos audio def bandwidth = 64, req-qos audio max bandwidth = 0,
req-qos video def bandwidth = 384, req-qos video max bandwidth = 0,
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-lp=114, iSAC=124
   lmr_tone=0, nte_tone=0
   h263=118, h264=119
   G726r16 using static payload
   G726r24 using static payload
RTP comfort noise payload type = 75
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 relyxx = 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,
Verifying SIP Binding

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 auto
  speed auto
  fair-queue 64 256 1000
  ip rsvp 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
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 127
- Media Flow-Through, on page 128
- Media Flow-Around, on page 130
- Media Anti-Trombone, on page 131

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 27: 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 15: Media Flow-Through Mode

Restrictions for Media Flow-Through

- Media flow-through for Delayed-Offer to Early-Offer audio and video calls is not supported.

Note

Cisco UBE supports Media-Flow Through video. However, Cisco UBE does not know the video SDP parameters that the various video endpoints support. Cisco UBE supports basic H264 SDP for Media-Flow Through video.

Cisco UBE supports the following video codecs:

- H261
- H263
- H263+
- H264
- MPEG-4

Note

Cisco UBE supports the feature SDP pass-thru in Media-Flow Through. This feature allows Cisco UBE to support a video SDP parameter. The following example explains SDP pass-thru configuration.

Router#conf t
Router(config)#voice service voip
Router(config-voi-serv)sip
Router(config-voi-serv)pass-thru content sdp
Configure 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> 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>
</tbody>
</table>

| **Step 2** configure terminal | Enter global configuration mode. |
| **Example:** |         |
| Device# configure terminal |         |

| **Step 3** Use one of the following commands to configure media flow-through: | Enables media packets to pass through the endpoints, without the intervention of the CUBE. |
| **Example:** |         |
| In dial-peer configuration mode |         |
| ! Applying SIP profiles to one dial peer only |         |
| Device (config) dial-peer voice 10 voip |         |
| Device (config-dial-peer) media flow-through |         |
| Device (config-dial-peer) end |         |
| **Example:** |         |
| In global VoIP mode |         |
| ! Applying SIP profiles globally |         |
| Device(config)# voice service voip |         |
| Device(config-voi-serv)#media flow-through |         |
| Device(config-voi-serv)#end |         |

| **Step 4** end | Exits to privileged EXEC mode. |
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.

Figure 16: Media Flow-Around

Configure 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
   • media flow-around 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>Step 1 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>Step 2 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>Step 3 Use one of the following commands to configure media flow-around:</td>
<td>Enables media packets to pass directly between the endpoints, without the intervention of the CUBE. The media packet is to flow around the gateway.</td>
</tr>
<tr>
<td>• media flow-around in dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>• media flow-around in global VoIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
### 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 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 17: Tromboning - Needlessly looping of Media Packets*

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

*Figure 18: 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>
</table>
| Step 1
   * enable
   * Example:
     Device> enable                                       | Enables privileged EXEC mode. Enter your password if prompted.         |
### Configuring Media Anti-Tromboning

<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><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
</tbody>
</table>

| **Step 3**        | Enables media anti-trombone for all calls. |
| 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 |
| **Example:**      | |
| In dial-peer configuration mode |
| ! 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` |

| In global VoIP SIP mode |
| ! Applying SIP profiles globally |
| `Device(config)#voice service voip` |
| `Device(config-voi-serv)#media anti-trombone` |
| `Device(config-voi-serv)#end` |

| **Step 4**        | Exits to privileged EXEC mode. |
| `end`             | |

---

**Example:**

```plaintext
! 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

Example:

! Applying SIP profiles globally
Device(config)#voice service voip
Device(config-voi-serv)#media anti-trombone
Device(config-voi-serv)#end
```
Configuring Media Anti-Tromboning
CHAPTER 13

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.

Figure 19: Incoming and Outgoing Messages Where SIP Profiles Can Be Applied

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 135
- Information About SIP Profiles, on page 136
- Restrictions for SIP Profiles, on page 139
- How to Configure SIP Profiles, on page 139

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 28: 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</td>
<td>This feature extends support to inbound messages.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.12S</td>
<td>This feature modifies the following commands:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</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.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS 12.4(20)T</td>
<td>This feature introduces the following commands: <strong>voice class sip-profiles, response, request</strong>.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 2.5</td>
<td></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>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</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>
<tr>
<td></td>
<td>Cisco IOS XE 3.17S</td>
<td></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 20: 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

The following are a few important notes for SIP Profiles:
Important Characteristics of 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 that are 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 must press "Ctrl+v" keys and then type ‘?’. This operation is needed to treat ‘?’ as an input character itself instead of the usual device help prompt.

Note: Regex features like look-ahead, look-behind, OR operator, and non capturing group are not supported (for example, ?!, ?, ?, <=, |, and so on).

- 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 occur, a backslash 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 the same header, the header values are stored as a comma-separated list. Consider this fact while modifying it.
- Modification by an inbound SIP profile takes place before regular SIP call processing happens so that behavior of CUBE is 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. This change might change the inbound dial-peer that is matched when an 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 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 udpt 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.

• If a SIP profile is applied to modify the SDP content of a SIP message, CUBE does not increment the "o=" line version, which may cause ITSPs to disconnect the call. CUBE does not store the modified SDP after the application of the SIP profile.

Note that manipulation of SIP messages by outbound SIP profiles occurs as the final step before the outgoing message leaves the CUBE device, and occurs after destination dial-peer matching has taken place. Changes to SIP messages are not remembered or acted on by CUBE. The Content-length field is recalculated after SIP Profile rules are applied to outgoing messages.

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

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 Profiles and enters voice class configuration mode. |
| Example: Device(config)# voice class sip-profiles 10 | |
| Step 4 Enter one of the following to add, remove, modify SIP headers: | 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. |
### 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
     
     ```
     pass-thru content custom-sdp
     ```
   - In dial-peer configuration mode (The configuration is applied on the incoming dial-peer)
     
     ```
     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 sdp mline-index index COPY match-pattern copy-variable`
   - `request/response ANY sdp-header mline-index index header-name ADD copy-variable`
   - `request/response ANY sdp-header mline-index index header-name MODIFY copy-variable + replace-pattern`
   - `request/response ANY sdp-header mline-index index header-name REMOVE`

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></td>
</tr>
<tr>
<td><code>enable</code></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></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# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<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>
</tr>
<tr>
<td><code>pass-thru content custom-sdp</code></td>
<td></td>
</tr>
<tr>
<td>• In dial-peer configuration mode (The configuration is applied on the incoming dial-peer)</td>
<td></td>
</tr>
<tr>
<td><code>voice-class sip pass-thru content custom-sdp</code></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>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td>Device(config-serv-sip)# pass-thru content custom-sdp</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 2 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip pass-thru content custom-sdp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>voice class sip-profiles profile-id</code></td>
<td>Voice class sip-profile is configured on the outbound dial-peer or as a global configuration.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Creates a SIP Profile and enters voice class configuration mode.</td>
</tr>
<tr>
<td>Device(config)# voice class sip-profiles 10</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>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:</td>
</tr>
<tr>
<td></td>
<td>• request/response ANY peer-header sdp mline-index index COPY match-pattern copy-variable</td>
</tr>
<tr>
<td></td>
<td>• request/response ANY sdp-header mline-index indexheader-name ADD copy-variable</td>
</tr>
<tr>
<td></td>
<td>• request/response ANY sdp-header mline-index indexheader-name MODIFY copy-variable + replace-pattern</td>
</tr>
<tr>
<td></td>
<td>• request/response ANY sdp-header mline-index indexheader-name REMOVE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>end</td>
</tr>
</tbody>
</table>

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"
```

Example: Configuring SIP Profile Rules (Parameter Passing)

```
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

```
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:
   - `rule before tag request method sdp-header | sip-header header-name add | copy | modify | remove string`
   - `rule before tag response method sdp-header | sip-header header-name add | copy | modify | remove string`
6. Enter the following to delete a rule:
   - `no rule tag`
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></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-profiles profile-id</td>
<td>Creates a SIP Profile and enters voice class configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice class sip-profiles 10</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>According to your choice, this step tags the SIP request or response header with a unique number.</td>
</tr>
<tr>
<td>Enter one of the following to add, copy, modify, or remove a SIP request or response headers to a SIP profile configuration:</td>
<td></td>
</tr>
<tr>
<td>• `rule tag request method sd-p-header</td>
<td>sip-header header-name add</td>
</tr>
<tr>
<td>• `rule tag response method sd-p-header</td>
<td>sip-header header-name add</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>According to your choice this steps inserts the rule at the position specified in the <code>before tag</code>. The subsequent rules in the existing SIP profile configuration is incremented sequentially.</td>
</tr>
<tr>
<td>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:</td>
<td></td>
</tr>
<tr>
<td>• `rule before tag request method sd-p-header</td>
<td>sip-header header-name add</td>
</tr>
<tr>
<td>• `rule before tag response method sd-p-header</td>
<td>sip-header header-name add</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>According to your choice, this step tags the SIP request or response with a unique number.</td>
</tr>
<tr>
<td>Enter the following to delete a rule:</td>
<td></td>
</tr>
<tr>
<td>• <code>no rule tag</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Exits voice class sip-profiles configuration mode.</td>
</tr>
</tbody>
</table>

## 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-remove remove`
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-add add non-standard-header-value-to-add`
   - `response message [method method-type] [sip-header] {non-standard-header-to-remove remove`
### 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-profiles profile-id</td>
<td>Creates a SIP Profiles and enters voice class configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></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>According to your choice, this step does one of the following:</td>
</tr>
<tr>
<td>Enter one of the following to add, copy, remove, or modify non-standard SIP request headers:</td>
<td></td>
</tr>
<tr>
<td>• <strong>request message</strong> {sip-header}</td>
<td>• Adds a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td>non-standard-header-to-add add</td>
<td>• Copies contents from a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td>non-standard-header-value-to-add</td>
<td>• Removes contents from a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td>• <strong>request message</strong> {sip-header}</td>
<td>• Modifies a non-standard SIP header to a SIP request.</td>
</tr>
<tr>
<td>non-standard-header-to-copy copy</td>
<td>• If <code>ANY</code> is used in place of a header, it indicates that a rule must</td>
</tr>
<tr>
<td>non-standard-header-value-to-match copy-variable</td>
<td>be applied to any message within the specified category.</td>
</tr>
<tr>
<td>• <strong>request message</strong> {sip-header}</td>
<td>• For <code>non-standard-header-value-to-add</code> used to add a non-standard</td>
</tr>
<tr>
<td>non-standard-header-to-remove remove</td>
<td>header, <code>non-standard-header-value-to-match</code> or <code>non-standard-header-value-to-replace</code> used to modify a non-standard header:</td>
</tr>
<tr>
<td>• <strong>request message</strong> {sip-header}</td>
<td>• If a whitespace occurs, the entire value must be included between</td>
</tr>
<tr>
<td>non-standard-header-to-modify modify</td>
<td>double quotes. For example, “User-Agent: CISCO CUBE”</td>
</tr>
<tr>
<td>non-standard-header-value-to-match</td>
<td>• If double quotes occurs, a back slash must prefix the double quotes.</td>
</tr>
<tr>
<td>non-standard-header-value-to-replace</td>
<td>For example, “User-Agent: &quot;CISCO&quot; CUBE”</td>
</tr>
<tr>
<td>• Regular expressions are supported.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>According to your choice, this step does one of the following:</td>
</tr>
<tr>
<td>Enter one of the following to add, copy, remove, or modify non-standard SIP response headers:</td>
<td></td>
</tr>
<tr>
<td>• <strong>response message</strong> [method method-type] {sip-header}</td>
<td>• Adds a non-standard SIP to a SIP response.</td>
</tr>
<tr>
<td>non-standard-header-to-add add</td>
<td>• Copies contents from a non-standard SIP header to a SIP response.</td>
</tr>
<tr>
<td>non-standard-header-value-to-add</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>• response message [method method-type] {sip-header} non-standard-header-to-remove remove</td>
<td>• Modifies a non-standard SIP 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>• 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>
<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> Enter the following to upgrade SIP profiles configurations to rule-format:</td>
<td>Upgrades all SIP Profiles to rule-format configurations.</td>
</tr>
<tr>
<td>• voice sip sip-profiles upgrade</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> In EXEC(#) mode:</td>
<td></td>
</tr>
</tbody>
</table>
Purpose

Command or Action

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

Example:

In EXEC(#) mode:

Device# voice sip sip-profiles downgrade

Step 4

end

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.
   • sip-profiles profile-id in the global VoIP configuration mode
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></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 Apply the SIP profile to a dial peer:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• voice-class sip profiles profile-id in the dial-peer configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• sip-profiles profile-id in the global VoIP configuration mode</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In dial-peer configuration mode</td>
</tr>
<tr>
<td></td>
<td>!Applying SIP profiles to one dial peer only</td>
</tr>
</tbody>
</table>
### 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>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>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>2</td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>3</td>
<td><code>voice service voip</code></td>
<td>Enters global VoIP configuration mode.</td>
</tr>
</tbody>
</table>

#### Example:

```
Device(config)# voice service voip
Device(config-voi-serv)# sip
Device(config-voi-sip)# sip-profiles 20
Device(config-voi-sip)# end
```
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

dbg csip 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: //-1/735085DC8F3D/SIP/Info/sipSPIGetShrlPeer:
Try match incoming dialpeer for Calling number:
  : sipOct 12 06:51:53.619:
  //1-735085DC8F3D/SIP/Info/sipSPIGetCallConfig:
Peer tag 2 matched for incoming call
Oct 12 06:51:53.619: //1/xxxxxxxxxxxx/SIP/Info/sipSPIGetCallConfig:
  voice class SIP profiles tag is set : 1
Oct 12 06:51:53.619: //1-735085DC8F3D/SIP/Info/sipSPIGetCallConfig:
  Not using Voice Class Codec
Oct 12 06:51:53.619: //1-735085DC8F3D/SIP/Info/sipSPIGetCallConfig:
  xcoder high-density disabled
Oct 12 06:51:53.619: //1-735085DC8F3D/SIP/Info/sipSPIGetCallConfig:
  Flow Mode set to FLOW_THROUGH

This command also displays the modifications performed by the SIP profile configuration, by preceding the modification information with the word sip_profiles, as highlighted in the example below.

Example:

Device# debug ccsip all

Oct 12 06:51:53.647: //1/xxxxxxxxxxxx/SIP/Info/
  sip_profiles_application_change_sdp_line:
  New SDP header is added : b=AS: 1600
Oct 12 06:51:53.647: //1/xxxxxxxxxxxx/SIP/Info/
  sip_profiles_update_content_length:
  Content length header before modification :
    Content-Length: 290
Oct 12 06:51:53.647: //1/xxxxxxxxxxxx/SIP/Info/
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"
Device(config-class)# end

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"
Device(config-class)# end

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

Device(config)# voice class sip-profiles 20
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 "(<.*:)(.*@)") \1gateway@"
```

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 "(<.*:)(.*@)") \1gateway@"
```

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)# request any sdp-header Audio-Attribute modify \texttt{"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 \texttt{a=inactive} with \texttt{a=inactive}, \texttt{a=recvonly}, or \texttt{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 \texttt{Sendrecv} prevents the provider from replaying back with \texttt{a=inactive}, \texttt{a=recvonly}, or \texttt{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
\texttt{a=sendrecv}

Case 3
Inbound Audio-Attribute
\texttt{a=sendonly}
Outbound Audio-Attribute
\texttt{a=sendrecv}

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

Device(config-class)# response any sdp-header Audio-Attribute modify \texttt{\"a=inactive\" \"a=sendrecv\"}
Device(config-class)# response any sdp-header Audio-Attribute modify \texttt{\"a=recvonly\" \"a=sendrecv\"}
Device(config-class)# response any sdp-header Audio-Attribute modify \texttt{\"a=sendonly\" \"a=sendrecv\"}
Example: Modifying Packetization Mode in a=fmt 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=fmt 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-voi-serv) sip-profiles 20
Device(config-voi-serv) sip-profiles 10
Device(config-voi-serv) sip-profiles 40
Device(config-voi-serv) sip-profiles 103
Device(config-voi-serv) sip-profiles 104
Device(config-voi-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 sip-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-class)# 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

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

Device(config)# voice class sip-profiles 1
Device(config-class)# rule 1 request INVITE sip-header Contact Modify "(.*)" ";temp=xyz"
Device(config-class)# rule 2 request INVITE sip-header Supported Add "Supported: "
Device(config-class)# rule before 2 request INVITE sip-header To Modify "(.*)" ";temp=abc"

The SIP Profile after inserting the new rule

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

The following is a snippet from show running-config command showing the SIP profiles after upgrading to 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: 
!

**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:

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

**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:

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

The SIP INVITE message before the SIP profile has been applied is show 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:11100001089.13.40.249>;tag=F11AE0-1D8D
To: <sip:222200002089.13.40.250>
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
```

v=0
o=CiscoSystemsSIP-GW-UserAgent 6906 8069 IN IP4 9.13.40.249
s=SIP Call
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.

```plaintext
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
```

Example: Copy a User-to-User from REFER Message

Copy a User-to-User from a REFER message

Apply the sip profile 1210 to the inbound dial-peer that receives the REFER. The sip profile 1211 need to be applied to the outbound dial-peer that is matched after the REFER is consumed.

```plaintext
voice class sip-profiles 1210
request REFER sip-header Refer-To copy "Refer-To:.User-to-User=(.*)">" u03
request REFER sip-header x-user add "x-user: TEST2"
request REFER sip-header x-user modify "x-user: (.*)" "x-user: \u03"
```
The variable does not get copied properly between the call legs. Non-standard header to store the value and then modify it on the out leg. An ADD operation puts new header under the content-length header.

To store, remove and re-add the header:

```plaintext
voice class sip-profiles 1211
request INVITE sip-header x-user copy "x-user: (.)" u05
request INVITE sip-header User-to-User add "User-to-User: TEST1"
request INVITE sip-header User-to-User modify "User-to-User: (.)" "User-to-User: \u05"
request INVITE sip-header x-user remove
request INVITE sip-header Content-length copy "Content-Length: (.)" u06
request INVITE sip-header Content-length remove
request INVITE sip-header Content-length add "Content-Length: Test3"
request INVITE sip-header Content-length modify "Content-Length: (.)" "Content-Length: \u06"
```
Example: Copy a User-to-User from REFER Message
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-Dial (OOD) OPTIONS ping setup.

- Information About SIP Out-Of-dialog OPTIONS Ping Group, on page 163
- How to Configure SIP Out-Of-dialog OPTIONS Ping Group, on page 164
- Configuration Examples For SIP Out-of-Dial OPTIONS Ping Group, on page 166
- Additional References, on page 168
- Feature Information for SIP Out-of-dialog OPTIONS Ping Group, on page 169

Information About SIP Out-Of-dialog OPTIONS Ping Group

SIP Out-of-Dialog OPTIONS Ping Group Overview

The SIP Out-Of-Dial 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.

Note

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 except the first configured one. But the dial-peer is always marked as ACTIVE. Similarly, it is also not supported in multi VRF setup.

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>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.</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>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td>Configures a keepalive profile and enters voice class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong> voice class sip-options-keepalive keepalive-group-profile-id</td>
<td>You can use the shutdown command to suspend keepalive activity for all dial peers associated with the keepalive profile.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# voice class sip-options-keepalive 171</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> description text</td>
<td>Configures a textual description for the keepalive heartbeat connection.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# description Target Boston</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> transport {tcp [tls]</td>
<td>udp</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# transport tcp</td>
<td>The default value is system.</td>
</tr>
<tr>
<td><strong>Step 6</strong> sip-profiles profile-number</td>
<td>Specifies the SIP profile that is to be used to send this message.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# sip-profiles 100</td>
<td>To configure a SIP profile, refer to “Configuring SIP Parameter Modification”.</td>
</tr>
<tr>
<td><strong>Step 7</strong> 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><strong>Example:</strong> Device(config-class)# down-interval 35</td>
<td>The default value is 30.</td>
</tr>
<tr>
<td><strong>Step 8</strong> 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><strong>Example:</strong> Device(config-class)# up-interval 65</td>
<td>The default value is 60.</td>
</tr>
<tr>
<td><strong>Step 9</strong> 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 heartbeat connection status should be switched from Up to Down.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# retry 30</td>
<td>The default value is 5.</td>
</tr>
<tr>
<td><strong>Step 10</strong> exit</td>
<td>Exits voice class configuration mode and 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(config-class)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>Defines a local dial peer and enters dial peer configuration mode.</td>
</tr>
<tr>
<td><strong>dial-peer voice dial-peer-id voip</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 123 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong></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><strong>session protocol sipv2</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 13</strong></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><strong>voice-class sip options-keepalive profile keepalive-group-profile-id</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip options-keepalive profile 171</td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong></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><strong>session server-group server-group-id</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session server-group 151</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td>Exits dial peer configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>end</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td>Displays information about voice class server group.</td>
</tr>
<tr>
<td><strong>show voice class sip-options-keepalive keepalive-group-profile-id</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<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**

```plaintext
!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 tcp
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
Peer Tag Server Group OOD SessID OOD Stat IfIndex
-------- ------------ ---------- -------- -------
123

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.
Device(config)# dial-peer voice 181 voip
!Associate a destination pattern map
Device(config-dial-peer)# destination e164-pattern-map 3000
Device(config-dial-peer)# session protocol sipv2
!Associate a server group with the dial peer
Device(config-dial-peer)# session server-group 151
!Associate 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
<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>106</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>151</td>
<td>106</td>
<td>Busy</td>
<td></td>
</tr>
</tbody>
</table>

Server Group: 151
OOD SessID: 1
Target: ipv4:10.1.1.1
Transport: system
Sip Profiles: 100

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

OOD SessID: 3
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 Command List, All Releases</td>
</tr>
<tr>
<td>SIP Configuration Guide</td>
<td></td>
</tr>
<tr>
<td>Configuring SIP profiles</td>
<td></td>
</tr>
<tr>
<td>Configuring server groups</td>
<td></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. 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. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></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>
<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: voice class sip-options-keepalive, description, transport, sip-profiles, down-interval, up-interval, voice-class sip options-keepalive profile, retry, show voice class sip options-keepalive.</td>
</tr>
</tbody>
</table>
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:

To identify the hardware platform or software image information associated with a feature in this chapter, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release.

- Tcl IVR Overview, on page 171
- Tcl IVR Enhancements, on page 172
- RTSP Client Implementation, on page 172
- TCL IVR Prompts Played on IP Call Legs, on page 173
- TCL Verbs, on page 174
- TCL IVR Prerequisite Tasks, on page 177
- TCL IVR Configuration Tasks List, on page 177
- Configuring the Call Application for the Dial Peer, on page 178
- Configuring TCL IVR on the Inbound POTS Dial Peer, on page 180
- Configuring TCL IVR on the Inbound VoIP Dial Peer, on page 182
- Verifying TCL IVR Configuration, on page 184
- TCL IVR Configuration Examples, on page 185
- TCL IVR for Gateway1 (GW1) Configuration Example, on page 185
- TCL IVR for GW2 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 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:

- Real Time Streaming Protocol (RTSP) client implementation
- TCL IVR prompt playout and digit collection on IP call legs
- New TCL verbs to utilize RTSP 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.

### 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 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) can not be on the IP call leg so the script cannot initiate a tone.
- When a 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 21: IVR Control of Scripts on an IP Call Leg, on page 174 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.

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.

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 30: clid_col_npw_3 Script Prompt Audio Files, on page 175 lists the prompt audio files associated with the clid_col_npw_3 script.

<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 31: Additional clid_col_npw_3 Script Audio Files, on page 175 lists additional audio files associated with the clid_col_npw_3 script.

<table>
<thead>
<tr>
<th>Audio Filename</th>
<th>Action</th>
</tr>
</thead>
</table>
Informsthecallerthatauthorizationfailed. Prompts thecaller toreentertheaccountnumberfollowedby thepoundsign (#).

<table>
<thead>
<tr>
<th>Audio Filename</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth_fail_retry.au</td>
<td>Informs the caller that authorization failed. Prompts the caller to reenter the account number followed by the pound sign (#).</td>
</tr>
<tr>
<td>auth_fail_final.au</td>
<td>Informs the caller, &quot;I'm sorry, your account number cannot be verified. Please hang up and try again.&quot;</td>
</tr>
</tbody>
</table>

- **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 32: clid_col_npw_npw Script Audio Files, on page 176 lists the audio files associated with the clid_col_npw_npw script.

Table 32: 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.
• clid_col_npw_npw_cli.tcl—Authenticates by using ANI and null for account and PIN respectively. 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.

| Note | To display the contents of the TCL IVR script, use the show call application voice command. |

**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 30:clid_col_npw_3 Script Prompt Audio Files, on page 175** 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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> call application voice name url</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: flash:scripts/session.tcl</td>
</tr>
</tbody>
</table>

*Example:*  
**Router(config)# call application voice name url**
### Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| tftp://dirt/sarvi/scripts/session.tcl  
ftp://sarvi-ultra/scripts/session.tcl  
slot0:scripts/tcl/session..tcl | You can only configure a url if the application named `name` has not been configured. |

### Step 2

**call application voice name language digit language**

**Example:**

```
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:**

```
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:**

```
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:**

```
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:**

```
Router(config)# call application voice name set-location language 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 33: TCL Scripts and Parameters**

<table>
<thead>
<tr>
<th>TCL Script Name</th>
<th>Description—Summary</th>
<th>Commands to Configure</th>
</tr>
</thead>
</table>
| cliid_4digits_npw_3_cli.tcl      | 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. | 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                                                   |
| cliid_authen_col_npw_cli.tcl     | 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. | call application voice retry-count min = 1, max = 5, default = 3                      |
| cliid_authen_collect_cli.tcl     | 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. | call application voice retry-count min = 1, max = 5, default = 3                      |
| cliid_col_npw_3_cli.tcl          | 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.                                                      | call application voice retry-count min = 1, max = 5, default = 3                      |
| cliid_col_npw_npw_cli.tcl        | 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. The account number and PIN are collected together. | call application voice retry-count min = 1, max = 5, default = 3                      |
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>
<tbody>
<tr>
<td>Step 1</td>
<td><code>aaa new-model</code></td>
<td>(Optional) Enables AAA security and accounting services.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# aaa new-model</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# gw-accounting h323</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>aaa authentication login h323 radius</code></td>
<td>(Optional) Defines a method list called H.323 where RADIUS is defined as the only method of login authentication.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# aaa authentication login h323 radius</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>aaa accounting connection h323 start-stop radius</code></td>
<td>(Optional) Defines a method list called H.323 where RADIUS is used to perform connection accounting, providing start-stop records.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# aaa accounting connection h323 start-stop radius</code></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>radius-server host ip-address auth-port number acct-port number</code></td>
<td>Identifies the RADIUS server and the ports that will be used for authentication and accounting services.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# radius-server host ip-address auth-port number acct-port number</code></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>radius-server key key</code></td>
<td>Specifies the password used between the gateway and the RADIUS server.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Router(config)# radius-server key key</code></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 7</strong>&lt;br&gt;dial-peer voice number pots&lt;br&gt;Example: &lt;br&gt;Router(config)# dial-peer voice number pots</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>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong>&lt;br&gt;application name&lt;br&gt;Example: &lt;br&gt;Router(dial-peer)# application name</td>
<td>Associates the TCL IVR application with the incoming POTS dial peer. Enter the selected TCL IVR application name.</td>
<td></td>
</tr>
</tbody>
</table>
| **Step 9**<br>destination-pattern string<br>Example: <br>Router(config-dial-peer)# destination-pattern string | Enters the telephone number associated with this dial peer. The *pattern* 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 from A through D. The following special characters can be entered in the string:

- Plus sign (+)—(Optional) Indicates an E.164 standard number. The plus sign (+) is not supported on the Cisco MC3810 multiservice concentrator.
- *string*—Specifies the E.164 or private dialing plan telephone number. Valid entries are the digits 0 through 9, the letters A through D, and the following special characters:
  - –Asterisk (*) and pound sign (#) that appear on standard touch-tone dial pads.
  - –Comma (,) inserts a pause between digits.
  - –Period (.) matches any entered digit (this character is used as a wildcard).
  - T—(Optional) Indicates that the destination-pattern value is a variable length dial-string. |
| **Step 10**<br>session target<br>Example: <br>Router(config-dial-peer)# session target | Specifies the session target IP address. |

---

**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>
</table>
| **Step 1**
- dial-peer voice 4401 voip  
  Example:  
  `Router(config)# dial-peer voice 4401 voip`  
| Enters the dial-peer configuration mode and identifies the call leg. |
| **Step 2**
- application application-name  
  Example:  
  `Router(config-dial-peer)# application application-name`  
| Specifies the name of the application and script to use. |
| **Step 3**
- destination-pattern pattern  
  Example:  
  `Router(config-dial-peer)# destination-pattern pattern`  
| Enters the destination pattern. |
| **Step 4**
- session protocol sipv2  
  Example:  
  `Router(config-dial-peer)# session protocol sipv2`  
| Specifies the session protocol. The default session protocol is H.323. The `sipv2` argument enables SIP. |
| **Step 5**
- session target  
  Example:  
  `Router(config-dial-peer)# session target`  
| Specifies the session target IP address. |
| **Step 6**
- dtmf-relay cisco-rtp  
  Example:  
  `Router(config-dial-peer)# dtmf-relay cisco-rtp`  
| 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`.  
  **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**
- codec g711ulaw  
  Example:  
  `Router(config-dial-peer)# codec g711ulaw`  
| Specifies the voice codec.  
  **Note**  
  If the configured application will be playing prompts to the VoIP call leg, the `g711ulaw` keyword is required. |
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><code>tftp://hostname/scripts/nb/nb_handoffTwoLegs.tcl</code></td>
</tr>
<tr>
<td>hoone</td>
<td><code>tftp://hostname/scripts/nb/nb_dohandoff.tcl</code></td>
</tr>
<tr>
<td>hodest</td>
<td><code>tftp://hostname/scripts/nb/nb_handoff.tcl</code></td>
</tr>
<tr>
<td>cid</td>
<td><code>tftp://hostname/scripts/tcl_ivr/clid Authen_collect.tcl</code></td>
</tr>
<tr>
<td>db102</td>
<td><code>tftp://hostname/scripts/1.02/debitcard.tcl</code></td>
</tr>
<tr>
<td>*hw</td>
<td><code>tftp://171.69.184.xxx/tr_hello.tcl</code></td>
</tr>
</tbody>
</table>
```
The following table shows the applications supported:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*hw1</td>
<td>tftp://san*tr_db</td>
</tr>
</tbody>
</table>

TFTP URL: 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 22: Example Configuration Topology, on page 185 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 22: Example Configuration Topology**

---

**TCL IVR for Gateway1 (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 set-location en 0 tftp://dirt/hostname/WV/en_new/
call application voice debit_card_rtsp tftp://dirt/IVR 2.0/scripts.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 gki ipaddr 1.14.128.19 lxxx
  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 sipv2
 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
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 authentication login con none
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 radiusserver1 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#
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.

- Prerequisites for VoIP for IPv6, on page 191
- Restrictions for Implementing VoIP for IPv6, on page 191
- 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

Prerequisites for VoIP for IPv6

- Cisco Express Forwarding for IPv6 must be enabled.
- Virtual routing and forwarding (VRF) is not supported in IPv6 calls.
- One of the following releases is installed and running on your CUBE:
  - Cisco IOS Release 12.4(22)T or a later
  - Cisco IOS XE Release 3.3S or a later

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 is 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 do 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.

**IPv6 and IPv4 SRTP Interoperability**

- IPv6 and IPv4 SRTP interoperability is not supported.
- When dual stack is configured with preference to IPv4, crypto keys appear only under IPv4. It does not appear under IPv6.

When dual stack is configured with preference to IPv6, crypto keys appear only under IPv6. It does not appear under IPv4.
Information About VoIP for IPv6

SIP Features Supported on IPv6

The Session Initiation Protocol (SIP) is an alternate protocol that is 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 disabled in dual-stack mode (can be enabled if necessary).
- 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 that is 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:
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 forwarded 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.


• **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.


• **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.


• **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.


• **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 as 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 23: H.323/SIP IPv4 – SIP IPv6 interworking in media flow-through mode*

![Media Flow-Through Diagram](image)

• **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 24: H.323/SIP IPv4 - SIP IPv6 interworking in media flow-around mode*

![Media Flow-Around Diagram](image)

• **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-origininate 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 held 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-SIP Connections 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 busied 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 disabled 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 25: 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> 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> shutdown [ forced]</td>
<td>Shuts down or enables VoIP call services.</td>
</tr>
<tr>
<td>Example: device(config-voi-serv)# shutdown forced</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>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 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 | call service stop [forced] | Shuts down or enables VoIPv6 for the selected submode.  
Example: Device(config-serv-sip)# call service stop |

**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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device&gt; enable</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>sip-ua</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# sip-ua</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>protocol mode ipv4</td>
</tr>
<tr>
<td>***</td>
<td>ipv6</td>
</tr>
<tr>
<td>***</td>
<td>dual-stack</td>
</tr>
<tr>
<td>***</td>
<td>[preference {ipv4</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-sip-ua)# protocol mode dual-stack</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
```

#### Enabling ANAT Mode

ANAT is automatically disabled on SIP trunks in dual-stack mode. Perform this task to enable ANAT in order to use a single-stack mode.

#### SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. anat

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> 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> anat</td>
<td>Enables ANAT on a SIP trunk.</td>
</tr>
<tr>
<td>Example: Device(conf-serv-sip)# anat</td>
<td></td>
</tr>
</tbody>
</table>

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:
Verifying SIP Gateway Status

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
---------Printing Detailed Connection Report---------
Note:
** Tuples with no matching socket entry
  - Do 'clear sip <tcp[|ssl]/udp> conn t ipv4:<addr>:<port>''
to overcome this error condition
++ Tuples with mismatched address/port entry
  - Do 'clear sip <tcp[|ssl]/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
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 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> 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> sip-ua</td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> protocol mode {ipv4</td>
<td>ipv6</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-sip-ua)# protocol mode ipv6</td>
<td>• protocol mode dual-stack preference {ipv4</td>
</tr>
<tr>
<td></td>
<td>• protocol mode {ipv4</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits SIP user-agent configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# end</td>
<td></td>
</tr>
</tbody>
</table>

Verifying RTP Pass-Through

To enable RTCP packet-related debugging, use the following command

SUMMARY STEPS

1. debug voip rtcp packets
**Detailed Steps**

```plaintext
debug voip rtcp packets

Example:
Device# debug voip rtcp packets

*Feb 14 06:24:58.799: //1/xxxxxxxxxxxx/RTP//Packet/voip_remote_rtcp_packet: Received RTCP packet
*Feb 14 06:24:58.799: SR: ssrc=0x1F7A35F0 sr_ntp_h=0xD10346B4 sr_ntp_l=0x13173D8F sr_timestamp=0x0 sr_npackets=381 sr_nbytes=62176
*Feb 14 06:24:58.799: RR: ssrc=0x1A1752F0 rr_loss=0x0 rr_ehsr=5748 rr_jitter=0 rr_lsr=0x0 rr_dlsr=0x0
*Feb 14 06:24:58.799: SDES: ssrc=0x1F7A35F0 name=1 len=39 data=0.0.0@2001:DB8:C18:5:21B:D4FF:FEDD:35F0

*Feb 14 06:24:58.799: SR: ssrc=0x0 sr_ntp_h=0xD10346B4 sr_ntp_l=0x13173D8F sr_timestamp=0x0 sr_npackets=381 sr_nbytes=62176
*Feb 14 06:24:58.799: RR: ssrc=0x0 rr_loss=0x0 rr_ehsr=5748 rr_jitter=0 rr_lsr=0x0 rr_dlsr=0x0
*Feb 14 06:24:58.799: SDES: ssrc=0x1F7A35F0 name=1 len=39 data=0.0.0@2001:DB8:C18:5:21B:D4FF:FEDD:35F0
*Feb 14 06:24:58.919:
```

**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> 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 SIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> bind {control</td>
<td>media</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>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>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><em>Example:</em></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Device&gt; <code>enable</code></td>
<td><em>Enter your password if prompted.</em></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><em>Example:</em></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Device# <code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters SIP user agent configuration mode.</td>
</tr>
<tr>
<td><em>Example:</em></td>
<td>Enters SIP user agent configuration mode.</td>
</tr>
<tr>
<td>Device(config)# <code>sip-ua</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Configures a network address for the SIP server interface.</td>
</tr>
<tr>
<td><em>Example:</em></td>
<td>Configures a network address for the SIP server interface.</td>
</tr>
<tr>
<td>Device(config-sip-ua)# <code>sip-server ipv6: 2001:DB8:0:0:8:800:200C:417A</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Identifies SIP servers that will receive keepalive packets from the SIP gateway.</td>
</tr>
<tr>
<td><em>Example:</em></td>
<td>Identifies SIP servers that will receive keepalive packets from the SIP gateway.</td>
</tr>
<tr>
<td>Device(config-sip-ua)# <code>keepalive target ipv6: 2001:DB8:0:0:8:800:200C:417A</code></td>
<td></td>
</tr>
</tbody>
</table>

**Example: Configuring the SIP Server**

```
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`  
   
4. `destination pattern`  

---

**Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5**

---
5. session target {ipv4: destination-address} ipv6: [destination-address] | dns: $s$ | $d$s | $e$s | $u$s | host-name | enum:table-num | loopback:rtp | ras | sip-server} [: port

**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>Example:</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>Example:</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 {mmoip</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# dial-peer voice 29 voip</td>
</tr>
<tr>
<td></td>
<td>Defines a particular dial peer, specifies the method of voice encapsulation, and enters dial peer configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>destination pattern [+ string T]</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# destination-pattern 7777</td>
</tr>
<tr>
<td></td>
<td>Specifies either the prefix or the full E.164 telephone number to be used for a dial peer.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>session target {ipv4: destination-address} ipv6: [destination-address]</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# session target ipv6:2001:DB8:0:0:8:800:200C:417A</td>
</tr>
<tr>
<td></td>
<td>Designates a network-specific address to receive calls from a VoIP or VoIPv6 dial peer.</td>
</tr>
</tbody>
</table>

**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
Configuring SIP Register Support

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>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>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>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>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>sip-ua</code></td>
<td>Enters SIP user agent configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>`registrar {dns: address</td>
<td>ipv4: destination-address [: port]</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-sip-ua)# registrar ipv6: 2001:DB8::1:20F:F7FF:FE0B:2972 expires 3600 secondary</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>retry register retries</code></td>
<td>Configures the total number of SIP register messages that the gateway should send.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-sip-ua)# retry register 10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>timers register milliseconds</code></td>
<td>Configures how long the SIP UA waits before sending register requests.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></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>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>voice service voip</td>
<td>Enters voice service VoIP 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>sip</td>
<td>Enters sip configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>outbound-proxy {ipv4: ipv4-address</td>
<td>ipv6: ipv6-address</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-serv-sip)# outbound-proxy ipv6: 2001:DB8:0:0:8:800:200C:417A</td>
<td></td>
</tr>
</tbody>
</table>
# 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><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> ipv6 udp checksum [process</td>
<td>cef</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# ipv6 udp checksum process</td>
<td>Use the following keywords with the <em>ipv6 udp checksum</em> command:</td>
</tr>
<tr>
<td></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><strong>Example:</strong> 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
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>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>Device# 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>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Step 4 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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-serv)# ip address trusted list</td>
<td></td>
</tr>
<tr>
<td>Step 5 ipv6 $X:X::X$·X</td>
<td>Enters IPv6 addresses for toll fraud prevention.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-iptrust-list)# ipv6 2001:DB8:/48</td>
<td></td>
</tr>
<tr>
<td>Step 6 end</td>
<td>Exits trusted list configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>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><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong> voice service voip</td>
<td>Enters voice service VoIP configuration mode.</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 mode for Cisco UBE.</td>
</tr>
<tr>
<td><strong>Step 5</strong> media-address range range</td>
<td>Configures the media-address range, which enables the media gateway to allocate the available free port for a given IP 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>
</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>
<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>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; <code>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 7</strong></td>
<td></td>
</tr>
<tr>
<td><code>exit</code></td>
<td>Exits voice service VoIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-voi-ser)# <code>exit</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 8</strong></td>
<td></td>
</tr>
<tr>
<td><code>dial-peer voice tag voip</code></td>
<td>Enters dial peer configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# <code>dial-peer voice 300 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 9</strong></td>
<td></td>
</tr>
<tr>
<td><code>voice-class sip bind media source-interface interface</code></td>
<td>Matches the local SIP bind media IP address to the IP address range entries. Binds media packets to the IPv4 or IPv6 address of a specific interface and specifies an interface as the source address of SIP packets.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# <code>voice-class sip bind media source-interface GigabitEthernet 0</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 10</strong></td>
<td></td>
</tr>
<tr>
<td><code>end</code></td>
<td>Exits dial peer configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# <code>end</code></td>
<td></td>
</tr>
</tbody>
</table>
### 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>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. |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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-port <em>port number</em></td>
<td>Enters voiceport configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice-port 3</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> vmwi [fsk</td>
<td>dc-voltage]</td>
</tr>
<tr>
<td>Example: Device(config-voiceport)# vmwi fsk</td>
<td></td>
</tr>
<tr>
<td>• If an FSK phone is connected to the voice port, use the fsk keyword. Similarly, if a DC voltage phone is connected to the voice port, use the dc-voltage keyword.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits voice-port configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device(config-voiceport)# end</td>
<td></td>
</tr>
</tbody>
</table>

### 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 midcall-signaling passthru media-change
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.  
  Example:  
  Device> enable |
| **Step 2** configure terminal | Enters global configuration mode.  
  Example:  
  Device# configure terminal |
| **Step 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 midcall-signaling passthru media-change`  
  Example:  
  In Global VoIP SIP configuration mode:  
  Device(config)# voice service voip  
  Device(conf-voi-serv)# sip  
  Device(conf-serv-sip)# midcall-signaling passthru media-change  
  Example:  
  In Dial-peer configuration mode:  
  Device(config)# dial-peer voice 2 voip  
  Device(config-dial-peer)# voice-class sip midcall-signaling passthru media-change |
| **Step 4** end | Exits to privileged EXEC mode. |

### 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

<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> dial-peer voice dial-peer tag voip</td>
<td>Enters dial-peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# dial-peer voice 2 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class sip mid-call signaling passthru media-change</td>
<td>Passes through SIP messages that involve media change.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# voice-class sip mid-call signaling passthru media-change</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> 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).
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).

**Figure 27: IPv4 to IPv6 Media Interoperating Through Cisco IOS MTP**

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>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> 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: Device(config-voi-serv)# allow-connections h323 to sip</td>
<td>Arguments are as follows:</td>
</tr>
<tr>
<td></td>
<td>• <em>from-type</em> -- Type of connection. Valid values: <strong>h323</strong>, sip.</td>
</tr>
<tr>
<td></td>
<td>• <em>to-type</em> -- Type of connection. Valid values: <strong>h323</strong>, sip.</td>
</tr>
</tbody>
</table>

---

*Example: Configuring H.323 IPv4-to-SIPv6 Connections in a Cisco UBE*

```
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>/<min>/m <max> <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 dci 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
MODEMRELAY info:<rcvd>/sent/resent> xid:<rcvd>/sent total:<rcvd>/sent drops>
speeds (bps): local <rfk>/<tx> remote <rx>/<tx>
Proxy <ip>:<audio udp>,<video udp>,<tcp0>,<tcp1>,<tcp2>,<tcp3> endpt: <type>/<manf>
bw: <req>/<act> codec: <audio>/<video>
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: 0
Multicast call-legs: 0
Total call-legs: 2
0 : 987 361904110ms.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:0xB8 video tos:0x0
IP 2001:1111:2222:3333:4444:5555:6666:1012:38356 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms
g711ulaw TextRelay: off Transcoded: No
media inactive detected: 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
0 : 988 361904120ms.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:0xB8 video tos:0x0
IP 2001:1111:2222:3333:4444:5555:6666:1012:28356 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms
g711ulaw TextRelay: off Transcoded: No
media inactive detected: 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

Step 2  
**show call active voice compact**

**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
  987 ANS  T61  g711ulaw  VOIP  P1005  2001:......:1012:38356
  988 ORG  T61  g711ulaw  VOIP  P2005  2001:......:1012:26827
```

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>Default Address-Range</th>
<th>Ports Available</th>
<th>Ports Reserved</th>
<th>Ports In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>8091</td>
<td>101</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9.0.0.0</td>
<td>8091</td>
<td>101</td>
<td>1</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:
```
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: 4231321369-2948862433-2168455193-0797538600
User-Agent: Cisco-SIPGateway/IOS-12.x
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, P RACK, 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
s=SIP Call
c=IN IP4 9.44.30.10
t=0 0
a=group:ANAT 1 2
m=audio 16970 RTP/AVP 18 19
c=IN IP4 9.44.30.10
a=midi:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 17066 RTP/AVP 18 19
c=IN IP6 2001:DB8:C18:2:219:2FF:FE89:7928
a=midi:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20

*Jun 7 09:17:41.159: //31/FC34D7198140/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 100 Trying
VIA: SIP/2.0/UDP [2001:DB8:C18:2:219:2FF:FE89:7928]:5060;branch=z9hG4bK1GA8CD
Date: Thu, 07 Jun 2012 09:17:41 GMT
Timestamp: 1339066037
CSeq: 101 INVITE
Allow-Events: telephone-event
Server: Cisco-SIPGateway/IOS-15.2.20120528.102328.
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>
Verifying and Troubleshooting Cisco UBE ANAT Flow-Through Call

Date: Thu, 07 Jun 2012 09:17:41 GMT
Call-ID: 7780227E-ABF811E1-8060F4DD-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 off hook
Content-Length: 437

v=0
o=CiscoSystemsSIP-GW-UserAgent 3184 51 IN IP4 9.44.30.14
s=SIP Call
t=0 0
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
c=IN IP4 9.44.30.14
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-ABF811E1-8060F4DD-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-ABF811E1-8060F4DD-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: 1339060637
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 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
To: <sip:6000@9.44.30.11>;tag=6D0FF4-14D3
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-15.2.2.20120528.102328.
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
t=0 0
a=group:ANAT 1
m=audio 17200 RTP/AVP 18 19
c=IN IP6 2001:DB8:C18:2:217:59FF:FEDE:8898
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
c=IN IP4 9.44.30.11
a=mid:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20

*Jun 7 09:17:41.227: //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: 7780227-E-ABF811E1-8060F4DD-5665A1B9@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
To: <sip:6000@[2001:DB8:C18:2:219:2FFF:FE89:7928]>;party=called;screen=no;privacy=off
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: sd-p-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
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=rtmpmap:18 729/8000
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:2
a=rtmpmap:18 729/8000
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 17:19:26:30.681: //–/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=19hG4bK1CB1E77
```

---

**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>
<tr>
<td>2</td>
<td>32</td>
<td>31</td>
<td>16438</td>
<td>17200</td>
<td>9.44.30.10</td>
</tr>
</tbody>
</table>


Found 2 active RTP connections
Date: Thu, 07 Jun 2012 17:35:05 GMT
Call-ID: F44F5437-APFD11E1-816CD9DB-F669887E@2001:DB8:C18:2:223:33FF:FEB1:B440
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
c=IN IP4 9.44.30.13
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=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
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=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000

*Jun 7 17:26:30.705: //106/45C2DA68166/SIP/Msg/ccsipDisplayMsg: Sent:
SIP/2.0 100 Trying
Date: Thu, 07 Jun 2012 17:26:30 GMT
Call-ID: F44F5437-APFD11E1-816CD9DB-F669887E@2001:DB8:C18:2:223:33FF:FEB1:B440
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: //107/45C2DA68166/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=z9hG4bK0BB
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

Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5
Verifying Cisco UBE ANAT Flow-Around Calls

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: 117039766-2953384417-2170945561-0797538600
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: 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
o=CiscoSystemsSIP-GW-UserAgent 9582 2407 IN IP4 9.44.30.14
s=SIP Call
c=IN IP4 9.44.30.13
t=0 0
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: 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>
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: 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>
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
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:33FF:FE80:1]:5060;branch=z9hG4bK14B25D
Date: Thu, 07 Jun 2012 17:26:30 GMT
Call-ID: F44F5437-AFFD11E1-816CD9DF-F669887E@2001:DB8:C18:2:223:33FF:FE80:1
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-15.2.2.5.T
Content-Length: 0

*Jun 7 17:26:30.765: //106/45C2DA468166/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
Server: Cisco-SIPGateway/IOS-15.2.2.5.T
Content-Length: 0

v=0
o=CiscoSystemsSIP-GW-UserAgent 2764 5975 IN IP4 9.44.30.11
s=IN IP4 9.44.30.11
t=0 0
a=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=fmttype: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:6000@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:6000@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
From: <sip:1001@2001:DB8:C18:2:223:33FF:FEAC:4540>;tag=5569ECC8-C79
To: <sip:6000@2001:DB8:C18:2:223:4FF:FEAC:4540>;tag=22C9880-150D
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
Remote-Party-ID: <sip:6000@2001:DB8:C18:2:223:4FF:FEAC:4540>;party=called;screen=no;privacy=off
Contact: <sip:6000@2001:DB8:C18:2:223:4FF:FEAC:4540>;tag=5665AA1B
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
c=IN IP4 9.44.30.11
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
c=IN IP6 2001:DB8:C18:2:217:59FF:FEDE:8898
a-mid:2
a-rtpmap:18 G729/8000
a-fmtp:18 annexb=no
a-rtpmap:19 CN/8000
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: 0
Port range not configured, Min: 16384, Max: 32767

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Ports</th>
<th>Ports</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Address-Range</td>
<td>8091</td>
<td>101</td>
<td>0</td>
</tr>
</tbody>
</table>

No active connections found

---

### Verifying VMWI SIP

#### SUMMARY STEPS

1. **show sip-ua mwi**
2. **debug vpm signal**
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

---
Step 2  debug vpm signal

Example:
Device# debug vpm signal

Process vmwi. vmwi state: OFF
The phone is not on hook (1). Delay the vmwi processing. //Phone is offhook//
Process dc-voltage vmwi. State: OFF //VMWI state is off//
Mar 2 02:33:34.841: [2/0] c2400 dc_vol_t_mwi: on=0
The phone is not onhook (1). Delay the vmwi processing. Process vmwi. vmwi state: ON //VMWI state is on//
Voice port 0/2/1 subscribed MWI //Subscription of port for MWI (Solicited)//

Step 3  debug ccsip messages

Example:
Device# debug ccsip messages

Note  The debug ccsip messages command shows the SIP Messages, such as Subscribe and Notify.

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=fmtp: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=fmtp: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.

Sent:
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.

Supported: timer,resource-priority, replaces, sdp-anat
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
Expires: 180
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=za9hG4bk15d1013
Date: Fri, 08 Jun 2012 12:15:49 GMT
Call-ID: FB05CC74-B08E11E1-82CF4D0-5665AA1B@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

Received:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:4FF:FEAC:4540]:5060;branch=za9hG4bk15d1013
Date: Fri, 08 Jun 2012 12:15:49 GMT
Call-ID: FB05CC74-B08E11E1-82CF4D0-5665AA1B@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

Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:33FF:FBEB:1B440]:5060;branch=za9hG4bk20277F
Date: Fri, 08 Jun 2012 10:53:14 GMT
Call-ID: 2D6EEC84-B09011E1-8235D90B-F669887E@2001:DB8:C18:2:223:33FF:FBEB:1B440
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-15.2.2.20120528.102328.
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: FB05CC74-B08E11E1-82C1F4DD-5665AA1B02001: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
s=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=fmtp: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=fmtp: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: FB05CC74-B08E11E1-82C1F4DD-5665AA1B02001: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

Supported: replaces
Supported: sdp-anat
Server: Cisco-SIPGateway/IOS-15.2.2.20120528.102328.
Supported: timer
Content-Type: application/sdp
Content-Disposition: session;handling-required
Content-Length: 389

v=0
s=SIP Call
t=0 0
a=group:ANAT 1
m=audio 16710 RTP/AVP 18 19
a=mid:1
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
m=audio 0 RTP/AVP 18 19
c=IN IP4 9.44.30.14
a=mid:2
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000

Received:
Via: SIP/2.0/UDP [2001:DB8:C18:2:223:33FF:FEB1:B440]:5060;branch=z9hG4bK203700
Date: Fri, 08 Jun 2012 11:01:48 GMT
Call-ID: 2D6EEC84-B09011E1-8235D9DB-F669887E@2001:DB8:C18:2:223:33FF:FEB1:B440
Max-Forwards: 70
CSeq: 101 ACK
Allow-Events: telephone-event
Content-Length: 0

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

Media-Address Range | Ports Available | Ports Reserved | Ports In-use
---------------------|----------------|---------------|------------------
Default Address-Range| 8091           | 101           | 2

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 34: 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>
</tbody>
</table>
## Feature Information for VoIP for IPv6

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<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>Dynamic Payload Type</td>
<td></td>
<td>Interworking (DTMF and Codec Packets)</td>
</tr>
<tr>
<td>Interworking (DTMF and Codec Packets)</td>
<td></td>
<td>Voice Class Codec (VCC) with or without Transcoding</td>
</tr>
<tr>
<td>PPI/PAI/Privacy and RPID Passing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPv6 Dual Stack</td>
<td>12.4(22)T</td>
<td>IPv6 supports this feature. 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>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TDM-SIP GW for IPv6</td>
<td>12.4(24)T</td>
<td>IPv6 supports this feature.</td>
</tr>
<tr>
<td></td>
<td>15.3(2)T</td>
<td>• Session Initiation Protocol Features Supported on IPv6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Cisco UBE features Supported on IPv6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP Gateway Generic Features</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP VMWI for FXS phones</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• History-Info</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Handling 181/183 Responses with/without SDP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP Session Timer (4028)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SIP Media Inactivity Detection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PPI/PAI &amp; Privacy (RFC3323/RFC3325) Headers</td>
</tr>
</tbody>
</table>
CHAPTER 17

Monitoring of Phantom Packets

The Monitoring of Phantom Packets feature allows you to configure port ranges specific to the VoIP Real-Time Transport Protocol (RTP) layer. This allows the VoIP RTP layer to safely drop packets without proper sessions (phantom packets) received on these ports of the Cisco Unified Border Element (CUBE) or Voice time-division multiplexing (TDM) gateways. Because the ports are configured specifically for the VoIP RTP layer, puntting the packets to UDP process is not required. This helps in reducing the performance issues.

- Restrictions of Monitoring of Phantom Packets, on page 247
- Information About Monitoring of Phantom Packets, on page 248
- How to Configure Monitoring of Phantom Packets, on page 248
- Configuration Examples For Monitoring of Phantom Packets, on page 250
- Additional References for Configurable Pass-Through of SIP INVITE Parameters, on page 250
- Feature Information for Monitoring of Phantom Packets, on page 251

Restrictions of Monitoring of Phantom Packets

- The authentication, authorization, and accounting (AAA) default port range of 21645 to 21844 must not be configured.
- Up to ten port range entries can be defined under a single media-address range.
- The minimum port must be numerically lower than the maximum port.
- Port ranges should not overlap.
- Address ranges should not overlap.
- Address ranges and single addresses should not overlap.
- Where a range of addresses are defined in a single command, they will share any port ranges assigned. If there is a requirement to have different port ranges for different media addresses, then the addresses must be configured separately.
- If you are using the Transmission Control Protocol (TCP), the interface used for media and signaling should be different.
- For TCP, the media address and the signaling address should not be identical. If the media address and the signaling address are identical, and the Cisco IOS XE based router platform (Cisco ASR 1000 Series Aggregation Services Router, Cisco 4000 Series Integrated Services Routers, or Cisco Cloud Services Router 1000V Series) selects an ephemeral port to send out signaling packets, the port may overlap with...
the port range of the media address. As a result, the signaling packets do not get punted up to the RP, and get dropped by the media packet filter. This may result in events such as incomplete TCP handshakes during the second leg of a call through CUBE or Voice Gateways.

Information About Monitoring of Phantom Packets

Monitoring of Phantom Packets

The Monitoring of Phantom Packets feature allows you to configure port ranges specific to the VoIP Real-Time Transport Protocol (RTP) layer. This configuration allows the VoIP RTP layer to safely drop packets without proper sessions (phantom packets) received on the ports of the Cisco Unified Border Element (CUBE) or Voice time-division multiplexing (TDM) gateways. Because the ports are configured specifically for the VoIP RTP layer, there is no need to punt the packets to the UDP process in case the packets were intended for some other application, thus reducing performance issues.

A phantom packet is a valid RTP packet meant for the CUBE or Voice TDM gateway without an existing session on the respective gateways. When a phantom packet is received by the VoIP RTP layers of the gateways, the packet is punted to the UDP process to check if it is required by any other applications causing performance issues, especially when a large number of such packets are received. A malicious attacker can also send a large number of phantom packets. The packet is punted to the UDP process because UDP port ranges are shared by many applications other than VoIP RTP and the VoIP RTP layer cannot drop the packet assuming the packet is for itself.

We recommend that you configure the IP address and port ranges specific to the media IP addresses, even if you are using a single virtual IP address for media. This feature allows you to configure port ranges specific to the VoIP RTP layer. If a phantom packet is received on the configured port, the VoIP RTP layer can safely drop the packet. If a phantom packet is received on any other port, the VoIP RTP layer punts the packet to the UDP process.

How to Configure Monitoring of Phantom Packets

Configuring Monitoring of Phantom Packets

SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. media-address range starting-ip-address ending-ip-address  port range starting-port-number ending-port-number
5. port-range starting-port-number ending-port-number
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>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</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>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>voice service voip</td>
<td>Specifies VoIP encapsulation and enters voice-service configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>media-address range starting-ip-address ending-ip-address port range starting-port-number ending-port-number</td>
<td>Configures an IPv4 or IPv6 media address range. And, creates a port range for the configured media addresses. <strong>Note</strong> If you do not configure any port range, the default port range is applied. The default port range is 8000-48198 for ASR and ISR G3 platforms, and 16384-32766 for ISR G2 platforms.</td>
</tr>
</tbody>
</table>
| Example: | Using IPv4 addresses:  
For single IP:  
Device(conf-voi-serv)# media-address range 10.1.1.1 10.1.1.1  
For a range of IPs:  
Device(conf-voi-serv)# media-address range 10.1.1.1 10.1.1.254  
Using IPv6 addresses:  
For single IP:  
Device(conf-voi-serv)# media-address range 2001:DB8:1::1 2001:DB8:1::1  
For a range of IPs:  
Device(conf-voi-serv)# media-address range 2001:DB8:1::1 2001:DB8:1::17  
Port range for media address.  
Device(cfg-media-addr-range)# port-range 8000 48198 |
<p>| Step 5 | port-range starting-port-number ending-port-number | Configures a port range. If you do not configure any port range nothing is applied. <strong>Note</strong> Ensure that the port range is not greater than the port range (if configured) specified in the media-address range command. |
| Example: | Device(cfg-media-addr-range)# port-range 8000 48198 |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 6</td>
<td><strong>end</strong></td>
</tr>
<tr>
<td>Example:</td>
<td><strong>Router(cfg-media-addr-range)# end</strong></td>
</tr>
<tr>
<td></td>
<td>Exits voice-service configuration mode and returns to</td>
</tr>
<tr>
<td></td>
<td>privileged EXEC mode.</td>
</tr>
</tbody>
</table>

**Configuration Examples For Monitoring of Phantom Packets**

```
 Device(config)# voice service voip  
 Device(conf-voi-serv)# media-address range 10.1.1.1 10.1.1.254  
 Device(cfg-media-addr-range)# port-range 8000 21643  
 Device(cfg-media-addr-range)# port-range 21846 48000  
 Device(cfg-media-addr-range)# exit  

 Device(conf-voi-serv)# media-address range 2001:DB8:1::1 2001:DB8:1::17  
 Device(cfg-media-addr-range)# port-range 8000 21643  
 Device(cfg-media-addr-range)# port-range 21846 48000  
 Device(cfg-media-addr-range)# end
```

**Note**
The ports from 21643 to 21845 are not used by the RTP layer. They might be used by applications such as AAA/Radius. These ports are allowed to be punt to the control plane if needed.

**Additional References for Configurable Pass-Through of SIP INVITE Parameters**

**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 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. 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. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
</tbody>
</table>

Feature Information for Monitoring of Phantom Packets

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 Monitoring of Phantom Packets

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring of Phantom Packets</td>
<td>Cisco IOS XE Release 3.9S 15.4(1)T</td>
<td>This feature allows you to configure port ranges specific to the VoIP Real-Time Transport Protocol (RTP) layer and drop phantom RTP packets (RTP packets that are configured in valid port range but for which there is no matching call or session). The following commands were introduced: <strong>port-range</strong>, <strong>media-address range</strong>.</td>
</tr>
</tbody>
</table>
CHAPTER 18

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, on page 253
• Prerequisites for Configurable SIP Parameters via DHCP, on page 253
• Restrictions for Configurable SIP Parameters via DHCP, on page 254
• Information About Configurable SIP Parameters via DHCP, on page 254
• How to Configure SIP Parameters via DHCP, on page 258
• Feature Information for Configurable SIP Parameters via DHCP, on page 265

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 https://cfnng.cisco.com/. 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 28: 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
- DHCPFORCE.RENEW 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.
Figure 30: DHCP Message Details

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 DHCPFORCERENEW to the Cisco Unified Border Element. If the authentication is successful, the Cisco Unified Border Element DHCP client accepts the DHCPFORCERENEW and moves to the next stage of sending DHCPREQUEST. Otherwise DHCPFORCERENEW is ignored and the current information is retained and used.

6. F6 and F7: In response to DHCPFORCERENEW, 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>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>interface type number</td>
<td>Configures an interface type and enters interface configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>router(config)# interface gigabitethernet 0/0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>ip dhcp client request sip-server-address</td>
<td>Configures the DHCP client to request a SIP server address from a DHCP server.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>router(config-if)# ip dhcp client request sip-server-address</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>ip dhcp client request vendor-identifying-specific</td>
<td>Configures the DHCP client to request vendor-specific information from a DHCP server.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>router(config-if)# ip dhcp client request vendor-identifying-specific</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td>ip address dhcp</td>
<td>Acquires an IP address on the interface from the DHCP.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>router(config-if)# ip address dhcp</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(config-if)# exit</td>
<td></td>
</tr>
</tbody>
</table>

**Configuring the DHCP Client Example**

The following is an example of how to enable the DHCP client:

```
Router> enable
Router# configure terminal
Router(config)# interface gigabitethernet 1/1
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> 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> interface type number</td>
<td>Configures an interface type and enters interface configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> 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> 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>
</tbody>
</table>
## Command or Action

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Command or Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Multiple interfaces on the CUBE can be configured with DHCP—this command specifies the DHCP interface used with SIP.</td>
<td>Router(sip-ua)# dhcp interface gigabitethernet 0/0</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

- **expires seconds** -- Specifies the default registration time, in seconds. Range is 60 to 65535. Default is 3600.
- **refresh-ratio seconds** -- Specifies the refresh-ratio, in seconds. Range is 1 to 100 seconds. Default is 80.

### Step 7

**credentials dhcp password [0-7] password realm domain-name**

**Example:**

Router(sip-ua)# credentials dhcp password cisco realm cisco.com

Sends a SIP registration message from a Cisco Unified Border Element in the UP state.

### Step 8

**exit**

**Example:**

Router(sip-ua)# exit

Exits the current mode.

---

### 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></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>Router&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:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters voice service VoIP configuration mode and specifies VoIP as the voice-encapsulation type.</td>
</tr>
<tr>
<td>voice service voip</td>
<td></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>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td>sip</td>
<td></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>Router(config-voi-srv)# sip</code></td>
<td>Configures the DHCP client to request a SIP server address from a DHCP server.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>outbound-proxy dhcp</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(conf-serv-sip)# outbound-proxy dhcp</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-serv-sip)# exit</td>
</tr>
</tbody>
</table>

**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-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]

**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`

<table>
<thead>
<tr>
<th><strong>DETAILED STEPS</strong></th>
<th><strong>Command or Action</strong></th>
<th><strong>Purpose</strong></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.

Feature History Table for the ISR.

Table 36: 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></td>
</tr>
</tbody>
</table>

The following commands were introduced or modified: credentials (sip-ua), debug ccsip dhcp, dhcp interface, ip dhcp-client forcerenew, outbound-proxy, registrar, session target (VoIP dial peer), show sip dhcp, voice-class sip outbound-proxy.

Feature History Table for the ASR.

Table 37: 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>
</tbody>
</table>

The following commands were introduced or modified: credentials (sip-ua), debug ccsip dhcp, dhcp interface, ip dhcp-client forcerenew, outbound-proxy, registrar, session target (VoIP dial peer), show sip dhcp, voice-class sip outbound-proxy.
PART II

Dial Peer Enhancements

• Matching Inbound Dial Peers by URI, on page 269
• URI-Based Dialing Enhancements, on page 273
• Multiple Pattern Support on a Voice Dial Peer, on page 287
• Outbound Dial-Peer Group as an Inbound Dial-Peer Destination, on page 293
• Inbound Leg Headers for Outbound Dial-Peer Matching, on page 303
• Server Groups in Outbound Dial Peers, on page 313
• Domain-Based Routing Support on the Cisco UBE, on page 323
• ENUM Enhancement per Kaplan Draft RFC, on page 331
CHAPTER 19

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 (neighbor SIP entity), which is the topmost via header.

Note
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.

- Configuring an Inbound Dial Peer to Match on URI, on page 269
- Examples for Configuring an Inbound Dial Peer to Match on a URI, on page 271

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
### 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>
</tbody>
</table>

| **Step 2** configure terminal | Enters global configuration mode. |
| Example: Device> configure terminal | |

| **Step 3** voice class uri `voice-class-uri-tag` | Creates a voice class for matching SIP dial peers and enters voice URI class configuration mode. |
| Example: Device(config)# voice class uri 200 | |

**Step 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](#)

**Example:**

- Device(config-vvoice-uri-class)# host server1
- Device(config-voice-uri-class)# host ipv4:10.0.0.0
- Device(config-voice-uri-class)# host dns:xxx.yyy.com

**Note** Patterns are case-sensitive.

**Step 5** exit

**Example:**

- Device(config-voice-uri-class)# exit

Enters global configuration mode.
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Step 6 | dial-peer voice tag voip  
Example:  
Device(config)# dial-peer voice 6000 voip | Enters dial peer voice configuration mode. |
| Step 7 | session protocol sipv2  
Example:  
Device(config-dial-peer)# session protocol sipv2 | Configures SIP as the session protocol type. |
| Step 8 | incoming uri { from | request | to | via } voice-class-uri-tag  
Example:  
Device(config-dial-peer)# incoming uri via 200 | Configures the voice class with an inbound dial peer, so that it is matches against configured URI fields. |
| Step 9 | end  
Example:  
Device(config-dial-peer)# end | Exits dial peer voice configuration mode and enters privileged EXEC mode. |

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

```
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.

```
INVITE sip:12301.2.3.4:5060 SIP/2.0  
Via: SIP/2.0/TCP 10.10.10.1:5093;branch=z9hG4bK-17716-1-0  
Via: SIP/2.0/TCP 10.14.20:5093;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.

```
voice class uri 202 sip  
host dns:sample.com
```
Examples for Configuring an Inbound Dial Peer to Match on a URI

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
dial-peer voice 103 voip
    session protocol sipv2
    incoming uri via 203

Incoming INVITE that can be matched against this dial peer.

INVITE sip:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP sample.com;branch=z9hG4bK-17716-1-0
INVITE sip:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP 10.10.10.25:5093;branch=z9hG4bK-17716-1-0

10.10.10.25 is a resolved IP address of sample.com.
```

```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.

INVITE sip:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP sample.com;branch=z9hG4bK-17716-1-0
INVITE sip:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP 10.10.14.20:5093;branch=z9hG4bK-28280-1-0

10.10.25 is a resolved IP address of sample.com.
```
CHAPTER 20

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).

- Feature Information for URI-Based Dialing Enhancements, on page 273
- Information About URI-Based Dialing Enhancements, on page 274
- How to Configure URI-Based Dialing Enhancements, on page 277
- Configuration Examples for URI-Based Dialing Enhancements, on page 284
- Additional References for URI-Based Dialing Enhancements, on page 285

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>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI-Based Dialing Enhancements</td>
<td></td>
<td>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:<a href="mailto:user@abc.com">user@abc.com</a>). The following commands were introduced or modified: contact-passing, requiri-passing, session target sip-uri and voice-class sip requiri-passing</td>
</tr>
</tbody>
</table>
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.

Note

The minimum supported release of Cisco IOS required for URI based call routing on dial-peers is Cisco IOS XE Gibraltar Release 16.12. You must configure the 'call-route-url' on the outgoing dial-peers to properly route the refer-to headers based on the URI matching.

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 `requiri-passing` command is enabled.

Case 3: The old behavior of setting the outbound Request-URI to session target is retained when the `requiri-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.

Case 8: URI routing based on BYE Also header.
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. requri-passing
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>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>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3 voice service voip</td>
<td>Specifies VoIP encapsulation and enters voice service configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

Command or Action | Purpose
--- | ---
Device(config)# voice service voip |  

#### Step 4

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip</td>
<td>Enters the Session Initiation Protocol (SIP) configuration mode.</td>
</tr>
</tbody>
</table>

#### Example:

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip</td>
<td>Device(conf-voi-serv)# sip</td>
</tr>
</tbody>
</table>

#### Step 5

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>requiri-passing</td>
<td>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.</td>
</tr>
</tbody>
</table>

#### Example:

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>requiri-passing</td>
<td>Router(conf-serv-sip)# requiri-passing</td>
</tr>
</tbody>
</table>

#### Step 6

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>end</td>
<td>Ends the current configuration session and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

#### Example:

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>end</td>
<td>Router(conf-serv-sip)# end</td>
</tr>
</tbody>
</table>

---

### Configuring Pass Though 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

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>voice class uri 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>
</tbody>
</table>

#### Example:

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Device&gt; enable</td>
</tr>
<tr>
<td>configure terminal</td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td>voice class uri tag sip</td>
<td>Device(config)# voice class uri mydesturi sip</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| **Step 4** | host *hostname-pattern*  
**Example:**  
Device(config-voice-uri-class)# host example.com | Matches a call based on the host field in a SIP Uniform Resource Identifier (URI). |
| **Step 5** | exit  
**Example:**  
Device(config-voice-uri-class)# exit | Exits voice URI class configuration mode. |
| **Step 6** | dial-peer voice *tag* voip  
**Example:**  
Device(config)# dial-peer voice 22 voip | Defines a VoIP dial peer and enters dial peer configuration mode. |
| **Step 7** | session protocol sipv2  
**Example:**  
Device(config-dial-peer)# 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*  
**Example:**  
Device(config)# destination uri mydesturi | 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*  
**Example:**  
Device(config-dial-peer)# session target ipv4:10.1.1.2 | Designates a network-specific address to receive calls from a VoIP. |
| **Step 10** | voice-class sip requiri-passing [system]  
**Example:**  
Device(config-dial-peer)# voice-class sip requiri-passing system | Enables the pass through of SIP URI headers. |
| **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>Step</th>
<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 | Specifies VoIP encapsulation and enters voice service 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 | contact-passing | Enables pass through of the contact header from one leg to the other leg in 302 pass through scenario.  
Example: Router(conf-serv-sip)# contact-passing | |
| Step 6 | end | Ends the current configuration session and returns to privileged EXEC mode.  
Example: Router(conf-serv-sip)# end | |

**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
### 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 a voice class for matching dial peers to a Session Initiation Protocol (SIP) and enters voice URI class configuration mode. 
**Example:** 
Device(config)# voice class uri mydesturi sip |
| Step 4 | **user-id id-tag** | Matches a call based on the User ID portion of the Uniform Resource Identifier (URI). 
**Example:** 
Device(config-voice-uri-class)# user-id 5678 |
| Step 5 | **exit** | Exits voice URI class configuration mode. 
**Example:** 
Device(config-voice-uri-class)# exit |
| Step 6 | **voice service voip** | Specifies Voice over IP (VoIP) as the voice encapsulation type and enters voice service configuration mode. 
**Example:** 
Device(config)# voice service voip |
| Step 7 | **allow-connections sip to sip** | Allows connections between SIP endpoints in a VoIP network. 
**Example:** 
Device(conf-voi-serv)# allow-connections sip to sip |
| Step 8 | **dial-peer voice tag voip** | Defines a VoIP dial peer and enters dial peer configuration mode. 
**Example:** 
Device(config)# dial-peer voice 200 voip |
| Step 9 | **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 10 | **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 |
### 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
13. 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> voice class uri destination-tag sip</td>
<td>Creates or modifies a voice class for matching dial peers to a Session Initiation Protocol (SIP) or telephone (TEL)</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>Device(config)# voice class uri mydesturi sip</code></td>
<td>Uniform Resource Identifier (URI) and enters voice URI class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>host  <em>hostname-pattern</em></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-voice-uri-class)# host destination.com</td>
</tr>
<tr>
<td></td>
<td>Matches a call based on the host field in a SIP URI.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>exit</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-voice-uri-class)# exit</td>
</tr>
<tr>
<td></td>
<td>Exits voice URI class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>dial-peer voice  <em>tag voip</em></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# dial-peer voice 25 voip</td>
</tr>
<tr>
<td></td>
<td>Defines a VoIP dial peer and enters dial peer configuration mode.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>session protocol sipv2</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td></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 8</strong></td>
<td>destination uri  <em>destination-tag</em></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# destination uri mydesturi</td>
</tr>
<tr>
<td></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 9</strong></td>
<td>session target sip-uri</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session target sip-uri</td>
</tr>
<tr>
<td></td>
<td>Derives session target from incoming URI.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>exit</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# exit</td>
</tr>
<tr>
<td></td>
<td>Exits dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>voice class uri  <em>source-tag sip</em></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# voice class uri mysourceturi sip</td>
</tr>
<tr>
<td></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></td>
<td>host  <em>hostname-pattern</em></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-voice-uri-class)# host abc.com</td>
</tr>
<tr>
<td></td>
<td>Matches a call based on the host field in a SIP URI.</td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>end</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-voice-uri-class)# end</td>
</tr>
<tr>
<td></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)

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(config)# sip
Device(config)# requri-passing
Device(config)# end
```

Example: Configuring Pass Though of Request URI and To Header URI (Dial Peer Level)

```
! 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)# sip
Device(config)# contact-passing
Device(config)# 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
Dial Peer Enhancements

Example: Deriving Session Target from URI

Device> enable
Device# configure terminal
Device(config)# voice class uri mydesturi sip
Device(config)# host destination.com
Device(config)# 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 mysourceuri sip
Device(config)# host abc.com
Device(config)# end

Additional References for URI-Based Dialing Enhancements

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 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</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</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>
CHAPTER 21

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 287
- Restrictions for Multiple Pattern Support on a Voice Dial Peer, on page 288
- Information About Multiple Pattern Support on a Voice Dial Peer, on page 288
- Configuring Multiple Pattern Support on a Voice Dial Peer, on page 288
- Verifying Multiple Pattern Support on a Voice Dial Peer, on page 290
- Configuration Examples for Multiple Pattern Support on a Voice Dial Peer, on page 292

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>
</table>
| Configuring Multiple Pattern Support on a Voice Dial Peer (Inbound Calls) | Cisco IOS 15.4 (1)T  
Cisco IOS XE 3.11S | This feature was extended for inbound VoIP dial peers for incoming calling and called numbers. The following commands were introduced or modified: incoming called e164-pattern-map, incoming calling e164-pattern-map |
### Feature Information

<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 (Outbound Calls)</td>
<td>Cisco IOS 15.2(4)M Cisco IOS XE 3.7S</td>
<td>This feature allows you to add more than one E.164 destination pattern inside a pattern map and configure that pattern map for one or more VoIP dial peers. This feature is supported for outbound peers only. The following commands were introduced or modified: destination e164-pattern-map, e164, show voice class e164-pattern-map, url, voice class e164-pattern-map load, voice class e164-pattern-map.</td>
</tr>
</tbody>
</table>

#### Restrictions 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> enable</td>
<td>Enters 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><strong>Step 3</strong> voice class e164-pattern-map pattern-map-id</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>Example: <code>Device(config)# voice class e164-pattern-map 1111</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> Do one of the following:</td>
<td>Configure one or more E.164 telephone number prefix match patterns for the pattern map.</td>
</tr>
<tr>
<td>• <code>e164 pattern-map-tag</code></td>
<td>• Repeat this step for each pattern if you are using the <code>e164</code> command.</td>
</tr>
<tr>
<td>• <code>url url</code></td>
<td>• You can specify a file URL containing the patterns for this dial peer using the <code>url url</code> 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>Example: Using URL text file: <code>Device(voice-class)# url http://http-host/config-files/pattern-map.cfg</code></td>
<td></td>
</tr>
<tr>
<td>Directly specifying match patterns: <code>Device(voice-class)# e164 5557123</code></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>Provides a description for the pattern map.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>description string</strong></td>
</tr>
<tr>
<td>Device(voice-class)# description It has 1 entry</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Exits voice class configuration mode and enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td>Device(voice-class)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Defines a VoIP dial peer and enters dial peer configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>dial-peer voice dial-peer-id voip</strong></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 2222 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Links a pattern-map group with a dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>**{destination</td>
</tr>
<tr>
<td>Device(config-dial-peer)# incoming calling e164-pattern-map 1111</td>
<td>• Use the <strong>destination</strong> keyword for outbound dial peers.</td>
</tr>
<tr>
<td></td>
<td>• Use the <strong>incoming called</strong> or <strong>incoming calling</strong> keywords for inbound dial peers using called or calling numbers.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>Exits dial peer configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>end</strong></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>Loads the specified pattern map with E.164 match patterns from a text file configured in the pattern map.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>(Optional) voice class e164-pattern-map load pattern-map-group-id</strong></td>
</tr>
<tr>
<td>Device# voice class e164-pattern-map load 1111</td>
<td>• This step is required only if patterns have been defined for the specified pattern map using a file URL in Step 4.</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>Displays the status of a pattern map when the pattern map is associated with a dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>**show dial-peer voice [summary</td>
</tr>
<tr>
<td>Device# show dial-peer voice 1111</td>
<td></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 = 300 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 by Specifying 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 22

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 293
- Restrictions, on page 294
- Information About Outbound Dial-Peer Group as an Inbound Dial-Peer Destination, on page 294
- Configuring Outbound Dial-Peer Group as an Inbound Dial-Peer Destination, on page 295
- Verifying Outbound Dial-Peer Groups as an Inbound Dial-Peer Destination, on page 297
- Troubleshooting Tips, on page 298
- Configuration Examples for Outbound Dial Peer Group as an Inbound Dial-Peer Destination, on page 299

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 40: 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>
### Feature Information

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound Dial-Peer Group as an Inbound Dial-Peer Destination</td>
<td>Cisco IOS 15.4(1)T, Cisco IOS XE 3.11S</td>
<td>This feature groups multiple outbound dial-peers into a dial-peer group and configures this dial-peer group as a destination of an inbound dial peer. The following commands were introduced or modified: <code>voice class dpg</code>, <code>description</code>, <code>dial-peer preference</code>, <code>destination dpg</code>, show <code>voice class dpg</code>.</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 though 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>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enable</td>
<td>Enters privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>dial-peer voice outbound-dial-peer-id [voip</td>
<td>pots]</td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| Step 4 | Configures a destination pattern. This step is required even though the value is not used for dial-peer matching. | **For VoIP Dial Peers**  
Device(config-dial-peer)# destination-pattern 1004  
**For POTS Dial Peers**  
Device(config-dial-peer)# destination-pattern .T |
| Step 5 | Disable unexpected dialed digit strip.                                       | **no digit-strip** for POTS dial peers.  
Device(config-dial-peer)# no digit-strip |
| Step 6 | Exits to global configuration mode.                                          | **exit**  
Device(config-dial-peer)# exit |
| Step 7 | Specifies a hunt selection mechanism for dial peers.  
• The default mechanism is random selection. | **(Optional) dial-peer hunt**  
Device(config)# dial-peer hunt 0 |
| Step 8 | Creates a dial-peer group for grouping multiple outbound dial peers and enters voice class configuration mode.  
• You can use the shutdown command to resume regular outbound dial-peer provisioning in dial-peers with this dial-peer group as destination. | **voice class dpg**  
Device(config)# voice class dpg 181 |
| Step 9 | Associates a configured outbound dial peer with this dial-peer group and configures a preference value.  
• Repeat this step for all outbound dial-peers that need to be added to this dial-peer group. | **dial-peer outbound-dial-peer-id [preference preference-order]**  
Device(config-class)# dial-peer 123 preference 1 |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• If preference is not specified, the order of selection is random or as specified by the <strong>dial-peer hunt</strong> command.</td>
<td></td>
</tr>
</tbody>
</table>

**Step 10**  
(Optional) **description string**  
**Example:**  
Device(config-class)# **description Boston Destination**  
Provides a description for the dial-peer group.  

**Step 11**  
**exit**  
**Example:**  
Device(config-class)# **exit**  
Exits voice class configuration mode and enters global configuration mode.  

**Step 12**  
**dial-peer voice inbound-dial-peer-id [voip | pots]**  
**Example:**  
For VoIP dial peer:  
Device(config)# **dial-peer voice 789 voip**  
**Example:**  
For POTS dial peer:  
Device(config)# **dial-peer voice 678 pots**  
Defines a dial peer and enters dial peer configuration mode.  

**Step 13**  
**destination dpg dial-peer-group-id**  
**Example:**  
Device(config-dial-peer)# **destination dpg 181**  
Specifies a dial peer group from which an outbound dial peer can be chosen.  

**Step 14**  
**end**  
**Example:**  
Device(config-dial-peer)# **end**  
Exits dial peer configuration mode and enters privileged EXEC mode.  

### 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

Device> enable
Device# configure terminal
  ! Configuring outbound dial peers that are to be grouped.
Device(config)# dial-peer voice 1001 voip
Device(config-dial-peer)# destination-pattern 1001
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.1.1.1
Device(config-dial-peer)# exit

Device(config)# dial-peer voice 1002 voip
Device(config-dial-peer)# destination-pattern 1002
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.1.1.2
Device(config-dial-peer)# exit

Device(config)# dial-peer voice 1003 voip
Device(config-dial-peer)# destination-pattern 1003
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.1.1.3
Device(config-dial-peer)# exit

Device(config)# dial-peer voice 1004 pots
Device(config-dial-peer)# destination-pattern 5...
Device(config-dial-peer)# no digit-strip
Device(config-dial-peer)# direct-inward-dial
Device(config-dial-peer)# port 1/0/0:23
Device(config-dial-peer)# forward-digits all
Device(config-dial-peer)# exit

!Grouping outbound dial peers and configuring preferences if needed.
Device(config)# voice class dpg 200
Device(config-class)# dial-peer 1001 preference 1
Device(config-class)# dial-peer 1002 preference 2
Device(config-class)# dial-peer 1003 preference 3
Device(config-class)# dial-peer 1004 preference 4
Device(config-class)# description Boston Destination
Device(config-class)# exit

!Associating outbound dial peer group with an inbound dial peer group.
Device(config)# dial-peer voice 100 voip
Device(config-dial-peer)# incoming called-number 13411
Device(config-dial-peer)# destination dpg 200
Device(config-dial-peer)# end

!Associating outbound dial peer group with an inbound POTS dial peer group.
Device(config)# dial-peer voice 600 pots
Device(config-dial-peer)# incoming called-number 4T
Device(config-dial-peer)# destination dpg 200
Device(config-dial-peer)# end

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>1004</td>
<td>0</td>
</tr>
<tr>
<td>1003</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

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 41: Feature Information for Inbound Leg Headers for Outbound Dial-Peer Matching

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound Leg Headers for Outbound Dial-Peer Matching</td>
<td>15.4(2)T, Cisco IOS XE Release 3.12S</td>
<td>The Inbound Leg Headers for Outbound Dial-Peer Matching feature allows you to match and provision an outbound call leg using the headers of an inbound call leg. The following commands were introduced by this feature: destination provision-policy, destination uri-via, destination uri-to, destination uri-from, destination uri-diversion, destination uri-referred-by, show voice class dial-peer provision-policy. The following commands were modified. show command incall, show dialplan dialpeer.</td>
</tr>
</tbody>
</table>

### 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)
- FROM (SIP Header)
- TO (SIP Header)
- DIVERSION (SIP Header)
- REFERRED BY (SIP Header)
- Called Number
- Calling Number
- Carrier ID

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><strong>Step 1</strong> enable</td>
<td>Enters 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 class dial-peer provision-policy tag</td>
<td>Creates a provision policy profile in which a set of attributes for dial-peer matching can be defined.</td>
</tr>
<tr>
<td>Example:</td>
<td>• You can use the <code>shutdown</code> command to deactivate the provision policy and allow normal outbound dial-peer provisioning.</td>
</tr>
<tr>
<td>Device(config)# voice class dial-peer provision-policy 200</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> (Optional) description string</td>
<td>Provides a description for the provision policy profile.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(voice-class)# description match both calling and called</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> preference preference-order first-attribute second-attribute</td>
<td>Configures a provision policy rule.</td>
</tr>
<tr>
<td>First Attribute</td>
<td>Second Attribute</td>
</tr>
<tr>
<td>diversion</td>
<td>from, referred-by, to, uri, via</td>
</tr>
<tr>
<td>from</td>
<td>diversion, referred-by, to, uri, via</td>
</tr>
<tr>
<td>referred-by</td>
<td>diversion, from, to, uri, via</td>
</tr>
<tr>
<td>to</td>
<td>diversion, referred-by, from, uri, via</td>
</tr>
<tr>
<td>uri</td>
<td>diversion, referred-by, to, from, via, carrier-id</td>
</tr>
<tr>
<td>via</td>
<td>diversion, referred-by, to, from, via, carrier-id</td>
</tr>
<tr>
<td>called</td>
<td>calling, carrier-id</td>
</tr>
<tr>
<td>calling</td>
<td>called</td>
</tr>
<tr>
<td>carrier-id</td>
<td>called, uri</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(voice-class)# preference 2 calling called</td>
<td>Exits voice class configuration mode and enters global configuration mode.</td>
</tr>
</tbody>
</table>

**Step 6**  
**Example:**  
Device(voice-class)# exit  

**Step 7**  
**Example:**  
Device(voice-class)# exit  

**Step 7**  
**Example:**  
Device(voice-class)# exit  

**Step 8**  
**Example:**  
Device(config)# dial-peer voice 100 voip  
Device(config-dial-peer)# destination provision-policy 200  
Device(config)# exit  

**Step 9**  
**Example:**  
Device(config)# dial-peer voice 100 voip  
Device(config-dial-peer)# destination provision-policy 200  
Device(config)# exit  

**Step 10**  
**Example:**  
Device(config)# dial-peer voice 100 voip  
Device(config-dial-peer)# destination provision-policy 200  
Device(config)# exit  

**Step 11**  
**Example:**  
Device(config)# dial-peer voice 100 voip  
Device(config-dial-peer)# destination provision-policy 200  
Device(config)# exit  

<table>
<thead>
<tr>
<th>Provision Policy Rule Attribute</th>
<th>Dial-peer Match command</th>
</tr>
</thead>
</table>
| called                         | destination-pattern pattern  
|                                 | destination e164-pattern-map  
|                                 | pattern-map-class-id  
| calling                        | destination calling  
|                                 | e164-pattern-map  
|                                 | pattern-map-class-id  
| carrier-id                     | carrier-id target  
| uri                            | destination uri uri-class-tag  
| via                            | destination uri-via uri-class-tag  
| to                             | destination uri-to uri-class-tag  
| from                           | destination uri-from uri-class-tag  
| diversion                      | destination uri-diversion uri-class-tag  
| referred-by                    | destination uri-referred-by uri-class-tag  

Configure a match command based on any of the four attributes defined in the provision policy rule.
### Dial Peer Enhancements

#### 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 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

Step 2

dialogplan 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

dialogplan 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
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(voice-class)# description match both calling and called
Device(voice-class)# preference 2 calling called
Device(voice-class)# 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(config)# description match both calling and called
Device(config)# preference 2 via from
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

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
Configuration Example: Inbound Leg Headers for Outbound Dial-Peer Matching
CHAPTER 24

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 313
- Information About Server Groups in Outbound Dial Peers, on page 314
- How to Configure Server Groups in Outbound Dial Peers, on page 315
- Configuration Examples for Server Groups in Outbound Dial Peers, on page 319

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 42: 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>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 command is introduced under: voice class server-group, description, ipv4 port preference, ipv6 port preference, hunt-scheme, show voice class server-group, shutdown (Server Group).</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>Hunt Stop for Server Groups</td>
<td>Cisco IOS XE Bengaluru 17.4.1a</td>
<td>This feature allows you to configure hunt-stop based on (configurable) response codes in the Server Group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following command is introduced under voice class server-group. <strong>huntstop</strong> <strong>rule-tag</strong> <strong>resp-code</strong> <em>from_resp_code</em> to <em>to_resp_code</em></td>
</tr>
</tbody>
</table>

**Information About Server Groups in Outbound Dial Peers**

Server groups allow you to create simpler configurations by specifying a list of destination SIP servers for a single dial peer. When a call matches a dial peer that is configured with a server group, the destination is selected from the list of candidates based on a configured policy. If it is not possible to complete that call, the next candidate is selected. Alternatively, you can also choose to stop hunting through the group if a specified response code is received. If the call cannot be placed to any of the servers in the group, or hunting is stopped, call processing continues to the next preferred dial-peer.

You can configure server groups for SIP dial peers to include up to five IPv4 and IPv6 target server addresses listed in strict order of preference, or with equal weight for round robin or random selection.

**Note**
Whenever destination server group is used, and multiple interfaces are involved, ensure that the server group must have the session targets, belonging to the same network as that of sip bind on the dial-peer, where the server-group is configured.

If there are session targets of different network, then different dial-peers must be created with appropriate grouping of the targets with respective binding of the interfaces.

If a server-group is in the shutdown mode, all dial-peers using this destination are out of service.

**Note**
- You can use Server Groups only with SIP dial-peers.
- If a destination IP on the server group responds with codes 404, 500, or 503, the server group hunts for the next destination. But if the server group receives codes 480, 486, or 600, hunting is not supported and hence the server group does not hunt to the next destination.
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. (Optional) huntstop rule-tag resp-code from_resp_code to to_resp_code
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

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.</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>
### Command or Action

**Step 2**

`configure terminal`  
**Purpose:** Enters global configuration mode.

**Example:**

```
Device# configure terminal
```

**Step 3**

`voice class server-group server-group-id`  
**Purpose:** Configures a voice class server group and enters voice class configuration mode.

- You can use the `shutdown` command to make the server group inactive.

**Example:**

```
Step 3
Device(config)## voice class server-group 171
```

**Step 4**

```
{ipv4 | ipv6} address [port port] [preference preference-order]
```

**Purpose:** Configures a server IP address as a part of this server group along with an optional port number and preference order.

- Repeat this step to add up to five servers to the server group.
- The servers are not selected by the preference value if round robin is configured in the next step.
- Default and highest value of preference is zero.

**Example:**

```
Step 4
Device(config-class)## ipv4 10.1.1.1 preference 3
```

**Step 5**

(Optional) `hunt-scheme round-robin`  
**Purpose:** 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.

- 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 value is configured, the selection of servers is random.

**Example:**

```
Step 5
Device(config-class)## hunt-scheme round-robin
```

**Step 6**

(Optional) `description string`  
**Purpose:** Provides a description for the server group.

**Example:**

```
Step 6
Device(config-class)## description It has 3 entries
```

**Step 7**

(Optional) `huntstop rule-tag resp-code from_resp_code to to_resp_code`  
**Purpose:** Stops hunting for servers in the Server Group based on configurable response codes.

- Huntstop rule identifier tags range: 1-1000.
- Configurable SIP error response codes range: 400 to 599. The range must be in between 400 and 599 and must be entered in minimum to maximum order (Example: 450 to 460). You can enter multiple ranges using additional instances of this command. Response codes do not need to be ordered between instances (Example: huntstop 1 500 to 510 and huntstop 2 400 to 450).

**Example:**

```
Step 7
Device(config-class)## huntstop 1 resp-code 400 to 410
```
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config-class)# huntstop 2 resp-code 414</td>
<td>The following error message appears if there is an invalid input for SIP Response codes.</td>
</tr>
<tr>
<td></td>
<td>Error: Invalid SIP response code range configured for hunt-stop.</td>
</tr>
<tr>
<td></td>
<td>• Overlapping of SIP error response codes configuration is not permitted. For example, huntstop 1 resp-code 400 to 510.</td>
</tr>
<tr>
<td></td>
<td>The following error message appears if you try to configure the overlapping SIP error response codes.</td>
</tr>
<tr>
<td></td>
<td>Error: Overlap of response codes.</td>
</tr>
<tr>
<td></td>
<td>When one of the cause codes is in the configured range, the following warning message appears.</td>
</tr>
<tr>
<td></td>
<td>Example, huntstop 1 resp-code 400 to 405</td>
</tr>
<tr>
<td></td>
<td>Warning: Range includes codes that will not stop hunting.</td>
</tr>
</tbody>
</table>

**Step 8**

dial-peer voice  
dial-peer-id voip

*Example:*

Device(config)# dial-peer voice 123 voip

Defines a VoIP dial peer and enters dial peer configuration mode.

**Step 9**

session protocol sipv2

*Example:*

Device(config-dial-peer)# session protocol sipv2

Specifies SIP version 2 as the session protocol for calls between local and remote routers using the packet network.

**Step 10**

destination-pattern  
[+] string [T]

*Example:*

Device(config-dial-peer)# destination-pattern +5550179

Specifies either the prefix or the full E.164 telephone number to be used for a dial peer.

**Step 11**

session server-group  
server-group-id

*Example:*

Device(config-dial-peer)# session server-group 171

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.

**Step 12**

end

*Example:*

Device(config-dial-peer)# end

Exits dial peer configuration mode and enters privileged EXEC mode.

**Step 13**

show voice class server-group  
server-group-id

*Example:*

Displays information about the voice class server group.
Verifying Server Groups in Outbound Dial Peers

SUMMARY STEPS

1. `show voice class server-group [server-group-id]`
2. `show running-config | section server-group`

DETAILED STEPS

Step 1  `show voice class server-group [server-group-id]`

The following example displays the configurations for all configured server groups or a specified server group.

**Example:**

```
Device# show voice class server-group 1

Voice class server-group: 1
    AdminStatus: Up    OperStatus: Up
    Hunt-Scheme: preference    Last returned server:
    Description: It has 3 entries
    Total server entries: 3

    Pref  Type  IP Address     IP Port
    ----  ----  --------------  -----
    1    ipv4  10.1.1.1
    2    ipv4  10.1.1.2
    3    ipv4  10.1.1.3

    Total Huntstop tags: 2
    Tag ID  From Response code  To Response code
            ----  ------------------  ------------------
    1    404                404
    2    410                599
```

The following example displays the configurations for dial peers that are associated with server groups.

**Example:**

```
Device# show voice class server-group dialpeer 1

Voice class server-group: 1    AdminStatus: Up
    Hunt-Scheme: preference
    Total Remote Targets: 3

    Pref  Type  IP Address     IP Port
    ----  ----  --------------  -----
    1    ipv4  10.1.1.1
    2    ipv4  10.1.1.2
    3    ipv4  10.1.1.3
```
Step 2  

show running-config | section server-group

The following example displays the running configuration for server groups.

Example:

Device#show running-config | section server-group
voice class server-group 1
ipv4 10.1.1.1 preference 1
ipv4 10.1.1.2 preference 2
ipv4 10.1.1.3 preference 3
description It has 3 entries
huntstop 1 resp-code 404 to 404
huntstop 2 resp-code 410 to 599
voice class server-group 2
ipv4 10.1.1.1
ipv4 10.1.1.2
ipv4 10.1.1.3
description It has 3 entries
hunt-scheme round-robin
huntstop 1 resp-code 401 to 599

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 1
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)# huntstop 1 resp-code 404
Device(config-class)# huntstop 2 resp-code 410 to 599
Device(config-class)# exit

! Configuring an outbound SIP dial peer.
Device(config)# dial-peer voice 1 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 1
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 dialpeer 1

Voice class server-group: 1  AdminStatus: Up
Hunt-Scheme: preference
Total Remote Targets: 3
Pref  Type    IP Address    IP Port
----  ----    ----------    -------
1     ipv4    10.1.1.1
2     ipv4    10.1.1.2
3     ipv4    10.1.1.3

! Displays the configurations made for the server group.

Device# show voice class server-group 1

Voice class server-group: 1
AdminStatus: Up    OperStatus: Up
Hunt-Scheme: preference    Last returned server:
Description: It has 3 entries
Total server entries: 3

Pref  Type    IP Address    IP Port
----  ----    ----------    -------
1     ipv4    10.1.1.1
2     ipv4    10.1.1.2
3     ipv4    10.1.1.3

-------------------------------------
Total Huntstop tags: 2
Tag ID    From Response code    To Response code
------    -------------------    ------------------
1         404                 404
2         410                 599
-------------------------------------

Server Groups in Outbound Dial Peers (Round-Robin-Based Selection)

! Configuring the Server Group
Device(config)# voice class server-group 2
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)# huntstop 1 resp-code 401 to 599
Device(config-class)# description It has 3 entries
Device(config-class)# exit

! Configuring an outbound SIP dial peer.
Device(config)# dial-peer voice 2 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 2
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 dialpeer 2

Voice class server-group: 2    AdminStatus: Up
Hunt-Scheme: round-robin
Total Remote Targets: 3
Device# show voice class server-group 2

Voice class server-group: 2
AdminStatus: Up OperStatus: Up
Hunt-Scheme: round-robin Last returned server: 10.1.1.2
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>10.1.1.3</td>
<td></td>
</tr>
<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>
</tbody>
</table>

Total Huntstop tags: 1
Tag ID From Response code To Response code
------ ------------------- ------------------
1 401 599

Dial Peer Enhancements
Configuration Examples for Server Groups in Outbound Dial Peers
Configuration Examples for Server Groups in Outbound Dial Peers
CHAPTER 25

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 323
- Restrictions for Domain-Based Routing Support on the Cisco UBE, on page 324
- Information About Domain-Based Routing Support on the Cisco UBE, on page 324
- How to Configure Domain-Based Routing Support on the Cisco UBE, on page 325
- Configuration Examples for Domain-Based Routing Support on the Cisco UBE, on page 330

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 43: 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>
</table>
| Domain Based Routing Support on the Cisco UBE     | 15.2(1)T                          | 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`. |
| Domain Based Routing Support on the Cisco UBE     | Cisco IOS XE Release 3.8S         | 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`. |

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.
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
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>
</tbody>
</table>

| Step 2 | configure terminal | Enters global configuration mode. |
| Example: Device# configure terminal | |

| Step 3 | voice service voip | Enters voice service configuration mode. |
| Example: Device(config)# voice service voip | |

| Step 4 | sip | Enters voice service SIP configuration mode. |
| Example: Device(config-vol-serv)# sip | |

| Step 5 | call-route url | Routes calls based on the URL. |
| Example: Device(config-vol-serv)# call-route url | |

| Step 6 | exit | Exits the current mode. |
| Example: Device(config-vol-serv)# 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>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>
### Dial Peer Enhancements

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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 dial-peer tag voip</td>
<td>Enter dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# dial-peer voice 2 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class sip call-route url</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)#</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Routes calls based on the URL</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> 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:
INVITE sip:5555555555@[2208:1:1:1:1:1:1118]:5060 SIP/2.0
Via: SIP/2.0/UDP [2208:1:1:1:1:1115]:5060;branch=z9hG4bK83AE3
s=SIP Call
m=IN IP6 2208:1:1:1:1:1:1:1116
m=image 17278 udptl t38

Step 3  debug voip dialpeer inout

The debug ccsip 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

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:

1: Dial-peer Tag=3600
2: Dial-peer Tag=36

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
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 331
• Restrictions for ENUM Enhancement per Kaplan Draft RFC, on page 332
• Information About ENUM Enhancement per Kaplan Draft RFC, on page 333
• How to Configure ENUM Enhancement per Kaplan Draft RFC, on page 333
• Troubleshooting Tips, on page 336
• Configuration Examples for ENUM Enhancement per Kaplan Draft RFC, on page 336

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 44: 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 \texttt{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 \texttt{voice enum-match-table}, including the options, needs to be specified whenever being referenced by its subcommand. If not, the defaults, \texttt{no source-uri} and \texttt{no no-cached} (or caching) will take effect.
- As the maximum number of characters of the host shown in the \texttt{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 \texttt{show} command outputs in this chapter.
- If \texttt{no-cache} is configured in the \texttt{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 \texttt{source-uri} is configured for the outbound \texttt{enum-match-table}.
- The route will not be cached if the server does not support the OPT RR (it is recommended to remove the \texttt{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 \texttt{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 \texttt{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></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</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> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></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><code>voice enum-match-table match-table-index [source-uri] [no-cache]</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# voice enum-match-table 5 source-uri no-cache</td>
<td></td>
</tr>
</tbody>
</table>
### 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></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>Device&gt; enable</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Tests the source-based routing ENUM.</td>
</tr>
<tr>
<td><code>test enum match-table-index input -pattern source-url source-url more parameter</code></td>
<td>• The source routing or no caching features depend on the <code>voice enum-match-table</code> command. If the <code>source-uri</code> command is not configured, the <code>source-url source-url</code> in the test command is ignored.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device# test enum 1117777 source sip:1116666@10.1.50.16 more &quot;ibcall-id=1-23735@10.1.150.16; obcall-id=7190DF-F1AA3CF1@10.1.110.222;sbc-id=1</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><code>end</code></td>
<td>Device# end</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*"

**DETAILED 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: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: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: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: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 inout`

- `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 pseudo 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$// /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$// /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 341
• Configuring Multi-Tenants on SIP Trunks, on page 395
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.

- Feature Information for VRF, on page 341
- Information About Voice-VRF, on page 343
- Information About Multi-VRF, on page 343
- VRF Preference Order, on page 344
- Restrictions, on page 344
- Recommendations, on page 345
- Configuring VRF, on page 345
- Configure VRF-Specific RTP Port Ranges, on page 351
- Directory Number (DN) Overlap across Multiple-VRFs, on page 354
- IP Overlap with VRF, on page 356
- Using Server Groups with VRF, on page 358
- Inbound Dial-Peer Matching Based on Multi-VRF, on page 359
- VRF Aware DNS for SIP Calls, on page 361
- High Availability with VRF, on page 362
- Configuration Examples, on page 362
- Troubleshooting Tips, on page 393

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.

### Table 45: Feature Information for VRF

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Voice-VRF (VRF-Aware)</td>
<td>Cisco IOS 12.4(11)XJ</td>
<td>This feature provides support to configure a VRF specific to voice traffic.</td>
</tr>
<tr>
<td>Support for Multi-VRF</td>
<td>Cisco IOS 15.6(2)T</td>
<td>This feature allows CUBE to have multiple instances of VRF to co-exist on the same device at the same time. The following commands are introduced: <code>media-address voice-vrf name port-range min-max, show voice vrf</code></td>
</tr>
<tr>
<td>Enhancement to support up to 54 VRF instances</td>
<td>Cisco IOS 15.6(3)M, Cisco IOS XE Denali 16.3.1</td>
<td>This feature enhancement provides support for up to 54 VRFs. Each of the VRFs supports up to 10 different RTP port ranges.</td>
</tr>
<tr>
<td>Support for Inbound Dial-peer Matching using VRF-ID</td>
<td>Cisco IOS 15.6(3)M, Cisco IOS XE Denali 16.3.1</td>
<td>This feature supports inbound dial-peer matching using VRF ID.</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>
| Support for media flow-around using Multi-VRF | Cisco IOS XE Gibraltar 16.12.2 | This feature adds media flow-around support for the following intra-VRF call flows in standalone and high availability scenarios:  
  - Basic Audio Call  
  - Call Hold and Resume  
  - Re-INVITE based Call Transfer  
  - 302 based Call Forward  
  - Fax Pass Through Calls  
  - T.38 Fax Calls  
  With media flow-around using Multi-VRF, only signalling is routed using VRFs and CUBE passes across the media IP and ports which it receives. For detailed information on media flow-around, see Media Path. |
| Support up to 100 VRF instances | Cisco IOS XE Amsterdam 17.3.1a | This feature enhancement provides support up to 100 VRFs. Each of the VRFs supports up to 10 different RTP port ranges. |

**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.
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 coexist. The following is the binding preference order for call processing:

<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 CUBE 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 CUBE is collocated with Cisco Unified Survivability Remote Site Telephony.</td>
</tr>
</tbody>
</table>

**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 on releases before Cisco IOS XE Fuji 16.7.1.
- IPv6 on VRF is not supported.
- SDP pass-through is not supported on releases before Cisco IOS Release 15.6(3)M and Cisco 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 through a dial-peer is not supported with the Multi-VRF.
• 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.
• Multi-VRF configured in media flow-around mode is supported only for intra-VRF calls. The following are not supported with Multi-VRF configured in media flow-around mode:
  • Supplementary services with REFER Consume, Mid-call (or Early Dialogue) block
  • Session Description Protocol (SDP) Passthrough
  • Media Recording
  • DSP flows (DTMF, transcode)

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.
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>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>enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td>creates a VRF with the specified name. In the example, VRF name is VRF1.</td>
</tr>
<tr>
<td><code>ip vrf vrf-name</code></td>
<td>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)</td>
</tr>
<tr>
<td><code>rd route-distinguisher</code></td>
<td>exits present mode.</td>
</tr>
</tbody>
</table>

Note: Space is not allowed in VRF name.
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>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> interface interface-name</td>
<td>Enters the interface configuration mode.</td>
</tr>
<tr>
<td>Example: 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>Example: Device(config-if)# ip vrf forwarding VRF1</td>
<td><strong>Note</strong> If there is an IP address associated with the interface, it will be cleared and you will be prompted to assign the IP address again.</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>Example: Device(config-if)# ip address 10.0.0.1 255.255.255.0</td>
<td></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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>enable</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; enable</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>configure terminal</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>dial-peer voice number voip</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# dial-peer voice 1111 voip</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>session protocol protocol</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th><strong>Step 5</strong></th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Create dial-peer:  
  - To create inbound dial-peer:  
    * incoming called number *number*  
  - To create outbound dial-peer:  
    * destination pattern *number* | Creates inbound and outbound dial-peer. |

**Example:**

Inbound dial-peer:

```
Device(config-dial-peer)# incoming called-number 1111
```

Outbound dial-peer:

```
Device(config-dial-peer)# destination pattern 3333
```

<table>
<thead>
<tr>
<th><strong>Step 6</strong></th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>codec codec-name</em></td>
<td>Specifies the codec associated with this dial-peer.</td>
</tr>
</tbody>
</table>

**Example:**

```
Device(config-dial-peer)# codec g711ulaw
```

<table>
<thead>
<tr>
<th><strong>Step 7</strong></th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>exit</em></td>
<td>Exits present mode.</td>
</tr>
</tbody>
</table>

**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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enabled</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> Bind control and media to the interface</td>
<td>Interface bind associates VRF to the specified dial-peer.</td>
</tr>
<tr>
<td>• At dial-peer level:</td>
<td></td>
</tr>
</tbody>
</table>
|   ```
   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:** | | |
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><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode</td>
</tr>
<tr>
<td><code>enable</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Entrys your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; <code>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>Device# <code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters <code>voice service voip</code> mode.</td>
</tr>
<tr>
<td><code>voice service voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# <code>voice service voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Associates the RTP Port range with the VRF.</td>
</tr>
<tr>
<td><code>media-address voice-vrf vrf-name port-range min max</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>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 <strong>8000-48198</strong> for ASR and ISR G3 platforms, and <strong>16384-32766</strong> for Cisco ISR G2 platforms.</td>
</tr>
<tr>
<td>conf-voi-serv)# <code>media-address voice-vrf vrf1-Dec-15 port-range 9112 9118</code></td>
<td></td>
</tr>
</tbody>
</table>
| **Example:** | **Note**  
- The port range must be configured on the same line as the media address. The port ranges configured using a second line on outgoing dial peer are not supported.  
- From Cisco IOS XE Amsterdam 17.3.1a onwards, you can configure 100 VRFs for up to 10 different RTP port ranges (that is, 10 different port ranges per each VRF). |
| conf-voi-serv)# `media-address voice-vrf VRF1 port 16000 32000` |  
| The output: | |
| Device# `show run` |  
| `section voice` | |
| voice-card 0/3 |  
| dsp services dspfarm |  
| voice service voip |  
| no ip address trusted authenticate |  
| media-address voice-vrf VRF1 port 16000 32000 |  
| *Here, the port-range is configured on the same line as the media address. |
| **Example:** | |
| Example 1 | |
| Device(conf-voi-serv)# `media-address voice-vrf VRF1 port 16000 32000` | |
| **Example:** | |
| Example 2 | CUBE supports up to 100 VRFs. Hence, you can configure up to 100 media address instances, that is, one instance per `voice-vrf`. This configuration is subject to the maximum number of VRFs supported by the host platform. |
| Device(conf-voi-serv)# | |
| `media-address voice-vrf VRF1 port-range 8000 48000` | |
### Example: VRF with overlapping and non-overlapping RTP Port Range

#### Example 1 - Non-overlapping Port Range

The following is example shows two VRFs with non-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 29000 32000
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

                Min    Max    Ports    Ports
Ports Media-Address Range       Port    Port Available Reserved In-use
--------------------------------------
Global Media Pool 8000 48198 19999 101  0
VRF ID Based Media Pool
vrf1 25000 28000 1501 0 1
vrf2 29000 32000 1501 0 1

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 example 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>Media-Address Range</th>
<th>Port</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>
<td></td>
</tr>
<tr>
<td>VRF ID Based Media Pool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vrf1</td>
<td>25000 28000</td>
<td>1501</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vrf2</td>
<td>25000 28000</td>
<td>1501</td>
<td>0</td>
<td>1</td>
<td></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>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>
</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>
</tr>
</tbody>
</table>

Found 2 active RTP connections
```

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 31: 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 VRF1 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(voice-class)# 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
```
Creating outbound dial-peer with destination pattern ‘3001’ associated with VRF1.

Device(config)# dial-peer voice 300 voip
Device(config)# destination-pattern 3001
Device(config)# video codec h264
Device(config)# session protocol sipv2
Device(config)# session target ipv4:10.0.0.1
Device(config)# voice-class sip bind control source-interface GigabitEthernet0/0
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/0
Device(config)# dtmf-relay sip-kpml
Device(config)# codec g711ulaw

Creating outbound dial-peer with destination pattern ‘3001’ associated with VRF2.

Device(config)# dial-peer voice 301 voip
Device(config)# destination-pattern 3001
Device(config)# video codec h264
Device(config)# session protocol sipv2
Device(config)# session target ipv4:11.0.0.1
Device(config)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/2
Device(config)# dtmf-relay sip-kpml
Device(config)# 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, 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>FER</th>
<th>PASS</th>
<th>THRU</th>
<th>SESS-TARGET</th>
<th>STAT</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td></td>
<td>3001</td>
<td>0</td>
<td>syst</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td></td>
<td>3001</td>
<td>0</td>
<td>syst</td>
<td>ipv4: 10.0.0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>301</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td></td>
<td>3001</td>
<td>0</td>
<td>syst</td>
<td>ipv4: 11.0.0.1</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: The above configurations are specific to VRF support only. For call routing, appropriate routing protocols must be configured in the network.

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</th>
<th>Status</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded-Service-Engine0/0</td>
<td>unassigned</td>
<td>YES NVRAM</td>
<td>administratively down</td>
<td>down</td>
</tr>
<tr>
<td>GigabitEthernet0/0</td>
<td>10.0.0.0</td>
<td>YES NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/1</td>
<td>10.0.0.0</td>
<td>YES NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/1.1</td>
<td>unassigned</td>
<td>YES NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/2</td>
<td>unassigned</td>
<td>YES NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
</tbody>
</table>
**show voip rtp connections** command shows a video call that is established on CUBE across different interfaces belonging to different VRFs having Overlap IP address:

Device# **show voip rtp connections**

VoIP RTP Port Usage Information:
Max Ports Available: 11700, Ports Reserved: 303, Ports in Use: 4

<table>
<thead>
<tr>
<th>Min</th>
<th>Max</th>
<th>Ports Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>22000</td>
<td>900</td>
<td>101</td>
<td>0</td>
</tr>
</tbody>
</table>

VRF ID Based Media Pool

<table>
<thead>
<tr>
<th>Media-Address Range</th>
<th>Min</th>
<th>Max</th>
<th>Ports Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media Pool</td>
<td>20000</td>
<td>22000</td>
<td>900</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>VRF 1</td>
<td>30002</td>
<td>32000</td>
<td>1000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VRF 2</td>
<td>20000</td>
<td>30000</td>
<td>4900</td>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>VRF 3</td>
<td>20000</td>
<td>30000</td>
<td>4900</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>37</td>
<td>39</td>
<td>20000</td>
<td>18164</td>
<td>10.0.0.0.0</td>
<td>11.0.0.3</td>
<td>NO</td>
<td>VRF1</td>
</tr>
<tr>
<td>2</td>
<td>38</td>
<td>40</td>
<td>20002</td>
<td>18166</td>
<td>10.0.0.0.0</td>
<td>11.0.0.3</td>
<td>NO</td>
<td>VRF1</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>37</td>
<td>20002</td>
<td>16388</td>
<td>10.0.0.0.0</td>
<td>11.0.0.3</td>
<td>NO</td>
<td>VRF2</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>38</td>
<td>20000</td>
<td>16390</td>
<td>10.0.0.0.0</td>
<td>11.0.0.3</td>
<td>NO</td>
<td>VRF2</td>
</tr>
</tbody>
</table>

Found 4 active RTP connections

**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:

Interfaces and associated IP address

- GigabitEthernet0/0/2 12.0.0.1
- GigabitEthernet0/0/1 11.0.0.1

Device# **show ip interface brief**

<table>
<thead>
<tr>
<th>Interface</th>
<th>IP-Address</th>
<th>OK?</th>
<th>Method</th>
<th>Status</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigabitEthernet0/0/0</td>
<td>10.0.0.0/32</td>
<td>Y</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/0/1</td>
<td>11.0.0.1/32</td>
<td>Y</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
<tr>
<td>GigabitEthernet0/0/2</td>
<td>12.0.0.1/32</td>
<td>Y</td>
<td>NVRAM</td>
<td>up</td>
<td>up</td>
</tr>
</tbody>
</table>

- 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:

```
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
```
codecs 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.

---

**Note**
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 check pointed 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 32: 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

```
```
Configure Interface GigabitEthernet0/1

Device(config)# enable
Device(config)# 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>PASS</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>
<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>
<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
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>RD</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mgmt-intf</td>
<td>&lt;not set&gt;</td>
<td></td>
<td>Gi0</td>
</tr>
<tr>
<td>VRF1</td>
<td>1:1</td>
<td></td>
<td>Gi0/1</td>
</tr>
<tr>
<td>VRF2</td>
<td>2:2</td>
<td></td>
<td>Gi0/2</td>
</tr>
</tbody>
</table>

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:**


```
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>MFSS</th>
<th>VRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>25000</td>
<td>16390</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>2</td>
<td>1</td>
<td>25002</td>
<td>16398</td>
<td>11.0.0.1</td>
<td>11.0.0.2</td>
<td>NO</td>
<td>VRF2</td>
</tr>
</tbody>
</table>
Device# show call active voice brief

Perf-AR1006# show call active voice brief
<ID>: <CallID> <start> <index> <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:<late>/min>/<max>ms <code> <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:<%>
VRF:<%
MODEMPASS <method> buf:<fills>/drains> loss <overall%> <multipkt>/<corrected>
last <buf event time>s dur:<Min>/Max>
FR <protocol> [int dlcid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
ATM <protocol> [int vpl/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
MODEMRELAY info:<rcvd>/sent/retrans> xid:<rcvd>/sent total:<rcvd>/sent/<drops>
speeds(bps): local <rx>/<tx> remote <rx>/<tx>
Proxy <ip>:[audio udp],<video udp>,<tcp0>,<tcp1>,<tcp2>,<tcp3> endpt: <type>/<manf>
bw: <req>/<act> codec: <audio>/<video>
x: <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: 0
Multicast call-legs: 0
Total call-legs: 2

Answer 777512373 active
dur 00:00:22 tx:1110/6660 rx:1111/6661 dscp:0 media:0 audio tos:0xB8 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 long duration call duration:n/a timestamp:n/a
LostPacketRate:0.00 OutOfOrderRate:0.00
VRF: VRF1

Answer 777512373 active
dur 00:00:22 tx:1110/6660 rx:1111/6661 dscp:0 media:0 audio tos:0xB8 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 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
Answer 777512373 active
dur 00:00:22 tx:1111/6660 rx:1110/6661 dscp:0 media:0 audio tos:0xB8 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 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
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2
Example: Configuring RG Infra High Availability with VRF

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.

Figure 33: Multi-VRF in High Availability Mode (RG Infra)
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
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

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

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.
Example: Configuring RG Infra High Availability with VRF

```
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

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-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.

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 10.0.0.4 255.0.0.0
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.

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:

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-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/2

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>Min</th>
<th>Max</th>
<th>Ports</th>
<th>Ports</th>
<th>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.</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>10.0.0.2</td>
<td>NO</td>
<td>VRF1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>8010</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>Min</th>
<th>Max</th>
<th>Ports</th>
<th>Ports</th>
<th>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.</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>10.0.0.2</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

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:0xB8 video tos:0x0
IP 10.0.0.2:16388 SRTF: off rtt:1ms 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

active
dur 00:00:14 tx:1028/61680 rx:843/50551 dscp:0 media:0 audio tos:0xB8 video tos:0x0
IP 11.0.0.2:16388 SRTF: off rtt:65522ms 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
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 ---------------
Conn-Id Local-Address
=========== =============================
2 [10.0.0.1]:5060:VRF1
3 [11.0.0.1]:5060:VRF2

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 SRTP: off rtt:65531ms p1:0/0ms 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 SRTP: off rtt:65523ms p1:0/0ms 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

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

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 34: 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
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
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.

If interface GigabitEthernet0/0/0 is used for configuring HSRP and therefore no VRF is associated with this interface.

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.

---

**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 14.2.43.81 255.255.0.0
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-dial-peer)# 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-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

Note 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
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)# 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 14.2.43.82
Device(config-ipc-local-sctp)# exit
Device(config-ipc-protocol-sctp)# remote port 5000
Device(config-ipc-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

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 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>
<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>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>8010</td>
<td>16388</td>
<td>11.0.0.1</td>
<td>11.0.0.2</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>
Media-Address Range | Port | Port | Available | Reserved | In-use
---------------------|------|------|-----------|----------|-------
Global Media Pool    | 8000 | 48198| 19999     | 101      | 2     
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>7</td>
<td>8</td>
<td>8012</td>
<td>16390</td>
<td>10.0.0.1</td>
<td>10.0.0.2</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>
</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 audio tos:0xB8 video tos:0x0
IP 10.0.0.2:16388 SRTPT: off rtt:1ms 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

active
dur 00:00:14 tx:1028/61680 rx:843/50551 dscp:0 audio tos:0xB8 video tos:0x0
IP 11.0.0.2:16388 SRTPT: off rtt:65522ms 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

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:
Example: Configuring Multi VRF where Media Flows Around the CUBE

The configuration in this scenario is as shown below where there is overlapping endpoint IP address across two customers and use CUBE for inter-enterprise calls. Here the media flows around the CUBE for the enterprises with Multi-VRF feature and both the enterprises have the same endpoint IP address.

*Figure 35: Multi-VRF with Media Flow Around CUBE*
Set-up Information

- Two enterprises ENT2 and ENT3 have the same endpoint IP address.
- Provider Edge (PE) router acts as DHCP for both enterprises.
- PSTN call flow is simulated with the Emulation Call Manager.
- When a call is initiated from ENT2 to ENT3, the call is a flow around call and both the endpoints are connected directly.

Configuration Information

The table below details the configuration information required to configure Multi-VRF, where the media (call) flows around the CUBE.

<table>
<thead>
<tr>
<th>Name</th>
<th>ENT2 Configuration</th>
<th>ENT3 Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PE VRF Configuration</strong></td>
<td>ENT2 PE VRF Configuration</td>
<td>ENT3 PE VRF Configuration</td>
</tr>
<tr>
<td>DHCP Configuration</td>
<td>ip dhcp pool DC30-ENT302</td>
<td>ip dhcp pool DC30-ENT303</td>
</tr>
<tr>
<td></td>
<td>vrf Ent302</td>
<td>vrf Ent303</td>
</tr>
<tr>
<td></td>
<td>network 30.18.2.0 255.255.255.0</td>
<td>network 30.18.2.0 255.255.255.0</td>
</tr>
<tr>
<td></td>
<td>domain-name hcsent2.ciscolabs.com</td>
<td>domain-name hcsent3.ciscolabs.com</td>
</tr>
<tr>
<td></td>
<td>option 150 ip 200.1.1.10</td>
<td>option 150 ip 200.1.1.10</td>
</tr>
<tr>
<td></td>
<td>default-router 30.18.2.1</td>
<td>default-router 30.18.2.1</td>
</tr>
<tr>
<td></td>
<td>dns-server 200.1.1.59</td>
<td>dns-server 200.1.1.59</td>
</tr>
<tr>
<td><strong>CUBE VRF Configuration</strong></td>
<td>ENT2 - CUBE VRF Configuration</td>
<td>ENT3 - CUBE VRF Configuration</td>
</tr>
<tr>
<td><strong>CUBE Voice Class URI Configuration</strong></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>voice class uri 2112 sip</td>
<td>voice class uri 3112 sip</td>
</tr>
<tr>
<td></td>
<td>pattern 200.1.1.10:8012</td>
<td>pattern 200.1.1.10:8013</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>voice class uri 2114 sip</td>
<td>voice class uri 3114 sip</td>
</tr>
<tr>
<td></td>
<td>pattern 172.16.30.62:8012</td>
<td>pattern 172.16.30.62:8013</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td>Name</td>
<td>ENT2 Configuration</td>
<td>ENT3 Configuration</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CUBE DPG Configuration</strong></td>
<td>voice class dpg 2012</td>
<td>voice class dpg 3012</td>
</tr>
<tr>
<td></td>
<td>dial-peer 2012</td>
<td>dial-peer 3012</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>voice class dpg 2014</td>
<td>voice class dpg 3014</td>
</tr>
<tr>
<td></td>
<td>dial-peer 2014</td>
<td>dial-peer 3014</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td><strong>CUBE COR Member Configuration</strong></td>
<td>dial-peer cor custom</td>
<td>dial-peer cor custom</td>
</tr>
<tr>
<td></td>
<td>name Ent3102</td>
<td>name Ent3103</td>
</tr>
<tr>
<td></td>
<td>name PGW-Ent3102</td>
<td>name PGW-Ent3103</td>
</tr>
<tr>
<td><strong>CUBE COR List Configuration</strong></td>
<td>dial-peer cor list From-Ent3102</td>
<td>dial-peer cor list From-Ent3103</td>
</tr>
<tr>
<td></td>
<td>member Ent3102</td>
<td>member Ent3103</td>
</tr>
<tr>
<td></td>
<td>member PGW-Ent3102</td>
<td>member PGW-Ent3103</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>dial-peer cor list To-Ent3102</td>
<td>dial-peer cor list To-Ent3103</td>
</tr>
<tr>
<td></td>
<td>member Ent3102</td>
<td>member Ent3103</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>dial-peer cor list From-PGW-Ent3102</td>
<td>dial-peer cor list From-PGW-Ent3103</td>
</tr>
<tr>
<td></td>
<td>member Ent3102</td>
<td>member Ent3103</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>dial-peer cor list To-PGW-Ent3102</td>
<td>dial-peer cor list To-PGW-Ent3103</td>
</tr>
<tr>
<td></td>
<td>member PGW-Ent3102</td>
<td>member PGW-Ent3103</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td><strong>Dial Peer Configuration</strong></td>
<td>ENT2 - Dial Peer Configuration</td>
<td>ENT3 - Dial Peer Configuration</td>
</tr>
</tbody>
</table>

**PE VRF Configuration - ENT2**

```
BLR-PE-BGL18-NEW#sh run vrf Ent302
Building configuration...
Current configuration : 1072 bytes
ip vrf Ent302
description Enterprise 3102 VRF
d rd 3102:1
route-target export 3102:1
route-target import 3102:1
route-target import 110:1
!
interface GigabitEthernet0/0/0
description Link to HCS-BGL18-CUBE(CUBE-ENT)
ip address 192.168.18.9 255.255.255.252
ip ospf network point-to-point
```
logging event link-status
load-interval 30
negotiation auto
mpls bgp forwarding
cdp enable
!
interface Port-channel1
  no ip address
  no negotiation auto
!
interface Port-channel1.302
  encapsulation dot1Q 302
  ip vrf forwarding Ent302
  ip address 30.18.2.1 255.255.255.0
!
router bgp 65535
!
  address-family ipv4 vrf Ent302
  redistribute connected
  exit-address-family

PE VRF Configuration - ENT3

BLR-PE-BGL18-NEW#sh run vrf Ent303
Building configuration...
Current configuration : 850 bytes
ip vrf Ent303
  description Enterprise 3103 VRF
  rd 3103:1
  route-target export 3103:1
  route-target import 3103:1
  route-target import 110:1
!
interface GigabitEthernet0/0/0
  description Link to HCS-BGL18-CUBE(CUBE-ENT)
  ip address 192.168.18.9 255.255.255.252
  ip ospf network point-to-point
  logging event link-status
  load-interval 30
  negotiation auto
  mpls bgp forwarding
cdp enable
!
interface Port-channel1
  no ip address
  no negotiation auto
!
interface Port-channel1.303
  encapsulation dot1Q 303
  ip vrf forwarding Ent303
  ip address 30.18.2.1 255.255.255.0
!
router bgp 65535
!
  address-family ipv4 vrf Ent303
  redistribute connected
  exit-address-family

end

CUBE VRF Configuration - ENT2
Example: Configuring Multi VRF where Media Flows Around the CUBE

DC30-BGL18-CUBE#sh run vrf Ent302
Building configuration...
Current configuration : 616 bytes
ip vrf Ent302
description Enterprise 3102 VRF
rd 3102:1
route-target export 3102:1
route-target import 3102:1
!
interface GigabitEthernet0/0/0
description Link to HCS-BGL18-PE1 from CUBE-VRF
ip address 192.168.18.10 255.255.255.252
ip ospf network point-to-point
load-interval 30
media-type rj45
negotiation auto
!
interface GigabitEthernet0/0/0.302
encapsulation dot1Q 302
ip vrf forwarding Ent302
ip address 172.131.2.21 255.255.255.252
ip ospf network point-to-point
!
router ospf 302 vrf Ent302
network 172.131.2.20 0.0.0.3 area 0.0.0.0
!
ip route vrf Ent302 0.0.0.0 0.0.0.0 172.131.2.22
end

CUBE VRF Configuration - ENT3

DC30-BGL18-CUBE#sh run vrf Ent303
Building configuration...
Current configuration : 616 bytes
ip vrf Ent303
description Enterprise 3103 VRF
rd 3103:1
route-target export 3103:1
route-target import 3103:1
!
interface GigabitEthernet0/0/0
description Link to HCS-BGL18-PE1 from CUBE-VRF
ip address 192.168.18.10 255.255.255.252
ip ospf network point-to-point
load-interval 30
media-type rj45
negotiation auto
!
interface GigabitEthernet0/0/0.303
encapsulation dot1Q 303
ip vrf forwarding Ent303
ip address 172.131.3.21 255.255.255.252
ip ospf network point-to-point
!
router ospf 303 vrf Ent303
network 172.131.3.20 0.0.0.3 area 0.0.0.0
!
ip route vrf Ent303 0.0.0.0 0.0.0.0 172.131.3.22
end

Dial Peer Configuration - ENT2

dial-peer voice 2011 voip
corlist incoming From=Ent3102
description Inbound Trunk from BT-Ent302 CUCM
Dial Peer Configuration - ENT3

dial-peer voice 3011 voip
  corlist incoming From-Ent3103

Example: Configuring Multi VRF where Media Flows Around the CUBE
description Inbound Trunk from BT-Ent303 CUCM
session protocol sipv2
destination dpg 3014
incoming uri via 3112
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface
GigabitEthernet0/0/0.303
voice-class sip bind media source-interface
GigabitEthernet0/0/0.303

! dial-peer voice 3012
voipcorlist outgoing To-Ent3103
description *** Outbound Trunk to BT-Ent303 DN Reouting ****
destination-pattern 8115
session protocol sipv2
session target ipv4:200.1.1.10:8013
session transport tcp
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface
GigabitEthernet0/0/0.303
voice-class sip bind media source-interface
GigabitEthernet0/0/0.303

! dial-peer voice 3013 voip
voipcorlist incoming From-PGW-Ent3103
description Inbound Trunk from PGW-Ent3103
session protocol sipv2
session transport tcp
destination dpg 3012
incoming uri via 3114
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface
GigabitEthernet0/0/0.3030
voice-class sip bind media source-interface
GigabitEthernet0/0/0.3030

! dial-peer voice 3014 voip
voipcorlist outgoing To-PGW-Ent3103
description Outbound Trunk to PGW-BT-Ent303
destination-pattern 8115
session protocol sipv2
session target ipv4:172.16.30.62:8013
session transport tcp
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface
GigabitEthernet0/0/0.3030
voice-class sip bind media source-interface
GigabitEthernet0/0/0.3030

!
Debug Information

Note  Execute `sh sip-ua calls called-number +14089135001`, to check the call behaviour.
Example: Configuring Multi VRF where Media Flows Through the CUBE

The configuration in this scenario is as shown below where there is overlapping endpoint IP address across two customers and use CUBE for inter-enterprise calls. Here the media is flowing through the CUBE for the enterprises with Multi VRF feature and both the enterprises having same end-point IP address.

Figure 37: Multi-VRF with Media Flow Through CUBE

Set-up Information

- ENT5 and ENT6 have same endpoint IP addressing, and ENT5 act as PSTN customer.
- DC30-ENT5-CE and BGL-18-PE act as DHCP servers for ENT5 and ENT6 endpoints respectively.
- When a call is initiated from ENT5 to ENT6, the call is connected between two endpoints, and signalling and media flows through the CUBE.
- Customer Edge (CE) router will directly communicate with CUBE-Enterprises' Public IP address.

Configuration Information

The table below details the configuration information required to configure Multi-VRF, where the media (call) flows through the CUBE.
### Table 47: Configuration Information

<table>
<thead>
<tr>
<th>Name</th>
<th>CE CUBE Configuration</th>
<th>SP CUBE Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DHCP Configuration</strong></td>
<td>ip dhcp pool DC30-ENT305</td>
<td>ip dhcp pool DC30-ENT306</td>
</tr>
<tr>
<td></td>
<td>network 30.18.5.0 255.255.255.0</td>
<td>vrf Ent306</td>
</tr>
<tr>
<td></td>
<td>domain-name hcsent5.ciscolabs.com</td>
<td>network 30.18.5.0 255.255.255.0</td>
</tr>
<tr>
<td></td>
<td>option 150 ip 200.1.1.10</td>
<td>domain-name hcsent6.ciscolabs.com</td>
</tr>
<tr>
<td></td>
<td>default-router 30.18.5.1</td>
<td>option 150 ip 200.1.1.10</td>
</tr>
<tr>
<td></td>
<td>dns-server 200.1.1.59</td>
<td>default-router 30.18.5.1</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>dns-server 200.1.1.59</td>
</tr>
<tr>
<td><strong>Voice Class URI Configuration</strong></td>
<td>! voice class uri 5112 sip pattern 200.1.1.10:8015</td>
<td>! voice class uri 102252 sip pattern 10.225.104.195:5060</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>voice class uri 5114 sip pattern 10.225.104.192</td>
<td>voice class uri 102254 sip pattern 200.1.1.10:7016</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td><strong>DPG Configuration</strong></td>
<td>! voice class dpg 5012 dial-peer 5012</td>
<td>! voice class dpg 9992 dial-peer 1022502</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>voice class dpg 5014 dial-peer 5014</td>
<td>voice class dpg 9994 dial-peer 1022504</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td><strong>COR Member Configuration</strong></td>
<td>dial-peer cor custom name Ent3105</td>
<td>dial-peer cor custom name BGL-Ent3105</td>
</tr>
<tr>
<td></td>
<td>name PGW-Ent3105</td>
<td>name PGW-Ent3105</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
</tbody>
</table>
### COR List Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>CE CUBE Configuration</th>
<th>SP CUBE Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>dial-peer cor list From-Ent3105</td>
<td>dial-peer cor list From-BGL-Ent3105</td>
</tr>
<tr>
<td></td>
<td>member Ent3105</td>
<td>member BGL-Ent3105</td>
</tr>
<tr>
<td></td>
<td>member PGW-Ent3105</td>
<td>member RCDN-Ent3106</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>dial-peer cor list To-Ent3105</td>
<td>dial-peer cor list To-BGL-Ent3105</td>
</tr>
<tr>
<td></td>
<td>member Ent3105</td>
<td>member BGL-Ent3105</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>dial-peer cor list From-PGW-Ent3105</td>
<td>dial-peer cor list From-RCDN-Ent3106</td>
</tr>
<tr>
<td></td>
<td>member Ent3105</td>
<td>member BGL-Ent3105</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>dial-peer cor list To-PGW-Ent3105</td>
<td>dial-peer cor list To-RCDN-Ent3106</td>
</tr>
<tr>
<td></td>
<td>member PGW-Ent3105</td>
<td>member RCDN-Ent3106</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
</tbody>
</table>

### Dial-Peer Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>CE - Dial Peer Configuration</th>
<th>SP - Dial Peer Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>interface GigabitEthernet0/0/0</td>
<td>interface GigabitEthernet0/0/1</td>
</tr>
<tr>
<td></td>
<td>ip address 10.225.104.195 255.255.255.0</td>
<td>ip address 10.225.104.192 255.255.255.0</td>
</tr>
<tr>
<td></td>
<td>negotiation auto</td>
<td>negotiation auto</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>!</td>
</tr>
</tbody>
</table>

### Dial Peer Configuration - CE

```plaintext
! 
dial-peer voice 5011 voip
corlist incoming From-Ent3105
description Inbound Trunk from Ent3105 CUCM
session protocol sipv2
destination dpg 5014
incoming uri via 5112
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/2
voice-class sip bind media source-interface GigabitEthernet0/0/2

! 
dial-peer voice 5012 voip
corlist outgoing To-Ent3105
```
description *** Outbound Trunk to BT-Ent304 DN
    Reouting ****
destination-pattern 8115
session protocol sipv2
session target ipv4:200.1.1.10:8015
session transport tcp
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/2
voice-class sip bind media source-interface GigabitEthernet0/0/2
!
dial-peer voice 5013 voip
corlist incoming From-PGW-Ent3105
description Inbound Trunk from PGW-Ent3105
session protocol sipv2
session transport tcp
destination dpg 5012
incoming uri via 5114
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/0
voice-class sip bind media source-interface GigabitEthernet0/0/0
!
dial-peer voice 5014 voip
corlist outgoing To-PGW-Ent3105
description Outbound Trunk to PGW-Ent3105
destination-pattern 8115
session protocol sipv2
session target ipv4:10.225.104.192
session transport tcp
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/0
voice-class sip bind media source-interface GigabitEthernet0/0/0
!
Dial Peer Configuration - SP
!
dial-peer voice 1022501 voip
corlist incoming From-BGL-Ent3105
description Inbound Trunk from Ent3105 CUCM
session protocol sipv2
destination dpg 9994
incoming uri via 102252
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/1
voice-class sip bind media source-interface GigabitEthernet0/0/1
dial-peer voice 1022502 voip
corlist outgoing To-BGL-Ent3105
description *** Outbound Trunk to BT-Ent304 DN
Reouting ****
destination-pattern 8115
session protocol sipv2
session target ipv4:10.225.104.195
session transport tcp
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/1
voice-class sip bind media source-interface GigabitEthernet0/0/1

!
dial-peer voice 1022503 voip
corlist incoming From-RCDN-Ent3106
description Inbound Trunk from PGW-Ent3105
session protocol sipv2
session transport tcp
destination dpg 9992
incoming uri via 102254
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/0.306
voice-class sip bind media source-interface GigabitEthernet0/0/0.306

!
dial-peer voice 1022504 voip
corlist outgoing To-RCDN-Ent3106
description Outbound Trunk to PGW-Ent3105
destination-pattern 8115
session protocol sipv2
session target ipv4:200.1.1.10:7016
session transport tcp
voice-class codec 1
voice-class sip profiles 1
voice-class sip options-keepalive
voice-class sip pass-thru content sdp
voice-class sip bind control source-interface GigabitEthernet0/0/0.306
voice-class sip bind media source-interface GigabitEthernet0/0/0.306

!

Debug Information

Note
Execute `sh sip-ua calls called-number +16089185043`, to check the call behaviour.
Figure 38: Debug Information

The following commands are helpful for troubleshooting:

- **show voip rtp connections**

  The following is an example where media flow-around is configured. The output shows 0 connections since media does not flow through CUBE.

  Device# show voip rtp connections
  VoIP RTP Port Usage Information:
  Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 0
  Port range not configured
  Media-Address Range Min Max Ports Available Reserved In-use
  Global Media Pool 8000 48198 19999 101 0

  No active connections found

- **show call active voice compact**
Device#show call active voice compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp> VRF
4021 ORG T45 g711ulaw VOIP P7474 8.41.17.71:27754 VRF1
4020 ANS T45 g711ulaw VOIP Psipp 8.41.17.71:17001 VRF1

• debug ccsip verbose

The output of debug ccsip verbose command is wordy and may cause issues when enabled on a busy network environment.
CHAPTER 28

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 395
- Information About Configuring Multi-tenants on SIP Trunks, on page 395
- How to Configure Multi-Tenants on SIP Trunks, on page 399
- Example: SIP Trunk Registration in Multi-Tenant Configuration, on page 401

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 48: 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 48: 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>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>call-route</td>
<td>Configure call routing options</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</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>encap</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>midcall -signaling</td>
<td>Configure method to handle mid-call signaling</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</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>
<tr>
<td>sip-profiles</td>
<td>SIP Profiles global 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. 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>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</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# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>Use the following commands to configure multi-tenants:</td>
<td>Use the <code>voice-class sip tenant &lt;tag&gt;</code> command in the global configuration mode to configure a tenant with sip-specific attributes. This command tag can then be applied to one or more dial-peers using the <code>voice-class sip tenant &lt;tag&gt;</code> command under the dial-peers.</td>
</tr>
<tr>
<td>• <code>voice class tenant &lt;tag&gt;</code> in the global configuration mode</td>
<td></td>
</tr>
<tr>
<td>Once you configure the <code>voice class tenant &lt;tag&gt;</code> command in the global mode, the configuration will move to the <code>voice class tenant &lt;tag&gt;</code> submode. You can configure all the sip-specific attributes in this submode.</td>
<td></td>
</tr>
<tr>
<td>• <code>voice-class sip tenant &lt;tag&gt;</code> in the dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In global configuration mode</td>
<td></td>
</tr>
<tr>
<td>! Configuring tenant 1</td>
<td></td>
</tr>
<tr>
<td>Device(config)# <code>voice class tenant 1</code></td>
<td></td>
</tr>
<tr>
<td>Device (config-class)# ?</td>
<td></td>
</tr>
<tr>
<td>aaa – sip-ua AAA related configuration</td>
<td></td>
</tr>
<tr>
<td>anat – Allow alternative network address types IPV4 and IPV6</td>
<td></td>
</tr>
<tr>
<td>asserted-id – Configure SIP-UA privacy identity settings</td>
<td></td>
</tr>
<tr>
<td>……</td>
<td></td>
</tr>
<tr>
<td>Video – video related function</td>
<td></td>
</tr>
<tr>
<td>Warn-header – SIP related config for SIP. SIP warning-header global config.</td>
<td></td>
</tr>
<tr>
<td>Device (config-voi-tenant)# end</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>! Configuring tenant 2</td>
<td></td>
</tr>
<tr>
<td>Device(config)# <code>voice class tenant 2</code></td>
<td></td>
</tr>
<tr>
<td>Device (config-class)# ?</td>
<td></td>
</tr>
<tr>
<td>aaa – sip-ua AAA related configuration</td>
<td></td>
</tr>
<tr>
<td>anat – Allow alternative network address types IPV4 and IPV6</td>
<td></td>
</tr>
<tr>
<td>asserted-id – Configure SIP-UA privacy identity settings</td>
<td></td>
</tr>
<tr>
<td>……</td>
<td></td>
</tr>
<tr>
<td>outbound-proxy – Configure an Outbound Proxy Server</td>
<td></td>
</tr>
<tr>
<td>pass-thru – SIP pass-through global config</td>
<td></td>
</tr>
<tr>
<td>……</td>
<td></td>
</tr>
<tr>
<td>srtp – Allow SIP related SRTP options</td>
<td></td>
</tr>
<tr>
<td>Warn-header – SIP related config for SIP. SIP warning-header global config.</td>
<td></td>
</tr>
<tr>
<td>Device (config-voi-tenant)# <code>end</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></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>In dial-peer configuration mode</strong></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)# 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)# 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**

| 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:

```plaintext
voice class tenant 1
  credentials number 1111 username test password 7 B245B5D1D realm ipvoice.jp
  authentication username test password 7 06120A325B
  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 B245F5A
  registrar ipv4:1.1.1.1 expires 120
  sip-server ipv4:1.1.1.1
```
PART IV

Codecs

• Codec Support and Restrictions, on page 405
• Codec Preference Lists, on page 411
CHAPTER 29

Codec Support and Restrictions

This chapter provides advanced information about the support of and restrictions for using certain codecs with 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 405
- OPUS Codec Support on CUBE, on page 406
- ISAC Codec Support on CUBE, on page 408
- AAC-LD MP4A-LATM Codec Support on Cisco UBE, on page 408

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>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opus Codec Support</td>
<td>Cisco IOS XE Amsterdam 17.3.1a</td>
<td>Opus audio codec support on CUBE was introduced.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following commands were introduced or modified as part of Opus codec feature:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• codec opus [profile tag]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• codec profile tag profile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• codec preference value codec-type [profile profile-tag]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• rtp payload-type opus payload-type-number</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>AAC-LD MP4A-LATM Codec Support</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 that is 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</code>, <code>codec preference tag mp4a-latm</code>.</td>
</tr>
<tr>
<td>ISAC Codec Support</td>
<td>15.1(1)T</td>
<td>The ISAC Codec Support on CUBE was introduced. The following commands were introduced by this feature: <code>codec isac</code>, <code>codec preference tag isac</code>.</td>
</tr>
</tbody>
</table>

### OPUS Codec Support on CUBE

The Opus interactive speech and audio codec is designed to handle a wide range of interactive audio applications. This includes VoIP, video conferencing, and in-game chat. The OPUS codec also supports live, distributed music performances. It scales from low bit rate narrowband speech at 6 kbps to high-quality stereo music at 510 kbps. Opus uses both Linear Prediction (LP) and Modified Discrete Cosine Transform (MDCT) algorithms to achieve good compression of both speech and music.

**Note**  
Opus codec is only supported for SIP-SIP call scenarios. It is not supported for TDM-SIP or Analog-SIP flows.

### Design Recommendations for Opus Codec

The VoIP dial peer and voice class codec options are enhanced to offer Opus codec support on CUBE:

- **codec opus profile tag**—The CLI command under dial-peer configuration mode is enhanced:

  ```
  router(config)#dial-peer voice 3002 voip
  router(config-dial-peer)#codec opus profile 2
  ```

- **codec profile tag profile**—The CLI command configured under global configuration mode is enhanced to configure opus as a supported codec:

  ```
  router(config)#codec profile 2 opus
  router(conf-codec-profile)#fmtp "fmtp:114 maxplaybackrate=16000;
  sprop-maxcapturerate=16000; maxaveragebitrate=20000; stereo=1; sprop-stereo=0;
  useinbandfec=0; usedtx=0"
  router(conf-codec-profile)#exit
  ```

- **codec preference value codec-type [ profile profile tag]**—The CLI command configured under voice class configuration mode is enhanced to configure opus as a preferred codec on the dial-peer:
router(config)#voice class codec 80
router(config-class)#codec preference 1 opus profile 79
router(config-class)#exit

- **rtp payload-type [opus number]**—The CLI command configured under dial-peer configuration mode is enhanced to configure opus as a supported payload type:

  router(config)#dial-peer voice 604 voip
  router(config-dial-peer)#rtp payload-type opus 126

  - The default payload-type for **opus** is set to 114.
  - The default payload-type for **cisco-codec-aacld** is set to 112.

- It is not mandatory that you configure a **codec profile** for Opus. For delayed offer to early offer flows, a codec profile must be used to configure an fmysql value for the initial invite.

- The CLI command **show call active voice [brief | compact]** is modified to include details on Opus codec in the command output.

- The CLI command **show sip-ua calls** is modified to include details on Opus codec in the command output.

- Calls that require Opus codec transcoding as part of the Offer-Answer exchange are disconnected.

- Opus Codec is supported for both secure and non-secure calls (RTP-to-RTP, SRTP-to-SRTP, SRTP-to-RTP, and RTP-to-SRTP).

- Opus supports several clock rates. Only the highest clock rate of 48000 is advertised in the SDP. The following is a sample configuration of SDP for Opus codec:

  m=audio 16000 RTP/AVP 114 a=rtpmap:114 opus/48000/2

- Opus codec defines the optional media format (fmysql) parameters in a call. CUBE passes through the optional fmysql parameters from one side to other if Opus codec is configured on both sides of the call.

- The following are the fmysql parameters defined by Opus codec:

  - maxaveragebitrate
  - maxplaybackrate
  - stereo
  - useinbandfeci
  - usedtx
  - sprop-maxcapturerate
  - sprop-stereo

- Dynamic payload interworking is enabled by default on CUBE. Configure **asymmetric payload [full]** if CUBE is not required to handle payload interworking. In this scenario, interworking is handled at the endpoints handling the media.

### Restrictions for Opus Codec Support on CUBE

- Media recording is not supported with Extended Media Forking (XMF).
• CUBE does not support processing of multiple fmtp lines. If the received SDP has multiple fmtp lines, then only the first fmtp line is passed in the outbound INVITE.

ISAC Codec Support on CUBE

The iSAC codec is an adaptive VoIP codec specially 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 for DO-EO (Delayed-Offerto Early-Offert) 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-SIP, that is, SIP to other service provider interface (SPI) interworking with MP4A-LATM codec
CHAPTER 30

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 411
- Codecs Configured Using Preference Lists, on page 412
- Prerequisites for Codec Preference Lists, on page 412
- Restrictions for Codecs Preference Lists, on page 413
- How to Configure Codec Preference Lists, on page 413
- Troubleshooting Negotiation of an Audio Codec from a List of Codecs, on page 416
- Verifying Negotiation of an Audio Codec from a List of Codecs, on page 417

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 50: 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, H.323-to-H.323, SIP-to-H.323 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 H.323-to-H.323 and SIP-to-H.323 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[profile profile-tag ]
   - 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>
<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&gt; configure terminal</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>Example: Device(config)# voice class codec 10</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> Do the following for each audio codec you want to configure in the voice class:</td>
<td>Configure a codec within the voice class and specifies a preference for the codec. This becomes part of a preference list</td>
</tr>
<tr>
<td>• codec preference value codec-type[profile profile-tag ]</td>
<td></td>
</tr>
<tr>
<td>• codec preference value codec-type[bytes payload-size fixed-bytes ]</td>
<td></td>
</tr>
<tr>
<td>• codec preference value isac [mode {adaptive</td>
<td>independent} [bit-rate value framesize { 30</td>
</tr>
<tr>
<td>• codec preference value ilbc [mode frame-size [bytes payload-size]]</td>
<td></td>
</tr>
<tr>
<td>• codec preference value mp4-latm [profile tag]</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exits the current mode.</td>
</tr>
</tbody>
</table>
## Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 6</strong></td>
<td>Enters dial peer configuration mode for the specified VoIP dial peer.</td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device(config)# dial-peer voice 1 voip</code></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Applies the previously configured voice class and associated codecs to a dial peer.</td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device(config-dial-peer)# voice-class codec 10</code></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device(config-dial-peer)# end</code></td>
</tr>
</tbody>
</table>

## 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.

Perform this task to disable codec filtering and allow all the codecs configured on an outbound leg.

### 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><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device&gt; enable</code></td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device(config-class)# exit</code></td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device(config)# dial-peer voice 1 voip</code></td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device(config-dial-peer)# voice-class codec 10</code></td>
</tr>
<tr>
<td><code>Example:</code></td>
<td><code>Device(config-dial-peer)# end</code></td>
</tr>
</tbody>
</table>
### Command or Action

**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 10 voip
```
Enters dial peer voice configuration mode.

**Step 4**
```
voice-class codec tag offer-all
```
Example:
```
Device(config-dial-peer)# voice-class codec 10 offer-all
```
Adds all the configured voice class codec to the outgoing offer from the Cisco UBE.

**Step 5**
```
end
```
Example:
```
Device(config-dial-peer)# end
```
Exits the dial peer voice configuration mode.

### 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
   Enables privileged EXEC mode.
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 pl:<play>/<gap>ms lost:<lost>/<early>/<late>
   delay:<last>/<min>/<max>ms codec
   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>
   MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
   last <buf event time> 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
   MODEMRELAY info:<rcvd>/<sent> xid:<rcvd> total:<rcvd>/<sent>/<drops>
   speedbps: local <rx>/remote <rx>/tx
   Proxy <ip>,<audio udp>,<video udp>,<tcp0>,<tcp1>,<tcp2>,<TCP3> endpt: <type>/<manf>
   bw: <req>/<act> codec: <audio>/<video>
   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 SRTF: 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
1243 : 12 971500ms.1 +1 pid: 2 Originate 3210000 connected
dur 00:00:00 tx: 5/10 rx: 4/8
IP 9.44.26.4: 16512 SRTP: off rtt: 0/0ms lost: 0/0/0 delay: 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
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: 0/0ms lost: 0/0/0 delay: 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 : 15 971570ms.1 +0 pid: 0 Originate connecting
dur 00:00:08 tx: 5/10 rx: 3/6
IP 192.0.2.3: 2000 SRTP: off rtt: 0/0ms lost: 0/0/0 delay: 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 192.0.2.1 192.0.2.3
2 12 11 17404 16512 192.0.2.2 192.0.2.3
3 13 14 18422 192.0.2.4 192.0.2.6 192.0.2.3
4 15 14 16576 192.0.2.6 192.0.2.6 192.0.2.5
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`

<table>
<thead>
<tr>
<th>SLOT</th>
<th>DSP VERSION</th>
<th>STATUS</th>
<th>CHNL</th>
<th>USE</th>
<th>TYPE</th>
<th>RSC_ID</th>
<th>BRIDGE_ID</th>
<th>PKTS_TXED</th>
<th>PKTS_RXED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>27.0.201</td>
<td>UP</td>
<td>1</td>
<td>USED</td>
<td>xcode</td>
<td>0x9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>27.0.201</td>
<td>UP</td>
<td>1</td>
<td>USED</td>
<td>xcode</td>
<td>0x8</td>
<td>2558</td>
<td>17</td>
</tr>
</tbody>
</table>

Total number of DSPFARM DSP channel(s) 1
PART V

DSP Services

• Transcoding, on page 423
• Transrating, on page 441
• Call Progress Analysis Over IP-to-IP Media Session, on page 443
• Voice Packetization, on page 451
• Fax Detection for SIP Call and Transfer, on page 453
Transcoding

Transcoding is a process of converting one voice codec to another. For example, transcoding between iLBC and G.711 or iLBC and G.729.

**Note**
In every transcoding operation, media flows through CUBE.

**LTI based Transcoding**

- Internal API is used to access Digital Signaling Processor (DSP) resources for transcoding.
- Transcoding resources (DSPFARM) and CUBE must be on the same platform.
- Only DSPFARM profile configuration is required. Skinny Client Control Protocol (SCCP) 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 is in line high-density mode that is 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**
The following support LTI-based transcoding:

- Cisco Aggregated Services Routers 1000 Series (ASR 1K)
- Cisco-Integrated Services Generation 2 Routers (Cisco ISR G2)
- Cisco 4000 Series-Integrated Services Routers (ISR G3)
- Cisco 8200 Catalyst Edge Series
- Cisco 8300 Catalyst Edge Series
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, sccp 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.
- DSPFARM profile is associated to SCCP using the following commands:

```
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.

---

Note

Integrated Services Routers Generation 1 series and Integrated Services Routers Generation 2 Series devices support SCCP-based Transcoding only.

- Configure LTI-Based Transcoding, on page 424
- Configuration Examples for LTI Based Transcoding, on page 426
- Configuring SCCP-based Transcoding (ISR-G2 devices only), on page 428
- TLS for SCCP Connection for DSP Services, on page 431
- Configuring Secure Transcoding, on page 431
- Configuration Examples for SCCP Based Transcoding, on page 439

---

Configure 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), Cisco 4000 Series Integrated Services Routers (ISR G3), Cisco 8200 Catalyst Edge Series, and Cisco 8300 Catalyst Edge Series.

---

SUMMARY STEPS

1. enable
2. configure terminal
3. voice-card voice-interface-slot-number
4. dspfarm services dspfarm
5. exit
6. `dsfparm profile profile-identifier transcode`
7. `codec codec`
8. `maximum sessions sessions`
9. `associate application CUBE`
10. `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> <code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device&gt; configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>voice-card voice-interface-slot-number</code></td>
<td>Configures a voice card and enters voice-card configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice-card 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>dsfparm services dsfparm</code></td>
<td>Enable voice-only DSP farm services on the Voice Card.</td>
</tr>
<tr>
<td><strong>Step 5</strong> <code>exit</code></td>
<td>Exits the voice-card configuration mode.</td>
</tr>
<tr>
<td><strong>Step 6</strong> <code>dsfparm profile profile-identifier transcode</code></td>
<td>Enters configuration mode for a DSP farm profile and defines a profile for DSP farm services.</td>
</tr>
<tr>
<td>Example: Device(config)# dsfparm profile 1 transcode</td>
<td>• <code>profile-identifier</code>- Number that uniquely identifies a profile. Range: 1–65535.</td>
</tr>
<tr>
<td></td>
<td>• <code>transcode</code>- Enables profile for transcoding.</td>
</tr>
<tr>
<td><strong>Step 7</strong> <code>codec codec</code></td>
<td>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.</td>
</tr>
<tr>
<td>Example: Device(config-dsfparm-profile)# codec ilbc</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> <code>maximum sessions sessions</code></td>
<td>Configures maximum number of sessions.</td>
</tr>
<tr>
<td><strong>Step 9</strong> <code>associate application CUBE</code></td>
<td>Configures an application to the profile for LTI-based transcoding.</td>
</tr>
<tr>
<td>Example: Device(config)# associate application CUBE</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> <code>exit</code></td>
<td>Exits interface configuration mode.</td>
</tr>
</tbody>
</table>
Configuration Examples for LTI Based Transcoding

LTI-based Transcoding

! Enabling dspfarm services under voice-card
Device(config)# voice-card 0/1
Device(config-voicecard)# dspfarm
Device(config-voicecard)# dsp services dspfarm
Device(config-voicecard)# 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 g729a
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 8E32BEB6

% 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 transcoding 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-ephones 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>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&gt; configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice-card voice-interface-slot-number</td>
<td>Configures a voice card and enters voice-card configuration mode.</td>
</tr>
<tr>
<td>Example: <code>Device(config)# voice-card 1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> dsfpfarm</td>
<td>Enable voice card for DSP.</td>
</tr>
<tr>
<td><strong>Step 5</strong> dsfp service dsfpfarm</td>
<td>Enable voice-only dsfpfarm services on the Voice Card.</td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits the voice-card configuration mode.</td>
</tr>
<tr>
<td><strong>Step 7</strong> telephony-service</td>
<td>Enters telephony-service configuration mode.</td>
</tr>
<tr>
<td><strong>Step 8</strong> sdspfarm units units</td>
<td>Define maximum number of dsfpfarm units.</td>
</tr>
<tr>
<td><strong>Step 9</strong> sdspfarm transcode sessions units</td>
<td>Define maximum number of dsfpfarm transcode session.</td>
</tr>
<tr>
<td><strong>Step 10</strong> sdspfarm tag value Device-Name</td>
<td>Configures a name for the transcoder.</td>
</tr>
<tr>
<td>Example: <code>Device(config-telephony)# sdspfarm tag 1 CUBE-XCODE</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong> max-ephones max-phones-to-be-supported</td>
<td>Configures the maximum number of phones that are to be supported.</td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td><code>max-dn</code> max-directorynumbers-to-be-supported</td>
<td>Configures the maximum number of directories to be supported.</td>
</tr>
<tr>
<td>13</td>
<td><code>ip source-address CUBE-internal-ipv4-address [port port-number]</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-telephony)# ip source-address 10.1.1.1 port 2000</td>
<td>Defines an IP address and port number for the telephony service.</td>
</tr>
<tr>
<td>14</td>
<td><code>exit</code></td>
<td>Exits the telephony-service configuration mode.</td>
</tr>
<tr>
<td>15</td>
<td><code>scpp local interface-type number</code></td>
<td>Configures the local gateway related parameters values.</td>
</tr>
<tr>
<td>16</td>
<td><code>scpp ccm CUBE-internal-ipv4-address identifier identifier-number version version-number</code></td>
<td>Configures call manager related parameter values.</td>
</tr>
<tr>
<td>17</td>
<td><code>scpp</code></td>
<td>Enable Skinny Client Control Protocol.</td>
</tr>
<tr>
<td>18</td>
<td><code>scpp ccm group group-id</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config)# scpp ccm group 1</td>
<td>Configures Call Manager Group and enters SCCP CCM configuration mode.</td>
</tr>
<tr>
<td>19</td>
<td><code>associate ccm CCM-identifier priority priority</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-sccp-ccm)# associate ccm 1 priority 1</td>
<td>Configures Call Manager Group and enters SCCP CCM configuration mode.</td>
</tr>
<tr>
<td>20</td>
<td><code>associate profile profile-identifier register Device-Name</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-sccp-ccm)# associate profile 1 register CUBE-XCODE</td>
<td>Specifies the device name that needs to register.</td>
</tr>
<tr>
<td>21</td>
<td><code>exit</code></td>
<td>Exits SCCP CCM configuration mode.</td>
</tr>
<tr>
<td>22</td>
<td><code>dsppfarm profile profile-id transcode</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config)# dsppfarm profile 1 transcode</td>
<td>Configures a Transcoding profile and enters DSP profile configuration mode.</td>
</tr>
<tr>
<td>23</td>
<td><code>codec codec</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-dsppfarm-profile)# codec ilbc</td>
<td>The codec rate to be attempted for SCCP-controlled connections.</td>
</tr>
<tr>
<td>24</td>
<td><code>maximum sessions sessions</code></td>
<td>Configures maximum number of sessions.</td>
</tr>
<tr>
<td>25</td>
<td><code>associate application sccp</code>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-dsppfarm-profile)# associate application sccp</td>
<td>Configures an application to the profile for SCCP-based transcoding.</td>
</tr>
<tr>
<td>26</td>
<td><code>exit</code></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>
</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. |
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 3</td>
<td><code>ip http server</code></td>
<td>Enables the HTTP server on your IPv4 or IPv6 system, including the Cisco web browser user interface.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(config)# ip http server</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>crypto pki server</code> <code>cs-label</code></td>
<td>Enables a Cisco IOS certificate server and enters certificate server configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(config)# crypto pki server 3854-cube</code></td>
<td>• In the example, 3854-cube is specified as the name of the certificate server.</td>
</tr>
<tr>
<td>Step 5</td>
<td><code>database level</code> <code>complete</code></td>
<td>Controls what type of data is stored in the certificate enrollment database.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(cs-server)# database level complete</code></td>
<td>• In the example, each issued certificate is written to the database.</td>
</tr>
<tr>
<td>Step 6</td>
<td><code>grant auto</code></td>
<td>Specifies automatic certificate enrollment.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(cs-server)# grant auto</code></td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><code>no shutdown</code></td>
<td>Reenables the certificate server.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(cs-server)# no shutdown</code></td>
<td>• Create and enter a new password when prompted.</td>
</tr>
<tr>
<td>Step 8</td>
<td><code>exit</code></td>
<td>Exits certificate server configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Device(cs-server)# exit</code></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 431.

**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>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 | `crypto pki trustpoint name`  
Example: Device(config)# crypto pki trustpoint secdsp | Declares the trustpoint that the router uses and enters ca-trustpoint configuration mode.  
- In the example, the trustpoint is named secdsp. |
| Step 4 | `enrollment url url`  
Example: Device(ca-trustpoint)# enrollment url http://10.13.2.52:80 | Specifies the enrollment parameters of a certification authority (CA).  
- In the example, the URL is defined as http://10.13.2.52:80. |
| Step 5 | `serial-number`  
Example: Device(ca-trustpoint)# serial-number | Specifies whether the router serial number should be included in the certificate request. |
| Step 6 | `revocation-check method`  
Example: Device(ca-trustpoint)# revocation-check crl | Checks the revocation status of a certificate.  
- In the example, the certificate revocation list checks the revocation status. |
| Step 7 | `rsakeypair key-label`  
Example: Device(ca-trustpoint)# rsakeypair 3845-cube | Specifies which key pair to associate with the certificate.  
- In the example, the key pair 3845-cube generated during enrollment is associated with the certificate. |
## Configuring DSPFARM Services

For configuration steps, see [Configure 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> 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><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> sccp local interface-type interface-number</td>
<td>Selects the local interface that SCCP applications (transcoding and conferencing) use to register with Cisco CallManager.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# sccp local GigabitEthernet 0/0</td>
<td>• In the example, the following parameters are set:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• GigabitEthernet is defined as the interface type that the SCCP application uses to register with Cisco CallManager.</td>
</tr>
<tr>
<td></td>
<td>• The interface number that the SCCP application uses to register with Cisco CallManager is specified as 0/0.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sccp ccm ip-address identifier identifier-number version version-number</td>
<td>Adds a Cisco Unified Communications Manager server to the list of available servers.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# sccp ccm 10.13.2.52 identifier 1 version 5.0.1</td>
<td>• In the example, the following parameters are set:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 10.13.2.52 is configured as the IP address of the Cisco Unified Communications Manager server.</td>
</tr>
<tr>
<td></td>
<td>• The number 1 identifies the Cisco Unified Communications Manager server.</td>
</tr>
<tr>
<td></td>
<td>• The Cisco Unified Communications Manager version is identified as 5.0.1.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> sccp</td>
<td>Enables SCCP and related applications (transcoding and conferencing) and enters SCCP Cisco CallManager configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# sccp</td>
<td></td>
</tr>
</tbody>
</table>
## DSP Services

### Associating SCCP to the Secure DSPFARM Profile

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 6**

*associate ccm identifier-number priority priority-number*

**Example:**

Device (config-sccp-ccm)# associate ccm 1 priority 1

- Associates a Cisco Unified CallManager with a Cisco CallManager group and establishes its priority within the group.
- In the example, the following parameters are set:
  - The number 1 identifies the Cisco Unified CallManager.
  - The Cisco Unified CallManager is configured with the highest priority within the Cisco CallManager group.

| **Step 7**

*associate profile profile-identifier register device-name*

**Example:**

Device (config-sccp-ccm)# associate profile 1 register sxcoder

- Associates a DSPFARM profile with a Cisco CallManager group.
- In the example, the following parameters are set:
  - The number 1 identifies the DSPFARM profile.
  - Sxcoder is configured as the user-specified device name in Cisco Unified CallManager.

| **Step 8**

*dsfarm profile profile-identifier transcode universal security*

**Example:**

Device (config-sccp-ccm)# dsfarm profile 1 transcode universal security

- 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.

| **Step 9**

*trustpoint trustpoint-label*

**Example:**

Device (config-dspfarm-profile)# trustpoint secdsp

- Associates a trustpoint with a DSPFARM profile.
  - In the example, the trustpoint to be associated with the DSPFARM profile is labeled secdsp.

| **Step 10**

*codec codec-type*

**Example:**

Device (config-dspfarm-profile)# codec g711ulaw

- Specifies the codecs that are supported by a DSPFARM profile.
  - In the example, the g711ulaw codec is specified.

| **Step 11**

Repeat Step 10 to configure required codecs.

**Example:**

- --

| **Step 12**

*maximum sessions number*

**Example:**

Device (config-dspfarm-profile)# maximum sessions 84

- 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.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 13</strong></td>
<td><strong>purpose</strong></td>
</tr>
<tr>
<td><em>associate application sccp</em></td>
<td>Associates SCCP to the DSPFARM profile.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dspfarm-profile)# <code>associate application sccp</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td><strong>purpose</strong></td>
</tr>
<tr>
<td><em>no shutdown</em></td>
<td>Allocates DSPFARM resources and associates them with the application.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dspfarm-profile)# <code>no shutdown</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td><strong>purpose</strong></td>
</tr>
<tr>
<td><em>exit</em></td>
<td>Exits DSPFARM profile configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config-dspfarm-profile)# <code>exit</code></td>
<td></td>
</tr>
</tbody>
</table>

### 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 associate SCCP to the secure DSPFARM profile, as described in the **Associating SCCP to the Secure DSPFARM Profile**, on page 434.

### 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></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td><em>Enter your password if prompted.</em></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
</tbody>
</table>

| **Step 2** | Enters global configuration mode. |
| `configure terminal` | |
| **Example:** | |
| `Device> configure terminal` | |

| **Step 3** | Enters telephony-service configuration mode. |
| `telephony-service` | |
| **Example:** | |
| `Device(config)# telephony-service` | |

| **Step 4** | Specifies the maximum number of transcoding sessions allowed per Cisco CallManager Express router. |
| `sdspfarm transcode sessions number` | |
| **Example:** | |
| `Device(config-telephony)# sdspfarm transcode sessions 84` | 

| **Step 5** | Permits a DSPFARM to be registered to Cisco Unified CallManager Express and associates it with an SCCP client interface's MAC address. |
| `sdspfarm tag number device-name` | |
| **Example:** | |
| `Device(config-telephony)# sdspfarm tag 1 sxcoder` | 

| **Step 6** | Configures three time-of-day-based timers for automatically logging out all Extension Mobility feature users. |
| `em logout time1 time2 time3` | |
| **Example:** | |
| `Device(config-telephony)# em logout 0:0 0:0 0:0` | 

| **Step 7** | Sets the maximum number of Cisco IP phones to be supported by a Cisco CallManager Express router. |
| `max-ephones max-ephones` | |
| **Example:** | |
| `Device(config-telephony)# max-ephones 4` | 

<p>| <strong>Step 8</strong> | Sets the maximum number of extensions (ephone-dns) to be supported by a Cisco Unified CallManager Express router. |
| <code>max-dn max-directory-numbers</code> | |
| <strong>Example:</strong> | |
| <code>Device(config-telephony)# max-dn 4</code> | |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>Identifies the IP address and port through which IP phones communicate with a Cisco Unified CallManager Express router.</strong></td>
</tr>
</tbody>
</table>
| *ip source-address ip-address*  
*Example:*  
Device(config-telephony)# ip source-address 10.13.2.52  
*Purpose:* 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.* |
| **Step 10** | **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.** |
| *secure-signaling trustpoint label*  
*Example:*  
Device(config-telephony)# secure-signaling trustpoint secdsp  
*Purpose:* 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.* |
| **Step 11** | **Specifies the PKI trustpoint that signs the phone configuration files.** |
| *tftp-server-credentials trustpoint label*  
*Example:*  
Device(config-telephony)# tftp-server-credentials trustpoint scme  
*Purpose:* Specifies the PKI trustpoint that signs the phone configuration files.  
*In the example, PKI trustpoint scme is configured.* |
| **Step 12** | **Builds the XML configuration files that are required for IP phones in Cisco Unified CallManager Express.** |
| *create cnf-files*  
*Example:*  
Device(config-telephony)# create cnf-files  
*Purpose:* Builds the XML configuration files that are required for IP phones in Cisco Unified CallManager Express. |
| **Step 13** | **Disables SCCP and its related applications (transcoding and conferencing) and exits telephony-service configuration mode.** |
| *no sccp*  
*Example:*  
Device(config-telephony)# no sccp  
*Purpose:* Disables SCCP and its related applications (transcoding and conferencing) and exits telephony-service configuration mode. |
| **Step 14** | **Enables SCCP and related applications (transcoding and conferencing).** |
| *sccp*  
*Example:*  
Device(config)# sccp  
*Purpose:* Enables SCCP and related applications (transcoding and conferencing). |
| **Step 15** | **Exits global configuration mode.** |
| *end*  
*Example:*  
Device(config)# end  
*Purpose:* Exits global configuration mode. |

**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)# sdspfarm units 1
Device(config-telephony)# sdspfarm transcode sessions 128
Device(config-telephony)# sdspfarm 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-XCODE
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 g729r8
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.

• Configuring Transrating for a Codec, on page 441

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>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>enable</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>Example:</td>
<td>configure terminal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device&gt; configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>dial-peer voice number voip</td>
<td>Enters dial peer configuration mode for the specified VoIP dial peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>dial-peer voice 1 voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 1 voip</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>codec codec-name bytes voice-payload-size [fixed-bytes]</td>
<td>Configures a different packetizations for a voice codec.</td>
</tr>
<tr>
<td>Example:</td>
<td>codec g729r8 bytes 30 fixed-byte</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 5</strong> end</td>
<td>Exits to privileged EXEC mode.</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```
Device(config-dial-peer)# end
```
CHAPTER 33

Call Progress Analysis Over IP-to-IP Media Session

The Call Progress Analysis Over IP-to-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 443
• Restrictions for Call Progress Analysis Over IP-to-IP Media Session, on page 444
• Information About Call Progress Analysis Over IP-IP Media Session, on page 445
• How to Configure Call Progress Analysis Over IP-to-IP Media Session, on page 446
• Configuration Examples for the Call Progress Analysis Over IP-to-IP Media Session, on page 449

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.
Table 51: Feature Information for Call Progress Analysis Over IP-IP Media Session

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Progress Analysis Over IP-to-IP Media Session</td>
<td>15.3(2)T</td>
<td>The Call Progress Analysis Over IP-to-IP Media Session feature enables detection of automated answering systems and live human voices on outbound calls and communicates the detected information to an external application. The following command was introduced: <code>call-progress-analysis</code>.</td>
</tr>
<tr>
<td>Call Progress Analysis Over IP-to-IP Media Session</td>
<td>Cisco IOS XE Release 3.9S</td>
<td>The Call Progress Analysis Over IP-to-IP Media Session feature enables detection of automated answering systems and live human voices on outbound calls and communicates the detected information to an external application. The following command was introduced: <code>call-progress-analysis</code>.</td>
</tr>
<tr>
<td>Support for additional call flows</td>
<td>15.5(2)T Cisco IOS XE Release 3.15S</td>
<td>Call Progress Analysis feature is enhanced to support the following call-flows: 180 SIP response received without SDP, Direct call connect (without 18x from Service Provider), Multiple 18x response to INVITE, Early dialog UPDATE, Dialer-CUBE CPA call record</td>
</tr>
</tbody>
</table>

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.
With VCC codec configured on the dial-peer, the list of codecs in the VCC should match with the list of codec provisioned in DSP transcoder profile when CPA is enabled.

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>
<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>
<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>
</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>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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPA Event</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS</td>
<td>Live human speech</td>
</tr>
<tr>
<td>LV</td>
<td>Low volume or dead air call</td>
</tr>
<tr>
<td>SitIC</td>
<td>Special information tone IC -- Intercept -- Vacant number or Automatic Identification System (AIS)</td>
</tr>
<tr>
<td>SitNC</td>
<td>SIT tone NC—No Circuit (NC), Emergency, or Trunk Blockage</td>
</tr>
<tr>
<td>SitVC</td>
<td>SIT tone VC—Vacant Code</td>
</tr>
<tr>
<td>SitRO</td>
<td>SIT tone RO—Reorder Announcement</td>
</tr>
<tr>
<td>SitMT</td>
<td>Miscellaneous SIT Tone</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>2</td>
<td>configure terminal</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td>3</td>
<td>dspfarm profile <em>profile-identifier</em> transcode</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# dspfarm profile 15 transcode</td>
</tr>
<tr>
<td>4</td>
<td>call-progress-analysis</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dspfarm-profile)# call-progress-analysis</td>
</tr>
<tr>
<td>5</td>
<td>exit</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dspfarm-profile)# exit</td>
</tr>
<tr>
<td>6</td>
<td>voice service voip</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# voice service voip</td>
</tr>
<tr>
<td>7</td>
<td>cpa timing live-person <em>max-duration</em></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(conf-voi-serv)# cpa timing live-person 2501</td>
</tr>
<tr>
<td>8</td>
<td>cpa timing term-tone <em>max-duration</em></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(conf-voi-serv)# cpa timing term-tone 15500</td>
</tr>
<tr>
<td>9</td>
<td>cpa threshold active-signal <em>signal-threshold</em></td>
</tr>
<tr>
<td>Example:</td>
<td>Device(conf-voi-serv)# cpa threshold active-signal 18db</td>
</tr>
<tr>
<td></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>
</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 ccapi 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
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 14: 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 54: Packetization for different Codecs

<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, on page 451

Configuring Transrating for a Codec

SUMMARY STEPS

1. enable
2. configure terminal
3. dial-peer voice number voip
### 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.  
Example: 
`Device> enable`  
• Enter your password if prompted. |
| 2    | `configure terminal` | Enters global configuration mode.  
Example: 
`Device> configure terminal` |
| 3    | `dial-peer voice number voip` | Enters dial peer configuration mode for the specified VoIP dial peer.  
Example: 
`Device(config)# dial-peer voice 1 voip` |
| 4    | `codec codec-name bytes voice-payload-size [fixed-bytes]` | Configures a different packetizations for a voice codec.  
Example: 
`Device(config-dial-peer)# codec g729r8 bytes 30 fixed-byte` |
| 5    | `end` | Exits to privileged EXEC mode.  
Example: 
`Device(config-dial-peer)# end` |
Fax Detection for SIP Call and Transfer

The fax detection feature detects whether an inbound call is from a fax machine. If the inbound call is from a fax machine, the call is rerouted appropriately.

- Restrictions for Fax Detection for SIP Call and Transfer On Cisco IOS XE, on page 453
- Information About Fax Detection for SIP Call and Transfer, on page 453
- Fax Detection with Cisco IOS XE High Availability, on page 456
- How to Configure Fax Detection for SIP Calls, on page 456
- Configuration Examples for Fax Detection for SIP Calls, on page 460
- Feature Information for Fax Detection for SIP Call and Transfer, on page 461

Restrictions for Fax Detection for SIP Call and Transfer On Cisco IOS XE

- The Fax Detect feature is only supported with routers fitted with DSP modules.
- Only the g711ulaw and g711alaw codecs can be used for detecting fax CNG tone.
- Each destination number can be of a maximum length of 32 characters.
- Fax Detection is only supported with LTI-based transcoding.

Information About Fax Detection for SIP Call and Transfer

Fax detection is typically used if you need to have a single phone number for both voice and fax services. Incoming calls are initially answered by an auto attendant or interactive voice response (IVR) service. At this point, the media stream is monitored for fax tones. Calls identified as coming from a fax machine are then rerouted to a new destination, such as a fax server.

For Fax detection to work, the `cng-fax-detect` command under DSP farm and the `detect-fax` command must be configured in the inbound dial-peer. The fax detection feature may be configured to redirect calls to a local voice port or a remote application.
Fax detection on CUBE is also supported through a TCL script. The script answers an incoming call, plays a prompt and makes an outgoing voice or fax call. You can download the TCL script from the CiscoDevNet Github.

Local Redirect Mode

Local redirect may be used to transfer a fax call to either a local port or remote destination. Multiple destinations may be used if required, allowing the CUBE to hunt for the first available resource. The configured hunt list can include any number of destination ports.

*Figure 39: Local Redirect Call Flow*

An initial connection is made as a voice call through CUBE to the IVR. On detection of fax tones in the media path, CUBE closes the connection to the IVR, then hunts through a list of numbers to establish a connection to a fax machine or fax server, allowing the originating fax machine to complete its transmission. In a scenario where T.38 is not supported by CUBE, it will fallback to passthrough.

For each call, a digital signal processor (DSP) channel is allocated to detect the fax CNG tone. This DSP remains allocated until the original call leg clears at the end of the call. In the call flow example above, the first fax machine is busy, so the CUBE establishes the call with the second fax machine.
For Local Redirect, new calls legs are negotiated as voice, not as fax session.

Refer Redirect Mode

In this mode, calls are redirected to a fax service by the original calling party. The redirect is based on information provided by CUBE in a SIP Refer message (similar to a blind transfer).

In this mode, only one redirection target can be configured.

Figure 40: Refer Redirect Call Flow

An initial connection is made as a voice call through CUBE to the IVR. On detection of fax tones in the media path, CUBE closes the connection to the IVR. To transfer the call, CUBE first sends a re-invite to put the original call leg on hold, then sends a SIP REFER with details of the remote fax server. From this point, CUBE is no longer involved in the call flow as the originating fax communicates directly with the destination server.

For each call, a DSP channel or resource is allocated to detect the CNG tone. This resource is released once the call transfer has been initiated.
Transcoder Behavior for Cisco IOS XE

For the fax tone detection support offered for Cisco IOS XE, the DSP resource behavior for local and refer redirect is as follows:

- For local redirect, CUBE doesn’t release the transcoder until the fax call disconnects.
- For refer redirect, CUBE releases the transcoder when the REFER message is sent to the peer leg.

Fax Detection with Cisco IOS XE High Availability

Fax detection and transfer are supported with CUBE High Availability (HA) deployments. In this mode, two CUBE routers are configured to run in Active-Standby mode.

The following behaviors specific to this feature must be noted:

- Failover after initial call has been established, but fax hasn’t been detected—The call is preserved, but tone detection is not available for the remainder of that call. The originating fax machine terminates the call after CNG time-out.
- Failover after fax detection, but before the transferred call leg is established—The initial call is preserved and the transfer fails. The originating fax machine terminates the call after CNG time-out.

How to Configure Fax Detection for SIP Calls

Configure DSP Resource to Detect Fax Tone

**SUMMARY STEPS**

1. enable
2. configure terminal
3. ds pfarm profile tag transcode universal
4. cng-fax-detect
5. maximum sessions sessions
6. associate application CUBE
7. 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: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>
### Command or Action

| Step 2 | configure terminal  
| Example: |  
| | Device# configure terminal |  
| Purpose | Enters global configuration mode. |
| Step 3 | dspfarm profile  tag  transcode universal  
| Example: |  
| | Device(config)# dspfarm profile 5 transcode universal |  
| Purpose | Enters DSP farm profile configuration mode and enables the profile for transcoding. |
| Step 4 | cng-fax-detect  
| Example: |  
| | Device(config-dspfarm-profile)# cng-fax-detect |  
| Purpose | Enables CNG tone detection. |
| Step 5 | maximum sessions  sessions  
| Example: |  
| | Device(config-dspfarm-profile)# maximum sessions 6 |  
| Purpose | Configures maximum number of sessions. |
| Step 6 | associate application  CUBE  
| Example: |  
| | Device(config-dspfarm-profile)# associate application CUBE |  
| Purpose | Configures an application to the profile for LTI-based transcoding. |
| Step 7 | end  
| Example: |  
| | Device(config-dspfarm-profile)# end |  
| Purpose | Returns to privileged EXEC mode. |

### Dial-peer Configuration to Redirect Fax Call

#### SUMMARY STEPS

1. enable  
2. configure terminal  
3. dial-peer voice  number  voip  
4. description  tag  
5. session protocol  sipv2  
6. incoming called number  number  
7. voice-class codec  tag  
8. no  vad  
9. detect-fax  [ mode  \{ refer  number  \}  local  number  ]
## 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 |  
- Enter your password if prompted. |
| **Step 2** | configure terminal | Enters global configuration mode.  
**Example:**  
Device# configure terminal | |
| **Step 3** | dial-peer voice number voip | Enters dial peer configuration mode for the specified VoIP dial peer.  
**Example:**  
Device(config)# dial-peer voice 401 voip | |
| **Step 4** | description tag | Provides a description for the incoming dial-peer for Fax.  
**Example:**  
Device(config-dial-peer)# description Incoming dial-peer for Fax | |
| **Step 5** | session protocol sipv2 | Configures SIP as the session protocol type.  
**Example:**  
Device(config-dial-peer)# session protocol sipv2 | |
| **Step 6** | incoming called number number | Creates inbound dial-peer.  
**Example:**  
Device(config-dial-peer)# incoming called-number 903309 | |
| **Step 7** | voice-class codec tag | Applies the previously configured voice class and associated codecs to a dial peer. The voice class codec can only include g711ulaw and g711alaw.  
**Example:**  
Device(config-dial-peer)# voice-class codec 111 | |
| **Step 8** | no vad | Disables voice activity detection (VAD) for the calls using the dial peer being configured.  
**Example:**  
Device(config-dial-peer)# no vad | |
| **Step 9** | detect-fax [mode { refer|local}number] | Defines fax detection as local or refer mode and refers to the directory number of the fax machine.  
**Example:** | |
Verifying Fax Detection for SIP Calls

SUMMARY STEPS

1. enable
2. show call active voice compact
3. show dspfarm dsp active

DETAILS STEPS

Step 1  enable

Example:
Device> enable

Enables privileged EXEC mode.

Step 2  show call active voice compact

Example:
This is a sample output of call setup when the call is connected:

Device# show call active voice compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 3

<table>
<thead>
<tr>
<th>A/O</th>
<th>Codec</th>
<th>Peer Address</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAX</td>
<td>g711ulaw</td>
<td>P808808</td>
<td>9.42.25.145:17940</td>
</tr>
<tr>
<td>ORG</td>
<td>g711ulaw</td>
<td>P309903</td>
<td>9.42.25.149:16396</td>
</tr>
<tr>
<td>ORG</td>
<td>g711ulaw</td>
<td>P808808</td>
<td>9.42.25.149:16394</td>
</tr>
</tbody>
</table>

Step 3  show dspfarm dsp active

Example:
This is a sample output of the DSP channel reserved to detect CNG tone after the call is set up.

Device# show dspfarm dsp active

SLOT DSP VERSION STATUS CHNL USE TYPE RSC_ID BRIDGE_ID PKTS_TXED PKTS_RXED
0 2 36.1.0 UP 1 USED xcode 1 9 228 119

Purpose

If local mode is configured, then a list of numbers, separated by a space may be entered. Refer mode only allows a destination number to be configured.

Example:
Device(config-dial-peer)# detect-fax refer 12101

Step 10

end

Example:
Device(config-dial-peer)# end

Returns to privileged EXEC mode.
Troubleshooting Fax Detection for SIP Calls

You can enable the logs of the following debug or show commands, which are helpful in debugging fax detection for SIP calls:

- debug voip ipipgw all
- debug ccsip verbose
- debug voip ccapi all
- debug voip dsmpl all
- debug voip hpi all
- debug media resource provisioning all
- show call active voice compact
- show dspfarm dsp active
- show voip rtp connections

Configuration Examples for Fax Detection for SIP Calls

Example: Configuring Local Redirect

The following is a sample configuration in local redirect mode for fax detection. In this example, the dial-peer has to be configured for the FAX directory numbers 9033010 and 9033011.

dspfarm profile 10 transcode universal
codec g729abr8
codec g729ar8
codec g711alaw
codec g711ulaw
codec g729r8
codec ilbc
codec g722-64
cng-fax-detect
maximum sessions 6
associate application CUBE

dial-peer voice 401 voip
description "Incoming dial-peer to ASR"
session protocol sipv2
incoming called-number 903309
voice-class codec 111
dtmf-relay rtp nte
no vad
Example: Configuring Refer Redirect

In Refer mode, only one fax number can be configured.

dial-peer voice 401 voip
  description "Incoming dial-peer to ASR"
  session protocol sipv2
  incoming called-number 903309
  voice-class codec 111
  dtmf-relay rtp-nte
  no vad
  detect-fax mode refer 9033010

dial-peer voice 406 voip
  description "Outbound dialpeer for ..."
  destination-pattern 9033010
  session protocol sipv2
  session target ipv4:9.41.36.11:14762
  voice-class codec 111
  dtmf-relay rtp-nte
  fax protocol pass-through g711ulaw
  no vad

Feature Information for Fax Detection for SIP Call and Transfer

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 Fax Detection for SIP Call and Transfer

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fax Detection for SIP Call and Transfer</td>
<td>Cisco IOS 15.4(2)T</td>
<td>Fax detection is the capability to detect automatically whether an incoming call is voice or fax. For calls coming from an IP trunk to CUBE, the Fax Detection for SIP Call and Transfer feature is used to detect CNG tones (calling tones) so that the fax server can handle the actual fax transmission or redirect the fax call to a configured fax number. The following commands were introduced: <code>cng-fax-detect</code> and <code>detect-fax mode</code>.</td>
</tr>
<tr>
<td>Fax Detection for SIP Call and Transfer on Cisco IOS XE Platforms</td>
<td>Cisco IOS XE Amsterdam 17.2.1r</td>
<td>Support was introduced for SIP call and transfer for IP-to-IP calls on Cisco IOS XE platforms for Cisco Unified Border Element.</td>
</tr>
</tbody>
</table>
PART VI

Video

- Video Suppression, on page 465
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 465
- Restrictions, on page 465
- Information About Video Suppression, on page 466
- Configuring Video Suppression, on page 466
- Troubleshooting Tips, on page 467

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 56: Feature Information for Video Suppression

<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 (for T.38 Fax) media types and drops all other media types in SDP. The following commands are introduced: <strong>audio forced</strong>, <strong>voice-class sip audio forced</strong></td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE Denali 16.3.1</td>
<td></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>
</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 audio forced`  
  - In the global VoIP SIP configuration mode  
    `audio forced`  
  **Example:**  
  In dial-peer configuration mode  
  ```sh  
  ! Applying audio-forced to one dial peer only  
  Device (config)# dial-peer voice 10 voip  
  Device (config-dial-peer)# voice-class sip audio forced  
  Device (config-dial-peer)# end  
  ```  
  **Example:**  
  In global VoIP SIP configuration mode  
  ```sh  
  ! Applying audio forced globally  
  Device(config)# voice service voip  
  Device (config-voi-serv)# sip  
  Device (config-voi-sip)# audio forced  
  Device (config-voi-sip)# end  
  ``` |
| **Step 4** end | Exits present configuration mode and enters privileged EXEC mode. |

### 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 ccapi inout
PART VII

Media Services

• Configuring RTCP Report Generation, on page 471
CHAPTER 37

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.

- Prerequisites, on page 471
- Restrictions, on page 471
- Configuring RTCP Report Generation on Cisco UBE, on page 472
- Troubleshooting Tips, on page 473
- Feature Information for Configuring RTCP Report Generation, on page 474

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 SRTP 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 to-type
5. rtcp keepalive
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>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>voice service voip</td>
<td>Enters voice service configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
<td></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>4</td>
<td>allow-connections <em>from-type</em> to <em>to-type</em></td>
<td>Allows connections between SIP endpoints in a VoIP network.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(conf-voi-serv)# allow-connections sip to sip</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>rtcp keepalive</td>
<td>Configures RTCP keepalive report generation.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(conf-voi-serv)# rtcp keepalive</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>end</td>
<td>Exits voice service configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(conf-voi-serv)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Troubleshooting Tips

Use the following debug commands for debugging related to RTCP keepalive packets:

- **debug voip rtcp 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 rtcp packet
01:06:27.450: //6/xxxxxxxxxxxx/RTP//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 .........J..2E5AF220: 07111E02 010F302E 302E3040 392E3435 ....0.0.0@9.45
    2E5AF230: 2E33302E 33000000 00 ........0.3....
```

**Caution**

Under moderate traffic loads, the **debug voip rtcp packet** command produces a high volume of output and the command should be enabled only when the call volume is very low.

- **debug voip rtcp packet** -- Shows details about VoIP RTP packet debugging trace.

```
Router# debug voip rtcp packet
VOIP RTP All Packets debugging is on
```

- **debug voip rtp session** -- Shows all RTP session debug information.

```
```
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. An account on Cisco.com is not required.

Feature History Table entry for the Cisco Unified Border Element.

Table 57: 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: rtcp keepalive, debug voip rtcp, debug voip rtp, debug ip rtp protocol, and ip rtcp report interval.</td>
</tr>
</tbody>
</table>

Feature History Table entry for the Cisco Unified Border Element (Enterprise).

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.
Table 58: 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>rtp 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 479
- SIPREC (SIP Recording), on page 507
- Video Recording - Additional Configurations, on page 537
- Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 543
- Cisco Unified Communications Gateway Services--Extended Media Forking, on page 551
CHAPTER 38

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 479
- Restrictions for Network-Based Recording, on page 480
- Information About Network-Based Recording Using CUBE, on page 481
- How to Configure Network-Based Recording, on page 485
- Additional References for Network-Based Recording, on page 505

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>
<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 show sip-ua calls 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: media-type audio.</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>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

- RecordTone insertion is not supported with SRTP calls.
• Forking does not stop when RTP stream changes mid call to RTP stream. This is for backward compatibility.
• MediaForkingReason tag is to notify midcall stream events. Notification for codec change is not supported.
• Server Groups in outbound dial-peers towards recorders is not supported.

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:
1. Incoming call from SIP trunk.
2. Outbound call to a Contact Centre
3. Media between endpoints flow through 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 an 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]
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></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>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Device# configure terminal</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Step 3</strong> media profile recorder profile-tag</td>
<td>Configures the media profile recorder and enters media profile configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></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 video. If this configuration is not done, both audio and video are recorded.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# media-type audio</td>
<td></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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# media-recording 8000 8001 8002</td>
<td>Note: You can specify a maximum of five dial-peer tags.</td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> media class tag</td>
<td>Configures a media class and enters media class configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# media class 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> recorder profile tag</td>
<td>Configures the media profile recorder.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaclass)# recorder profile 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> exit</td>
<td>Exits media class configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaclass)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> 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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 8000 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong> media-class tag</td>
<td>Configures media class on a dial peer.</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-dial-peer)# media-class 100</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><strong>Step 12</strong></td>
<td></td>
</tr>
<tr>
<td><strong>destination-pattern</strong> [+] <strong>string</strong> [T]</td>
<td>The predefined valid entries for <strong>string</strong> are the digits 0 to 9, the letters A to F and, the following special characters:</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>- The asterisk (*) and pound sign (#) that appear on standard touch-tone dial pads.</td>
</tr>
<tr>
<td>Device(config-dial-peer)# destination-pattern 595959</td>
<td>- Plus sign (+), which indicates that the preceding digit occurred one or more times.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>- Backslash symbol (), which is followed by a single character, and matches that character.</td>
</tr>
<tr>
<td>Media Forking functionality does not work with the wildcard entries other than the predefined set.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td><strong>session protocol sipv2</strong> Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td><strong>session target ipv4:</strong>[recording-server-destination-address</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>- <strong>ipv4:</strong> destination address --IP address of the dial peer, in this format: xxx.xxx.xxx.xxx</td>
</tr>
<tr>
<td>Device(config-dial-peer)# session target ipv4:10.42.29.7</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td><strong>session transport tcp</strong> Configures a VoIP dial peer to use Transmission Control Protocol (TCP).</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session transport tcp</td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td><strong>end</strong> Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# end</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>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;Device&gt; enable&lt;br&gt;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;Device# configure terminal&lt;br&gt;Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>media class tag</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config)# media class 100&lt;br&gt;Configures the media class and enters media class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>recorder parameter</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(cfg-mediaclass)# recorder parameter&lt;br&gt;Enters media class recorder parameter configuration mode to enable you to configure recorder-specific parameters.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>(Optional) <strong>media-type audio</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(cfg-mediaclass)# media-type audio&lt;br&gt;Configures recording of audio only in a call with both audio and video.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# media-type audio</td>
<td>Note: 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

*Note:* 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

*Note:* Exits media class recorder parameter configuration mode.

**Step 8**

exit

*Example:*

Device(cfg-mediaclass)# exit

*Note:* Exits media class configuration mode.

**Step 9**

dial-peer voice dummy-recorder-dial-peer-tag voip

*Example:*

Device(config)# dial-peer voice 8000 voip

*Note:* 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

*Note:* Configures media class on a dial peer.

**Step 11**

destination-pattern [+ string [T]]

*Example:*

Device(config-dial-peer)# destination-pattern 595959

*Note:* Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer.
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.

### 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

### Step 14

**session transport tcp**

**Example:**

```
Device(config-dial-peer)# session transport tcp
```

Configures a VoIP dial peer to use Transmission Control Protocol (TCP).

### Step 15

**end**

**Example:**

```
Device(config-dial-peer)# end
```

Returns to privileged EXEC mode.

---

**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 voice 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 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. 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 :
No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP
1 1 2 16384 20918 10.104.45.191 10.104.8.94
2 2 1 16386 17412 10.104.45.191 10.104.8.98
3 3 4 16388 29652 10.104.45.191 10.104.8.98
4 4 3 16390 20036 10.104.45.191 10.104.8.94
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 : 7C5946D3BECD

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 : 1E01B6D000000
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-near-end 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-near-end 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 forked 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 ssr 0x53
```
Verifying the Network-Based Recording Using CUBE

- 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

<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 : C9A3A0-8499188-8018A94B-90B050010.0.0.1
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 : 7d1e2d62e5c504f9aa4ba29dd136
Remote UUID : 252eaa82f505c668037da95438fc49b
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 : bTqQXbgFjddA1hE9wJGV3aKxo5vPV+Z1234tVb2
Remote Crypto Key : bTqQXbgFjddA1hE9wJGV3aKxo5vPV+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
    AES_CM_128_HMAC_SHA1_32 }
Remote Crypto Suite : AES_CM_128_HMAC_SHA1_80
Local Crypto Key : bTqQXbgFjddA1hE9wJGV3aKxo5vPV+Z2345tVb2
Remote Crypto Key : bTqQXbgFjddA1hE9wJGV3aKxo5vPV+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: 0xC0401C 0x10000100 0x4
CC Call ID: 11
Local UUID: 2522ean2f505ce868037da95438fc49b
Remote UUID: 7d14e2e622ec504f9aaa4ba0292d136
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: bTQqZXbFjddA1h3E9wJGyV3aKx5VvP+Z9876tVb2
Remote Crypto Key: bTQqZXbFjddA1h3E9wJGyV3aKx5VvP+Z1234tVb2

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
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]:8018
Media Dest IP Addr: Port: [10.0.0.2]:5062
Local Crypto Suite: AES_CM_128_HMAC_SHA1_80
Remote Crypto Suite: AES_CM_128_HMAC_SHA1_80
Local Crypto Key: bTQqZXbFjddA1h3E9wJGyV3aKx5VvP+Z9876tVb2
Remote Crypto Key: bTQqZXbFjddA1h3E9wJGyV3aKx5VvP+Z1234tVb2
Remote Crypto Key: bTqQ2XbgFJddAhE9wJGV3aKxo5PVVz2345tVb2
Mid=Call Re-Association Count: 0
SRTP-RTP Re-Association DSP Query Count: 0

Options-Ping ENABLED:NO ACTIVE:NO
Number of SIP User Agent Server(UAS) calls: 1

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 (*12:23:53.573 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:0/0/0 delay:0/0/0ms H264 TextRelay: off
Transcoded: No

0 : 2 87424930ms.1 (*12:23:53.583 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:0/0/0 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:0/0/0 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=z9hG4bKBy622CF
X-Cisco-Recording-Participant: sip:708090@10.42.30.10;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:595959@10.42.29.7>
Date: Fri, 18 Mar 2011 07:01:50 GMT
Call-ID: 6E6CF813-506411E0-80EAE01B-3427AA62@10.42.30.10
Supported: 100rel, timer, resource-priority, replaces, sdp-anat
Min-SE: 1800
Cisco-Guid: 1334370502-1348997600-239669092-3395863316
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
Verifying the Network-Based Recording Using CUBE

Content-Length: 449
v=0
c=IN IP4 10.42.30.10
m=audio 24544 RTP/AVP 0 101 19
c=IN IP4 10.42.30.10
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
c=IN IP4 10.42.30.10
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:23232323810.104.46.201>;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:23232323810.104.46.201:5060;transport=tcp>
Content-Type: application/sdp
Allow: INVITE, BYE, CANCEL, ACK, NOTIFY, INFO, UPDATE
Server: Cisco-ORA/8.5
v=0
c=IN IP4 10.104.46.201
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:23232323810.104.46.201:5060;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:23232323810.104.46.201>;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: 1334370502-1348997600-239669092-3395863316</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: 0884935168-0000065536-000000401-3475859466

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

<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:575757@9.45.38.39>;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 ccsip 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:432189.45.38.39;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
c=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
Verifying the Network-Based Recording Using CUBE

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
a=rtpmap:0 PCMU/8000
a=ptime:20
a=recvonly

m=video 0 RTP/AVP 98
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 ccapi all
• debug voip fpi all (for ASR devices only)

Displays detailed debug messages.

For Audio:

Media forking initialized:


Media forking started:
Verifying the Network-Based Recording Using CUBE

Media Recording

*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_service_get_event_data: Event id = 30
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Function/sipSPIIsValidCcb:
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Function/ccsip_is_valid_ccb:
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking: MF: Current State = 1, event =30
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking: MF: State & Event combination is cracked..

Forking header populated:


Media forking setup record session is successful:

*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_get_recording_participant_header: MF: Sipuser = 98459845 - 9.42.30.34
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_get_recording_participant_header: MF: Host = 9.42.30.34
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Function/sipSPIGetFirstStream:
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Function/voip_media_dir_to_cc_media_dir:
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_BuildMediaRecStream: MF: dtmf is inband
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_write_to_TDContainer: MF: Data written to TD Container..
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/act_sip_mf_idle_callsetup_ind: MF: setup_record_session is success..

Media forking forked stream started:

*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_ipip_media_forking_forked_leg_config: MF: Overwriting the GUID with the value got from MSP.
Verifying the Network-Based Recording Using CUBE

Media Recording

For Video:
Media Forking Initialized:

Media forking started:

Recording participant for anchor leg:

Adding an audio stream:
data for audio stream..

*Mar 19 16:40:01.789 IST: //522/34BF0A000000/SIP/Info/info/32800/ccsip_ipip_media_forking_BuildAudioRecStream: MF: dtmf is inband

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>MediaSense Installation and Administration Guide</th>
<th>Cisco MediaSense Installation and Administration Guide</th>
</tr>
</thead>
</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 507
- Prerequisites for SIPREC Recording, on page 508
- Restrictions for SIPREC Recording, on page 508
- Information About SIPREC Recording Using CUBE, on page 509
- How to Configure SIPREC-Based Recording, on page 510
- Configuration Examples for SIPREC-based Recording, on page 515
- Configuration Example for Metadata Variations with Different Mid-call Flows, on page 521
- Configuration Example for Metadata Variations with Different Transfer Flows, on page 534
- Configuration Examples for Metadata Variations with Caller-ID UPDATE Flow, on page 535
- Configuration Example for Metadata Variations with Call Disconnect, on page 536

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: <code>recorder parameter</code> and <code>recorder profile</code>.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.17S</td>
<td></td>
</tr>
</tbody>
</table>
Prerequisites for SIPREC Recording

Make sure that:

- 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, see Network-Based Recording section.
- 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) pass through 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
- Server Groups in outbound dial-peers towards recorders is not supported.
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

You need to have:

- 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 42: 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 flowthrough 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 represents 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 profile-tag siprec
9. exit
10. dial-peer voice dp-tag voip
11. session protocol sipv2
12. media-class tag
13. dial-peer voice dial-peer-tag voip
14. destination-pattern [+] string [T]
15. session protocol sipv2
16. session target ipv4:[recording-server-destination-address | recording-server-dns]
17. session transport tcp
18. 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 profile recorder profile-tag</td>
<td>Configures the media profile recorder and enters media profile configuration mode.</td>
</tr>
<tr>
<td>Example: 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 video. If this configuration is not done, both audio and video are recorded.</td>
</tr>
<tr>
<td>Example: Device(cfg-mediaprofile)# media-type audio</td>
<td></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>Example: Device(cfg-mediaprofile)# media-recording 8000 8001 8002</td>
<td>Note You can specify a maximum of five dial-peer tags.</td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td>Example: Device(cfg-mediaprofile)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> media class tag</td>
<td>Configures a media class and enters media class configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# media class 100</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring SIPREC-Based Recording (with Media Profile Recorder)

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>recorder profile</strong> profile-tag siprec</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(cfg-mediaclass)# recorder profile 100 siprec</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Configures the media profile SIPREC recorder.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>exit</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(cfg-mediaclass)# exit</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Exits media class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>dial-peer voice dp-tag voip</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# dial-peer voice 1 voip</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Dial peer that needs to be forked.</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>session protocol sipv2</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td>media-class tag</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# media-class 100</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Configures media class on a dial peer.</td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>dial-peer voice dial-peer-tag voip</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# dial-peer voice 8000 voip</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Configures a recorder dial peer and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td>destination-pattern [+] string [T]</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# destination-pattern 595959</td>
</tr>
<tr>
<td><strong>Purpose</strong></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><strong>Step 15</strong></td>
<td>session protocol sipv2</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td>session target IPv4:[recording-server-destination-address</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Specifies a network-specific address for a dial peer. Keyword and argument are as follows:</td>
</tr>
<tr>
<td></td>
<td>• <strong>IPv4</strong>: destination address --IP address of the dial peer, in this format: xxx.xxx.xxx.xxx</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 dp-tag voip
10. session protocol sipv2
11. media-class tag
12. dial-peer voice dial-peer-tag voip
13. destination-pattern [+] string [T]
14. session protocol sipv2
15. session target ipv4:[recording-server-destination-address | recording-server-dns]
16. session transport tcp
17. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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>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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> media class <em>tag</em></td>
<td>Configures the media class and enters media class configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>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:</td>
<td></td>
</tr>
<tr>
<td>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>Device(cfg-mediaprofile)# media-type audio</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> media-recording <em>dial-peer-tag</em></td>
<td>Configures voice-class recording parameters.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaclass-recorder)# media-recording 8000, 8001, 8002</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> exit</td>
<td>Exits media class recorder parameter configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaclass-recorder)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> exit</td>
<td>Exits media class configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaclass)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> dial-peer voice <em>dp-tag</em> voip</td>
<td>Dial peer that needs to be forked.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 1 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> 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>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 11</strong> media-class <em>tag</em></td>
<td>Configures media class on a dial peer.</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# media-class 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong> dial-peer voice <em>dial-peer-tag</em> voip</td>
<td>Configures a recorder dial peer and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# dial-peer voice 8000 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 13</strong> destination-pattern [+] <em>string</em> [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>Example: Device(config-dial-peer)# destination-pattern 595959</td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong> session protocol sipv2</td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong> session target [ipv4: [recording-server-destination-address</td>
<td>recording-server-dns]</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# session target ipv4:10.42.29.7</td>
<td></td>
</tr>
<tr>
<td>• ipv4: destination address --IP address of the dial peer, in this format: xxx.xxx.xxx.xxx</td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</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 17</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
Example: Configuring SIPREC-based Recording without Media Profile Recorder

Router> enable
Router# configure terminal
Router(config)# media class 101
Router(cfg-mediaclass)# recorder profile 201 siprec

Validate SIPREC Functionality

Use the command `show voip rtp connections` to verify that media forking configuration is correct:

```
CUBE#show voip rtp connections
VoIP RTP active connections :
No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP
1 36 37 18358 19362 209.165.201.5 209.165.201.10
2 37 36 17294 17690 10.0.0.5 10.0.0.20
3 39 38 19812 42196 172.16.0.10 10.0.0.10
4 40 38 24230 60234 172.16.0.10 10.0.0.10
Found 4 active RTP connections
```

In this example, the call between the 2 phones has resulted into 2 RTP streams (1 and 2). The 2 RTP streams (3 and 4) are the recorded streams that are sent to the Recording Server (10.0.0.10 in this example). The call Recording Server receives a duplicated RTP stream that represents the recorded call. Use the command `show voip recmsp session` to verify:

```
CUBE#sh voip recmsp session
RECMSP active sessions:
MSP Call-ID AnchorLeg Call-ID ForkedLeg Call-ID
143 141 145
Found 1 active sessions
```

To get more details of the streams run the command `show voip recmsp session detail call-id <the value specified in the above o/p>`:

```
CUBE#show voip recmsp session detail call-id 143
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: 2001
Non-anchor Leg Details:
Call ID: 140
Forking Stream type: voice-farend
Participant: 1001
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

Where:

- **Stream State**: This state shows the state of the call – can be either in ACTIVE or HOLD state.

- **Anchor Leg Call-id**: This ID is the call-id of the anchor leg (Dial-peer where forking is enabled) which is also internal to the system. The output in brief describes the participant number and stream type as voice near-end, which is called party side.

- **Non-Anchor Call-id**: This ID is the call-id of nonanchor leg (Dial-peer where forking is not enabled).

- **Forked Call-id**: This forking leg call-id shows near-end and far-end stream call-id details with state of the Stream.

If you want to know the remote IPs and ports for the near-end and far-end legs, use the `show voip rtp forking` command:

```
CUBE#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 10.0.0.10, 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 10.0.0.10, 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
```

Remote IP/Port is the recording server ip and port address. Codec indicates which codec is negotiated to record the call leg. Packets that are sent indicate the number of packets that are sent to Recording Server from each stream.

**Troubleshoot**

The following is a sample SIPREC configuration on IOS/IOS-XE voice routers.

```
media class 777
recorder parameter siprec
media-recording 777
!
dial-peer voice 11 voip
description CUCM
session protocol sipv2
session target ipv4:10.0.0.15
destination e164-pattern-map 164
media-class 777
codec g711ulaw
!
dial-peer voice 777 voip
destination-pattern AAAA
session protocol sipv2
session target ipv4:10.0.0.10
codec g711ulaw
```
Working Scenario

After the call is connected, the inbound/outbound CCS SIP info legs helps to understand that a recording call has been initiated. In the following example, the outbound call leg 4536 posts a media forking start indication to its peer inbound leg 4535. This inbound leg ignores this event because it is not the anchor leg (in this example, media-class command is configured on the outgoing dial peer (Peer ID 4536)).

017895: May 13 15:32:45.273: //4536/2FD863BAA01F/SIP/Info/info/32768/ccsip_trigger_media_forking: MF: EO leg. set the pending flag. wait for peer leg to indicate start
017902: May 13 15:32:45.273: //4535/2FD863BAA01F/SIP/Info/verbose/32768/ccsip_event_handler: Ignoring the event on non-anchor leg

Similarly, the outbound call leg 4536 posts a media forking start indication to the inbound call leg 5435.

018221: May 13 15:32:45.290: //4536/2FD863BAA01F/SIP/Info/notify/32768/ccsip_event_handler: CC_EV_H245_MEDIA_FORKING_START_IND: peer ID 4535, event = 217 type = 1

Outbound leg processes the event and triggers the recording session.

018230: May 13 15:32:45.290: //-1/xxxxxxxxxxxx/Inout/recmsp_api_create_session: Exit with Success

Recording dial-peer lookup.

018390: May 13 15:32:45.296: //-1/xxxxxxxxxxxx/SIP/Info/verbose/5120/sipSPIGetOutboundHostAndDestHostPrivate: CCSIP: target_host : 10.0.0.10 target_port : 5060

Create XML metadata.


<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
<datamode>complete</datamode>
<session session_id="MigZ2nTLEemWFaQi1vyb4Q==">
<sipSessionID>a0b9b2a1e4db51f082e777c0df9015e5;remote=6bea155500105000a000231246a214b</sipSessionID>
<start-time>2019-05-13T15:32:45.293Z</start-time>
</session>
<participant participant_id="MThBMXTLEemWFq1lvyb4Q==-">
<nameID aor="sip:1234@10.0.0.15"/>
</participant>
<participants>
</participants>
</recording>
INVITE is sent to recorder with metadata in XML format where:

- The **nameID** attribute represents the name and SIP/SIPS/tel URI (also called the address of record) of each participant.

- The **participant_id** attribute indicates the unique ID assigned to each participant in the recording session.

- The **stream_id** attribute indicates the unique ID assigned to each media stream in the recording session.

- The **session_id** attribute is used to reference the communication session to which a given media stream belongs.

- The label metadata attribute provides the value of **a=label** attribute assigned to this media stream in the SDP of the SIP request and responses of the recording session. It plays a key role in associating a media stream with its metadata information.
Allow-Events: telephone-event
Session-ID: a62dd6d8be0059c38d142bae9b46880b;remote=00000000000000000000000000000000
Session-Expires: 1800
Content-Type: multipart/mixed;boundary=uniqueBoundary
Mime-Version: 1.0
Content-Length: 2470

--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required
v=0
c=CiscoSystemsSIP-GW-UserAgent 5511 2889 IN IP4 y.y.y.y
s=SIP Call
c=IN IP4 y.y.y.y
t=0 0
m=audio 8086 RTP/AVP 0 101 19
c=IN IP4 y.y.y.y
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 8088 RTP/AVP 0 101 19
c=IN IP4 y.y.y.y
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=sendonly
a=label:2

--uniqueBoundary
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">
  <datamode>complete</datamode>
  <session session_id="MIgZ2nTLEemWFaQi1vyb4Q==">
    <sipSessionID>a0b9b2a1e4db51f082e777c0df9015e5;remote=6bea155500105000a0002c31246a214b</sipSessionID>
    <start-time>2019-05-13T15:32:45.293Z</start-time>
  </session>
  <participant participant_id="MIhBMXTLEemWFq11vbyb4Q==">
    <nameID aor="sip:1234@10.0.0.15">
      <name xml:lang="en">Emergency</name>
    </nameID>
  </participant>
  <participantsessionassoc participant_id="MIhBMXTLEemWFq11vbyb4Q==" session_id="MIgZ2nTLEemWFq11vbyb4Q==">
    <associate-time>2019-05-13T15:32:45.293Z</associate-time>
  </participantsessionassoc>
  <stream stream_id="MIlSKnTLEemWG6Qii1vbyb4Q==" session_id="MIgZ2nTLEemWFq11vbyb4Q==">
    <label>1</label>
  </stream>
  <participant participant_id="MIhBMXTLEemWFq11vbyb4Q==">
    <nameID aor="sip:911@209.165.201.1">
      <name xml:lang="en">Emergency</name>
    </nameID>
  </participant>
  <participantsessionassoc participant_id="MIhBMXTLEemWFq11vbyb4Q==" session_id="MIgZ2nTLEemWFq11vbyb4Q==">
    <associate-time>2019-05-13T15:32:45.293Z</associate-time>
  </participantsessionassoc>
</recording>
Configuration Example for 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.

```
--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
```
Example: Complete SIP Recording Metadata Information Sent in INVITE or Re-INVITE

```
c=IN IP4 9.42.25.149
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=sendonly
a=label:1
m=audio 16554 RTP/AVP 8 101
```
```
c=IN IP4 9.42.25.149
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=sendonly
a=label:1
m=audio 16554 RTP/AVP 8 101
```
```
c=IN IP4 9.42.25.149
b=TIAS:1000000
a=rtpmap:119 H264/90000
a=fmtp:119 profile-level-id=42801E;packetization-mode=0
a=sendonly
a=label:3
m=video 16558 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
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
  <datamode>complete</datamode>
  <session session_id="JaPQeP1CDee6sYHx7yVg=="/>
    <sipSessionID>276ac102a3c05270a4375d9512ea11;remote=110b0c0f50775078b13d60be0044db11</sipSessionID>
    <start-time>2015-05-19T09:42:06.911Z</start-time>
  </session>
  <participant participant_id="JaPQeP1CDee6sYHx7yVg=="/>
    <nameID aor="sip:808808@9.0.0.174"/>
    <name xml:lang="en">808808</nameID>
  </participant>
  <participantsessionassoc participant_id="JaPQeP1CDee6sYHx7yVg==" session_id="JaPQeP1CDee6sYHx7yVg=="/>
    <associate-time>2015-05-19T09:42:06.911Z</associate-time>
  </participantsessionassoc>
  <stream stream_id="JaPQeP1CDee6sYHx7yVg==" label="1">
    <participant participant_id="JaPQeP1CDee6sYHx7yVg=="/>
      <nameID aor="sip:909909@9.0.0.174"/>
      <name xml:lang="en">909909</nameID>
    </participant>
  </stream>
  <stream stream_id="JaPQeP1CDee6sYHx7yVg==" label="3">
    <participant participant_id="JaPQeP1CDee6sYHx7yVg=="/>
      <nameID aor="sip:909909@9.0.0.174"/>
      <name xml:lang="en">909909</nameID>
    </participant>
  </stream>
  <stream stream_id="JaPQeP1CDee6sYHx7yVg==" label="4">
    <participant participant_id="JaPQeP1CDee6sYHx7yVg=="/>
      <nameID aor="sip:909909@9.0.0.174"/>
      <name xml:lang="en">909909</nameID>
    </participant>
  </stream>
</recording>
```
Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5

Example: Complete SIP Recording Metadata Information Sent in INVITE or Re-INVITE

<participantsessionassoc>
  <stream stream_id="JaPQeP1CEeSA86sYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
    <label>2</label>
  </stream>
</participantsessionassoc>

<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—

<table>
<thead>
<tr>
<th>Output 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 276ac102a3c05270a4375d99512ea1a1; remote=110b0c0f50775078b13d60be0044db11</td>
<td>This attribute carries a SIP Session-ID of the original call between the participants.</td>
</tr>
<tr>
<td>&lt;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>
</tbody>
</table>
| a=label:1;
  <stream stream_id="JaPQeP1CEeSA86sYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
    <label>1</label>
  </stream> | The <stream> element represents a Media Stream object. Stream element indicates the SDP media lines associated with the session and participants. The <label> element within the <stream> element references an SDP "a=label" attribute that identifies an m-line within the RS SDP. This m-line carries the media stream from the SRC to the SRS. |
<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>participantsessionassoc</td>
<td>Participant CS Association 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.</td>
</tr>
<tr>
<td>participant_id=&quot;JaPQeP1CEeSA76sYHx7YVg==&quot;</td>
<td></td>
</tr>
<tr>
<td>session_id=&quot;JaPQeP1CEeSA66sYHx7YVg==&quot;</td>
<td></td>
</tr>
<tr>
<td>participantsessionassoc</td>
<td>Participant CS Association class describes the association of the second participant to a CS for a period of time. A participant can associate and dissociate from a CS several times. The 'session_id' attribute helps to identify unique CS sessions.</td>
</tr>
<tr>
<td>participant_id=&quot;JaPQeP1CEeSA8qsYHx7YVg==&quot;</td>
<td></td>
</tr>
<tr>
<td>session_id=&quot;JaPQeP1CEeSA66sYHx7YVg==&quot;</td>
<td></td>
</tr>
<tr>
<td>participantstreamassoc</td>
<td>Participant stream association class describes the association of either participant 1 or 2 to a media stream for a period of time, as a sender or as a receiver, or both. These streams can be either audio or video or both.</td>
</tr>
<tr>
<td>participant_id=&quot;JaPQeP1CEeSA76sYHx7YVg==&quot;</td>
<td></td>
</tr>
<tr>
<td>participantstreamassoc</td>
<td></td>
</tr>
<tr>
<td>participant_id=&quot;JaPQeP1CEeSA8qsYHx7YVg==&quot;</td>
<td></td>
</tr>
</tbody>
</table>

**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. Here, in a normal recording session, both participants sent audio and video streams.

```plaintext
--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required

v=0
c=IN IP4 9.42.25.149
a=SIP Call
```
Example: Hold with Send-only / Recv-only Attribute in SDP
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:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
  ...
  <stream stream_id="jIBTUf1BEEsADoRg7YVg==" session_id="jH+2kf1BEEsADoRg7YVg=="/>
  ...
</recording>
```

Content-Type: application/rs-metadata+xml
Content-Disposition: recording-session

---

Example: Hold with Send-only / Recv-only Attribute in SDP
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.

```plaintext
--uniqueBoundary--

Content-Type: application/sdp
Content-Disposition: session;handling=required

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:1000000
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:1000000
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42801E;packetization-mode=0
a=sendonly
a=label:4
```
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.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
    ...  
    <stream stream_id="uV/B4f1BEeSAmKsYHx7YVg==" session_id="uV/B4f1BEeSAk6sYHx7YVg==">  
        <label>1</label>  
    </stream>  
    ...  
    <stream stream_id="uV/B4f1BEeSAmasYHx7YVg==" session_id="uV/B4f1BEeSAk6sYHx7YVg==">  
        <label>3</label>  
    </stream>  
    ...  
    <stream stream_id="uV/B4f1BEeSAm6sYHx7YVg==" session_id="uV/B4f1BEeSAk6sYHx7YVg==">  
        <label>2</label>  
    </stream>  
    ...  
    <stream stream_id="uV/B4f1BEeSAAl6sYHx7YVg==" session_id="uV/B4f1BEeSAk6sYHx7YVg==">  
        <label>4</label>  
    </stream>  
    <participantstreamassoc participant_id="uV/B4f1BEeSAl6sYHx7YVg==">  
    </participantstreamassoc>  
    <participantstreamassoc participant_id="uV/B4f1BEeSAmqsYHx7YVg==">  
    </participantstreamassoc>  
</recording>  
--uniqueBoundary--
```
After escalation, video streams get added into the `participantstreamassociation` element in metadata for both the participants. There will be 4 streams in total.

---

Content-Type: application/sdp

v=0
o=CiscoSystemsSIP-GW-UserAgent 6360 4789 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
t=0 0
m=audio 16628 RTP/AVP 18 101
c=IN IP4 9.42.25.149
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=sendonly
a=label:1
m=audio 16630 RTP/AVP 18 101
c=IN IP4 9.42.25.149
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
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.
Example: De-escalation
After de-escalation, video streams are removed from the metadata and only audio calls will be present in the `participantStream association` element.

```xml
<v=0 o=CiscoSystemsSIP-GW-UserAgent 7616 8309 IN IP4 9.42.25.149 s=SIP Call 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
```

```xml
<stream stream_id="j5OQdP1CEeSBTasYHx7YVg==" session_id="j5LOTP1CEeSBRasYHx7YVg==" label:1></label>
</stream>

<stream stream_id="j5OQdP1CEeSBTasYHx7YVg==" session_id="j5LOTP1CEeSBRasYHx7YVg==" label:2></label>
</stream>
```
Configuration Example for 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>
...

Configuaration Examples for 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+z2xM6EeWAIN4iOrLrag==">
  <nameID aor="sip:7774442214@10.104.54.52">
  </nameID>
</participant>
<participantsessionassoc participant_id="vm+z2xM6EeWAIN4iOrLrag==" session_id="vACJ+xM6EeWAP94iOrLrag==">
  <associate-time>2015-06-16T08:44:32.869Z</associate-time>
</participantsessionassoc>
<participant participant_id="vm+z2xM6EeWAIN4iOrLrag==">
  <nameID aor="sip:7774442218@10.104.54.52">
  </nameID>
</participant>
<participantsessionassoc participant_id="vm+z2xM6EeWAIN4iOrLrag==" session_id="vACJ+xM6EeWAP94iOrLrag==">
  <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+xM6EeWAP94iOrLrag==">
  <disassociate-time>2015-06-16T08:44:32.869Z</disassociate-time>
```
Configuration Example for 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">
<session session_id="t5nW8RM6EeWACN4iOrLrag==">
  <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 40

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 537
Information About Additional Configurations for Video Recording, on page 538
How to Configure Additional Configurations for Video Recording, on page 538
Verifying Additional Configurations for Video Recording, on page 541

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 60: 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following commands were introduced or modified: media profile video, ref-frame-req rtpc, 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><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></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> media profile video</td>
<td>Configures a video media profile and enters media profile configuration mode.</td>
</tr>
<tr>
<td>media-profile-tag</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# media profile video 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ref-frame-req rtcp retransmit-count</td>
<td>Enables FIR using the RTCP or SIP INFO method.</td>
</tr>
<tr>
<td>retransmit-number</td>
<td></td>
</tr>
<tr>
<td>ref-frame-req sip-info</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# ref-frame-req rtcp retransmit-count 4</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# ref-frame-req sip-info</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>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></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>Configures a video media profile and enters media profile configuration mode.</td>
</tr>
<tr>
<td>media profile video media-profile-tag</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# media profile video 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></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>h264-packetization-mode packetization mode</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(cfg-mediaprofile)# h264-packetization-mode 2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Exits media profile configuration mode.</td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>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 IDR.

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></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>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>2</td>
<td><code>configure terminal</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device# configure terminal</code></td>
</tr>
<tr>
<td>3</td>
<td><code>media profile video media-profile-tag</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device(config)# media profile video 1</code></td>
</tr>
<tr>
<td>4</td>
<td><code>monitor-ref-frames</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device(cfg-mediaprofile)# monitor-ref-frames</code></td>
</tr>
<tr>
<td>5</td>
<td><code>end</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>Device(cfg-mediaprofile)# end</code></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`
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

```plaintext
VideoRtcpIntraFrameRequestCount=1
VideoRtcpIntraFrameRequestCount=1

!CUBE does not generate FIR request on forked leg
VideoRtcpIntraFrameRequestCount=0

Step 3  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  show call active video | include VideoTimeOfLastReferenceFrame
Displays the timestamp of latest IDR frame.

Step 5  show call active video | include VideoReferenceFrameCount
Displays the number of IDR frames received on that call leg.
```
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 543
• Restrictions for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 544
• Information About Third-Party GUID Capture for Correlation Between Calls and SIP-based recording, on page 544
• How to Capture Third-Party GUID for Correlation Between Calls and SIP-based Recording, on page 544
• Verifying Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 547
• Configuration Examples for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording, on page 548

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.
Table 61: Feature Information for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording</td>
<td>Cisco IOS 15.4(3)M</td>
<td>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 via an established SIP session, making CUBE recording more interoperable with third-party vendors.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.13S</td>
<td></td>
</tr>
</tbody>
</table>

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>• 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:</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>Step</td>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>5</td>
<td>exit</td>
<td>Exits voice class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# exit</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>dial-peer voice inbound-dialpeer-tag voip</td>
<td>Enters inbound dial-peer 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>7</td>
<td>voice class sip-copylist tag</td>
<td>Applies the copy list to the dial peer.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# voice class sip-copylist 100</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>exit</td>
<td>Exits 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>
<tr>
<td>9</td>
<td>dial-peer voice outbound-dialpeer-tag voip</td>
<td>Enters outbound dial-peer configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 3 voip</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>voice class sip-copylist tag</td>
<td>Applies the copy list to the dial peer.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# voice class sip-copylist 100</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>exit</td>
<td>Exits 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>
<tr>
<td>12</td>
<td>voice class sip-profiles profile-id</td>
<td>Creates a SIP profile 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-profiles 10</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>request INVITE peer-header sip GUID-header-to-copy copy header-value-to-match copy-variable</td>
<td>Copies headers from the INVITE message of the incoming dial peer into a copy variable.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-class)# request INVITE peer-header sip Third-Party-GUID copy &quot;(.*)&quot; u01</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>request INVITE sip-header header-to-add add header-value-to-add</td>
<td>Adds a SIP header to a SIP request.</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>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# request INVITE sip-header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unsupported add &quot;Unsupported: Dummy Header&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>request INVITE sip-header GUID-header-to-modify</td>
<td>Modifies the outgoing header using the copy variable defined in the</td>
<td></td>
</tr>
<tr>
<td>modify header-value-to-match header-value-to-replace</td>
<td>previous step.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# request INVITE sip-header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unsupported modify &quot;.*&quot; &quot;Third-Party-GUID: \u01&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>Exits to global configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# exit</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 17</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dial-peer voice recorder-dial-peer-tag voip</td>
<td>Enters the dial peer configuration mode for the specified outbound</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>outlaw recorder dial peer.</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 2 voip</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 18</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>voice-class sip profiles profile-tag</td>
<td>Applies the SIP profile to the recording dial peer.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip profiles 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 19</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Exits to privileged EXEC mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></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:**

```
Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.44.29.32:5060;branch=z9hG4bK121F62
From: "sipp " <sip:1111000010@9.44.29.32>;tag=906F9C-21B9
To: "sut" <sip:4321@9.0.0.120>;tag=30050SIPpTag0111
Call-ID: 67B65D26-473711E3-8029B214-265DCDFE@9.44.29.32
CSeq: 101 INVITE
Contact: <sip:9.0.0.120:6019;transport=UDP>
Cisco-Guid: passthru
Content-Type: application/sdp
Content-Length: 108
```

```
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 42

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 551
• Restrictions for Extended Media FORKING, on page 552
• Information About Cisco Unified Communications Gateway Services, on page 552
• How to Configure UC Gateway Services, on page 558
• Configuration Examples for UC Gateway Services, on page 565

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 Extended Media Forking

The Extended Media Forking does not support the following:

- Media renegotiation.
- Media mixing on forked media streams.
- recordTone insertion with SRTP calls.
- mediaForkingReason tag is only to Notify midcall stream events; notification for events such as codec change.
- Supplementary services such as hold/resume, call forward, Call Transfer, and so on.
- High Availability.
- Virtual Routing and Forwarding (VRF) or Multi-VRF.
- Hair-pinning calls from CUBE to Cisco Unified Customer Voice Portal (CVP) and back to the same CUBE for Extended Media Forking (XMF) Gateway recording.
- Forking of calls on a TDM leg.

Note

- Supports only voice media stream.

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 a near-end stream and the stream from the called party is termed as a far-end stream.

The XMF provider actively handles single media forking request per session. Any new media forking request from the external application overrides or stops the current forking instance and initiates a new forking instance (to the appropriate target IP address or ports).

After accepting the media forking request, the XMF provider returns a response message and starts to fork media streams of a connection to the target forked streams. The application receives a NotifyXmfCallData message notification 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.

*Figure 43: XMF Connection-Based Media Forking*

![Diagram showing XMF Connection-Based Media Forking]

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.

**Extended 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 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.
  - RTP-to-SRTP call flow does not involve transcoding.
- 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 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 primary 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</td>
<td>a=crypto:1 AES_CM_128_HMAC_SHA1_32</td>
</tr>
<tr>
<td>inline:NzB4d1BINUAvLEw6UzF3WSJp+PsFdGtUjShpXIzj</td>
<td>inline:NzB4d1BINUAvLEw6UzF3WSJp+PsFdGtUjShpXIzj</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 45: 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`
**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.  
  | Example:          | • Enter your password if prompted.  
  |       | Device> enable    |         |
| 2    | configure terminal | Enters global configuration mode.  
  | Example:          |         |
  |       | Device# configure terminal |         |
| 3    | ip http server    | Enables the HTTP server (web server) on the system.  
  | Example:          |         |
  |       | Device(config)# ip http server |         |
| 4    | ip http max-connections value | Sets the maximum number of concurrent connections to the HTTP server that will be allowed. The default value is 5.  
  | Example:          |         |
  |       | Device(config)# ip http max-connection 100 |         |
| 5    | ip http timeout-policy idle seconds life seconds requests value | Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are:  
  | Example:          |         |
  |       | Device(config)# ip http timeout-policy idle 600 life 86400 requests 86400 | • idle—The maximum number of seconds the connection will be kept open if no data is received or response data can not be sent out on the connection. Note that a new value may not take effect on any already existing connections. If the server is too busy or the limit on the life time or the number of requests is reached, the connection may be closed sooner. The default value is 180 seconds (3 minutes).  
  | |         |
  |       |         | • life—The maximum number of seconds the connection will be kept open, from the time the
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>http client connection idle timeout  seconds</td>
<td>sets the number of seconds that the client waits in the idle state until it closes the connection.</td>
</tr>
<tr>
<td>uc wsapi</td>
<td>enters cisco unified communication IOS Service configuration mode.</td>
</tr>
<tr>
<td>message-exchange max-failures number</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>
<tr>
<td>probing max-failures number</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>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 10</strong> probing interval keepalive  seconds</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-uc-wsapi)# probing interval keepalive 255</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>• RESPONSE_XMF_REGISTER</td>
</tr>
<tr>
<td></td>
<td>• RESPONSE_XMF_CONN_MEDIA_FORKING</td>
</tr>
<tr>
<td></td>
<td>• SOLICIT_XMF_PROBING</td>
</tr>
<tr>
<td></td>
<td>• NOTIFY_XMF_CONNECTION_DATA</td>
</tr>
<tr>
<td><strong>Step 11</strong> probing interval negative  seconds</td>
<td>Configures the interval between negative probing messages, in seconds.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-uc-wsapi)# probing interval negative 10</td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong> source-address ip-address</td>
<td>Configures the IP address (hostname) as the source IP address for the UC IOS service.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-uc-wsapi)# source-address 192.1.12.14</td>
<td><strong>Note</strong> The source IP address is used by the provider in the NotifyProviderStatus messages.</td>
</tr>
<tr>
<td><strong>Step 13</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-uc-wsapi)# end</td>
<td></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>
</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**
  uc wsapi                 | Enters Cisco Unified Communication IOS Service configuration mode.       |
  **Example:**
  Device(config)# uc wsapi  |
| **Step 4**
  source-address **ip address** | Configures the source ip address.                                      |
  **Example:**
  Device(config)# source-address 172.156.19.38 |
| **Step 5**
  provider xmf             | Enters XMF provider configuration mode.                                 |
  **Example:**
  Device(config-uc-wsapi)# provider xmf |
| **Step 6**
  no shutdown              | Activates XMF provider.                                                |
  **Example:**
  Device(config-uc-wsapi)# no shutdown |
| **Step 7**
  remote-url **index url** | 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. |
  **Example:**
  Device(config-uc-wsapi)# remote-url 1 http://test.com:8090/ucm_xmf |
| **Step 8**
  end                      | Returns to privileged EXEC mode.                                        |
  **Example:**
  Device(config-uc-wsapi)# end |

### 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
appState: 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

registration index: 1
id: 2E7C304A:XMF:myapp:27
appUrl: http://pascal-lnx.cisco.com:8092/xmf
appState: 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

registration index: 21
id: 2E7C6423:XMF:myapp:28
appUrl: http://pascal-lnx.cisco.com:8096/xmf
appState: 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

registration index: 31
Step 3  **show wsapi registration xmf  remote-url-index**

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

<table>
<thead>
<tr>
<th>registration index: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>id: 2E7C6423:XMF:myapp:28</td>
</tr>
<tr>
<td>appUrl: <a href="http://pascal-lnx.cisco.com:8096/xmf">http://pascal-lnx.cisco.com:8096/xmf</a></td>
</tr>
<tr>
<td>appName: myapp</td>
</tr>
<tr>
<td>provUrl: <a href="http://9.45.46.16:8090/cisco_xmf">http://9.45.46.16:8090/cisco_xmf</a></td>
</tr>
<tr>
<td>prober state: STEADY</td>
</tr>
</tbody>
</table>

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!

<table>
<thead>
<tr>
<th>Session</th>
<th>Call</th>
<th>n/f</th>
<th>Destination (port address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>187</td>
<td>BA</td>
<td>near 45864 10.104.105.232</td>
<td></td>
</tr>
<tr>
<td>188</td>
<td>BA</td>
<td>far 54922 10.104.105.232</td>
<td></td>
</tr>
<tr>
<td>189</td>
<td>B9</td>
<td>near 45864 10.104.105.232</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>B9</td>
<td>far 54922 10.104.105.232</td>
<td></td>
</tr>
</tbody>
</table>

**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)
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>
      </callData>
      <connData>
        <connID>132</connID>
      </connData>
      <event>
        <mediaForking>
          <mediaForkingState>FORK_DONE</mediaForkingState>
        </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>

Troubleshooting Tips

Use the following debug commands to troubleshoot the UC Gateway Services configurations.

- debug wsapi infrastructure 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 569
CUBE Media Proxy

CUBE Media Proxy is a solution that provides multiple forking function, and is built on CUBE architecture. Multiple forks are required for recorder redundancy and advanced media processing needs. The CUBE Media Proxy solution supports mandatory and optional recorders.

CUBE Media Proxy supports Unified CM Network-Based Recording (NBR) and SIP-Based Media Recording (SIPREC), to enable forking and recording of Real-Time Transport Protocol (RTP) streams.

- Feature Information for CUBE Media Proxy, on page 569
- Supported Platforms, on page 570
- Restrictions for CUBE Media Proxy, on page 570
- CUBE Media Proxy Using Unified CM Network-Based Recording, on page 571
- SIPREC-Based CUBE Media Proxy, on page 571
- About Multiple Media Forking Using CUBE Media Proxy, on page 571
- Secure Forking of Secure and Nonsecure Calls, on page 572
- Deployment Scenarios for CUBE Media Proxy, on page 572
- Recording Metadata, on page 575
- Session Identifier, on page 577
- Recording State Notification, on page 579
- How to Configure CUBE Media Proxy, on page 582
- Verification of CUBE Media Proxy Configuration, on page 587
- Supported Features, on page 598

Feature Information for CUBE Media Proxy

The following table provides release information about the feature or features that are 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. You do not require an account on Cisco.com.
Table 62: Feature Information for Recording Proxy

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure forking of nonsecure</td>
<td>Cisco IOS XE Bengaluru 17.5.1a</td>
<td>CUBE Media Proxy supports both secure and nonsecure forking of nonsecure calls.</td>
</tr>
<tr>
<td>calls</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIPREC-Based CUBE Media Proxy</td>
<td>Cisco IOS XE Amsterdam 17.3.1a</td>
<td>The SIPREC-based CUBE Media Proxy solution supports forking to multiple recorders.</td>
</tr>
<tr>
<td>CUBE Media Proxy</td>
<td>IOS XE Gibraltar Release 16.10.1a</td>
<td>The CUBE Media Proxy solution provides multiple forking functions 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 (ISR4321, ISR4331, ISR4351, ISR4431, ISR4451, and ISR4461)
- Cisco Aggregated Services Routers (ASR - ASR1001-X, ASR1002-X, ASR1004 with RP2, ASR1006 with RP2, Cisco ASR1006-X Aggregated Services Routers with RP2 and ESP40, ASR 1006-X with RP3 and ESP40/ESP100)
- Cisco Cloud Services Routers (CSR1000V series)
- Cisco Catalyst 8000V Edge Software (Catalyst 8000V) series
- Cisco 8300 Catalyst Edge Series Platforms
- Cisco 8200 Catalyst Edge Series Platform (C8200-1N-4T)
- Cisco 8200L Catalyst Edge Series Platform (C8200L-1N-4T)

**Note**

When upgrading to C8000V software from a CSR1000V release, an existing throughput configuration will be reset to a maximum of 250Mbps. Install an HSEC authorization code, which you can obtain from your Smart License account, before reconfiguring your required throughput level.

**Restrictions for CUBE Media Proxy**

CUBE Media Proxy using Unified CM NBR, and SIPREC-Based CUBE Media Proxy do not support the following:

- Forking of video sessions
- Recording of calls from endpoints that are registered with the Cloud. For example, Cisco Webex Calling.
- SRTP fallback
• Midcall block
• Concurrent use with CUBE B2BUA SBC features.
• Server Groups in outbound dial-peers toward recorders.
• 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.

• Unified CM NBR and SIPREC for the same call flow.

The following restriction applies when using CUBE Media Proxy with Unified CM NBR:
• If the primary recorder sends a=inactive in the response SDP, the same is forwarded to Unified CM. Forking is not triggered to any of the recorders.

CUBE Media Proxy Using Unified CM Network-Based Recording

CUBE Media Proxy using Unified CM Network-Based Recording (NBR), is Unified CM dependent and requires you to configure inbound dial-peers from Unified CM. After receiving a media forking request from Unified CM, the CUBE Media Proxy establishes media forks to the configured targets.

SIPREC-Based CUBE Media Proxy

The SIPREC (SIP Media Recording) feature supports media recording for Real-Time Transport Protocol (RTP) streams in compliance with section 3.1.1. of RFC 7245, with CUBE Media Proxy acting as the Session Recording Client (SRC). SIP is used to establish a Recording Session between the CUBE Media Proxy and recorders (or any other media application).

For SIPREC solutions, CUBE Media Proxy accepts an inbound RTP fork from a CUBE SBC and replicates this RTP fork to multiple SIPREC targets based on its inbound configuration.

About Multiple Media Forking Using CUBE Media Proxy

Unified CM Network-Based CUBE Media Proxy and SIPREC-Based CUBE Media Proxy support the following functions:
• Media forking for up to five destinations per call
• Destination redundancy by hunting algorithm
• Media fork policy control
• Load balancing during initial call setup
• High Availability
Secure Forking of Secure and Nonsecure Calls

From Cisco IOS XE Bengaluru 17.5.1a onwards, you can configure a combination of secure and nonsecure forks for a nonsecure call.

CUBE Media Proxy Using Unified CM Network-Based Recording, on page 571 supports secure forking of secure and nonsecure calls.

Note
You cannot use the mandatory policy command with secure forking configurations.

For SRTP pass through to work in secure media forking, the Command Line Interface srtp pass-thru should be configured at global or dial-peer level.

Deployment Scenarios for CUBE Media Proxy

Note
From Cisco IOS XE Bengaluru 17.5.1a onwards, you can deploy a combination of secure and nonsecure destinations.

CUBE Media Proxy Using Unified CM Network-Based Recording

In Network Based Recording (NBR) deployments, Cisco Unified Communications Manager establishes an initial forked media leg with CUBE Media Proxy. This may either be from a phone using its built-in bridge (Deployment Scenario for CUBE Media Proxy Using Unified CM NBR for Internal Call), or from a CUBE SBC using the eXtended Media Forking (XMF) API (Deployment Scenario for CUBE Media Proxy Using Unified CM NBR for External Call).
The information flow is as follows:

1. External or internal call is set up between the endpoints.

2. CUBE Media Proxy receives the media forking request from UCM.

3. CUBE Media Proxy sets up sessions with the recorders based on the proxy policy.
   - Mandatory recorder: Proxy policy is configured to set a recorder as mandatory. CUBE Media Proxy tries to establish 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 tries to establish a connection with the remaining recorders even if any of the recorders fail.
SIPREC-Based CUBE Media Proxy

CUBE Media Proxy may be configured to fork media autonomously using SIPREC, as shown in the following scenario.

1. CUBE SBC receives a call from a SIP trunk and routed to the intended destination.
2. CUBE SBC uses SIPREC to establish a media fork of the call with CUBE Media Proxy.
3. CUBE Media Proxy uses SIPREC to establish secure or nonsecure media forks with up to five destinations.

The information flow in this scenario is as follows:
On receiving BYE from the primary secure recorder, Media Proxy disconnects all secure and nonsecure recording sessions. BYE received from any other recorder, secure or nonsecure, will not impact other active recording sessions.

**Recording Metadata**

Metadata is the information that a Recording Server (RS) receives from a Recording Client (RC) in a SIP session. Metadata has the following functions:

- Carries the communication session data that describes the call to the Recording Server.
- Identifies the participants list.
- Identifies the session and media association time.

**Recording Metadata in CUBE Media Proxy Using Unified CM NBR**

Unified CM passes information about the forked call to CUBE Media Proxy in up to 16 metadata parameters that are included in the `From` header of the SIP Invite. CUBE Media Proxy includes a copy of this metadata in the Invite it sends to the configured destinations. The following is an example of a `From` header with metadata.

**Note** The `From` header, including all metadata must not exceed 583 bytes.

Following is a sample SIP header of a recording request:

```
From: "abcd" <sip:198101@10.200.25.137;
x-nearend;x-refci=27298699;x-nearendclusterid=NY-NJ-Labcluster;
x-nearenddevice=SEP2834A28318ECE;
x-nearendaddr=198101;x-farendrefci=27298699;
x-farendclusterid=NY-NJ-Labcluster;x-farenddevice=AFIFIM-VI1;x-farendaddr=172001;
x-sessionid=696dd5d3f7755c53abdc438e95db51febf;
tag=14087-b35a5915-3167-4d6a-8710-c121221502b2f-27298703
```

**Recording Metadata in SIPREC-Based CUBE Media Proxy**

The initial SIPREC Invite from CUBE to CUBE Media Proxy, and the SIPREC Invite from CUBE Media Proxy to the recorders, includes recording metadata in a SIPREC XML body.

Following is a sample SIPREC INVITE:

```
INVITE sip:9876@8.43.33.203:5060 SIP/2.0
Via: SIP/2.0/UDP 8.43.33.205:5060;branch=z9hG4bK20959B
From: <sip:8.43.33.209>;tag=678813-6AC
To: <sip:9876@8.43.33.203>
Date: Thu, 13 Feb 2020 03:35:19 GMT
Call-ID: B0FA2851-4D4811EA-82E5D263-E98F802408.43.33.209
```
Supported: 100rel, timer, resource-priority, replaces, sdp-anat
Require: siprec
Min-SE: 1800
Cisco-Guid: 2967454021-1296568810-2195116643-3918495780
User-Agent: Cisco-SIPGateway/IOS-17.3.20200207.160928
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1581564919
Contact: <sip:8.43.33.209:5060>;+sip.src
Expires: 180
Allow-Events: telephone-event
Session-ID: 812ee4f57c50b39e897d75d8e12809;remote=0000000000000000000000000000000000000000
Content-Type: multipart/mixed;boundary=uniqueBoundary
Mime-Version: 1.0
Content-Length: 2250

--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session; handling=required

v=0
c=IN IP4 8.43.33.209
a-label:1
m=audio 8278 RTP/AVP 0
c=IN IP4 8.43.33.209
a-label:2
m=audio 8280 RTP/AVP 0
c=IN IP4 8.43.33.209

--uniqueBoundary
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"><datamode>complete</datamode><session session_id="sPVtz01IEeqC3dJj6Y+AJA=="><sipSessionID>0e0960d88013509f86e7ad2d78da208a;remote=4d0de1325c205fa08f77d8d31c1b3a6f</sipSessionID><start-time>2020-02-13T03:35:19.008Z</start-time></session><participant participant_id="sPVtz01IEeqC3tJj6Y+AJA=="><nameID aor="sip:3478@8.41.17.71"></nameID></participant><participantsessionassoc participant_id="sPVtz01IEeqC3tJj6Y+AJA==" session_id="sPVtz01IEeqC3dJj6Y+AJA=="><associate-time>2020-02-13T03:35:19.008Z</associate-time></participantsessionassoc><stream stream_id="sPsgFkk1IEeqC49Jj6Y+AJA==" session_id="sPVtz01IEeqC3dJj6Y+AJA=="><label>1</label></stream>
For a SIPREC call, the Require header in the SIP Invite (from Cisco UBE to CUBE Media Proxy, and from CUBE Media Proxy to the recorders) must have a "siprec" extension. The Require header must also have metadata in the XML body, else the call is dropped. The Contact header in a SIP invite has a "+sip.src" extension.

### Session Identifier

In both NBR and SIPREC modes, CUBE Media Proxy uses the Session-ID header in request and response messages to exchange session identifiers for tracking a recording session between peers.

The Session-ID comprises of the following 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.

### Session-ID Handling

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.
- Response to Unified CM (Network-Based Recording) or CUBE (SIPREC-Based).

The following events are involved in the Session-ID handling by CUBE Media Proxy:

1. The initial Invite received by CUBE Media Proxy includes a local UUID generated by the originating platform and a null remote UUID as shown in the following example.
2. When sending an Invite to the primary recorder, CUBE Media Proxy generates a new UUID to use for the local Session Identifier. The remote UUID remains null.

3. The subsequent 200 OK response from the primary recorder includes a local session identifier that it generated and the UUID provided by CUBE Media Proxy in the Invite as the remote session identifier.

4. When sending a 200 OK to the originating platform, CUBE Media Proxy uses the UUID it generated as the local session identifier and the UUID it received initially as the remote session identifier.

5. CUBE Media Proxy sends a forking request to the remaining four recorders with Session-ID header containing the same locally generated UUID as the local UUID and a "NULL" value for the remote UUID.

6. CUBE Media Proxy receives 200OK 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. In NBR mode, CUBE Media Proxy sends a SIP Info Message to Unified CM. For more information on SIP Info Message, see SIP Info Messages from CUBE Media Proxy to Unified CM, on page 579. The Session-ID header of the SIP Info Message contains locally generated UUID by CUBE Media Proxy as local UUID and the UUID of Unified CM as the remote UUID.
Recording State Notification

SIP Info Messages from CUBE Media Proxy to Unified CM

After trying or establishing an NBR session with the recorders, the CUBE Media Proxy sends SIP Info message to Unified CM to provide the consolidated status of all the recorders.

A SIP Info message is sent during the following stages of a recording session:

1. Initial Call: After receiving a 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 the status of any of the recorders changes during a call, another SIP Info message with status of each recorder is sent to the initiator of the recording session. A change in status may result from any of the recorders sending a "BYE" or rejecting a midcall RE-INIVITE.

Note: The examples in the following sections illustrate CUBE Media Proxy forking to two of the maximum five destinations.

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 following is the XML format of a SIP info message.

```
<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 63: 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>
The primary recorder in a secure forking scenario functions the same way as a mandatory recorder functions in a nonsecure forking scenario except that the `recorderType` tag is shown as optional. The following is the XML format of a SIP INFO message in a combination of secure and nonsecure forking scenario:

```xml
<recorderList>
  <recorder>
    <recorderType>Optional</recorderType>
    <status>Success</status>
  </recorder>
  <recorder>
    <recorderType>Optional</recorderType>
    <status>Success</status>
  </recorder>
  <recorder>
    <recorderType>Optional</recorderType>
    <status>Success</status>
  </recorder>
  <recorder>
    <recorderType>Optional</recorderType>
    <status>Success</status>
  </recorder>
  <recorder>
    <recorderType>Optional</recorderType>
    <status>Success</status>
  </recorder>
</recorderList>
```

### 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 Configure CUBE Media Proxy, on page 584.

The SIP Info Message that is sent during a recording session depends on the scenarios that are given in the following table:

<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 <code>recorder-1</code> is established and forking to <code>recorder-2</code> is triggered successfully.</td>
<td>&lt;success&gt;</td>
<td>&lt;success&gt;</td>
</tr>
<tr>
<td>Call to the primary recorder <code>recorder-1</code> is established and forking to <code>recorder-2</code> is rejected with 503 Service Unavailable.</td>
<td>&lt;success&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
--- | --- | ---
Call to the primary recorder recorder-1 is established and there is no response from recorder-2 to the forking request. | <success> | <failure>
Call to the recorder recorder-1 and recorder-2 is rejected with 503 Service Unavailable. | <failure> | <failure>
There is no response from recorder-1 or recorder-2 are down. | <failure> | <failure>
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.
- In all failure scenarios, an error code is sent in the <errormessage>.

---

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 Configure CUBE Media Proxy, on page 584.

The SIP Info Message that is sent during a recording session depends on the scenarios that are given in the following table.

Table 65: Call Scenarios and Recorder Status During the Initial Call with a Mandatory Recorder

Scenario | <status> of recorder-1 in a SIP Info Message | <status> Of recorder-2 in a SIP Info Message
--- | --- | ---
Call to the mandatory recorder recorder-1 is established and forking to the optional recorder recorder-2 is triggered successfully. | <success> | <success>
### How to Configure CUBE Media Proxy

#### How to Configure CUBE Media Proxy for Network-Based Recording Solutions

Following are the steps to configure CUBE Media Proxy for Network-Based Recording solutions:

**Configure Outbound Dial-Peers to the Recorders**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice recorder-dial-peer-tag voip
4. destination-pattern [+|] string
5. session protocol sipv2
6. session target ipv4:[recording-server-destination-address | recording-server-dns]
7. session transport [udp|tcp|tls]
### 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>Device&gt; enable</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>Device# configure terminal</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Configures a recorder dial peer and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>dial-peer voice recorder-dial-peer-tag voip</td>
<td></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>Specifies either the prefix or full E.164 number required to reach the recorder. A destination pattern must not include regular expressions in this case.</td>
</tr>
<tr>
<td>destination-pattern [+</td>
<td>string</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# destination-pattern 595959</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td>session protocol sipv2</td>
<td></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>Specifies the target network address for the recorder. Keyword and argument are as follows:</td>
</tr>
<tr>
<td>session target ipv4: [recording-server-destination-address</td>
<td>recording-server-dns]</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>Configures a VoIP dial peer to use TCP. Using the session transport command, you can also configure UDP and TLS protocols.</td>
</tr>
<tr>
<td>session transport [udp</td>
<td>tcp</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# session transport tcp</td>
</tr>
</tbody>
</table>
Configure CUBE Media Proxy

Before you begin

For secure forking, outbound dial peers must be configured for TLS or SRTP. For further information, refer to Configuring CUBE for SIP TLS.

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. media-recording proxy secure [dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5]
6. proxy policy mandatory dial-peer-tag
7. exit
8. media class tag
9. recorder profile tag
10. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 8</strong></td>
<td>(Optional) voice-class sip srtp crypto &lt;crypto-tag&gt; OR srtp pass-thru</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)#voice-class sip srtp crypto 20 OR Device(config-dial-peer)#srtp pass-thru</td>
</tr>
<tr>
<td></td>
<td>Configures SRTP crypto profile on the dial-peer. OR Configures the SRTP pass through on the outbound dial-peer for incoming INVITE.</td>
</tr>
<tr>
<td></td>
<td>• This step is optional and is required only for secure media forking. • The voice-class sip srtp crypto &lt;crypto-tag&gt; is configured for RTP-SRTP Interworking. • The srtp pass-thru is configured for SRTP-SRTP pass through.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>end</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# end</td>
</tr>
<tr>
<td></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
### Purpose

**Command or Action**

| Device> enable |

**Step 2**

**configure terminal**

**Example:**

Device# configure terminal

**Step 3**

**media profile recorder**  profile-tag

**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]

**Example:**

Device(cfg-mediaprofile)# media-recording proxy 8000 8001 8002

**Step 5**

**media-recording proxy secure**  [dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5]

**Example:**

Device(cfg-mediaprofile)# media-recording proxy secure 9000 9001 9002

**Step 6**

**proxy policy mandatory**  dial-peer-tag

**Example:**

Device(cfg-mediaprofile)# proxy policy mandatory 8001

---

**Note**

From Cisco IOS XE Bengaluru 17.5.1a onwards, CUBE Media Proxy supports both secure and nonsecure forking. You can configure the dial-peers for both secure and nonsecure forking. The permitted number of configured secure and nonsecure dial peers for forking is five. The behaviour in Cisco IOS XE Bengaluru 17.4.1a and earlier releases is unchanged if there are no secure dial peers configured.

**Note**

- All secure dial peers must use the same voice class srtp-crypto profile.

**Note**

- The proxy policy mandatory command cannot be used when dial peers are configured using media recording proxy secure command.

- Only one mandatory dial peer may be configured for each profile.

- The mandatory dial peer must be one of those configured with the media-recording proxy command.
Purpose
Command or Action

Step 7
exit
Example:
Device(cfg-mediaprofile)# exit
Exits media profile configuration mode.

Step 8
media class tag
Example:
Device(config)# media class 100
Configures a media class and enters media class configuration mode.

Step 9
recorder profile tag
Example:
Device(cfg-mediaclass)# recorder profile 100
Configures the media profile recorder.

Step 10
exit
Example:
Device(cfg-mediaclass)# exit
Exits media class configuration mode.

Configure Inbound Dial-Peer from Unified CM

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. (Optional) srtpp pass-thru
7. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</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>Step 2</td>
<td>Eneters global configuration mode.</td>
</tr>
<tr>
<td>configure terminal</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
</tbody>
</table>
### How to Configure CUBE Media Proxy for SIPREC Solutions

Following are the steps to configure SIPREC-based CUBE Media Proxy:

1. Configure Outbound Dial-Peers to the Recorders, on page 582.
2. Configure CUBE Media Proxy, on page 584.
3. Configure SIPREC on CUBE. For more information, see SIPREC (SIP Recording).

### Verification of CUBE Media Proxy Configuration

You can verify the configuration of CUBE Media Proxy using Unified CM NBR and SIPREC-Based CUBE Media Proxy using the following `show` and `debug` commands.

- `debug voip fpi all` (for ASR devices only)
- `debug voip ccapi all`
• debug voip recmsp all

• debug ccsip all

• debug ccsip 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.

• show voip rtp connections

Displays Real-Time Transport Protocol (RTP) connections.

Example:

For CUBE Media Proxy with Unified CM NBR, recording sessions consist of two sets of RTP streams that are set up independently for near-end and far-end streams. The following example shows RTP connections from 198.51.100.1 is forked to three recorders 8.41.17.71 to 73.

This example shows NBR with 3 recorders. Two inbound INVITEs (one each for near-end or far-end).

Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 8
Port range not configured
Min Max Ports Ports

Media-Address Range Port Port Available Reserved In-use
Global Media Pool 8000 48198 19999 101 8

VoIP RTP active connections :
No. CallId dstCallId LocalRTP RmtRTP LocalIP RemoteIP MPSS VRF
1 100 101 8218 8372 198.51.100.1 192.0.2.1 NO NA
2 101 100 8220 9000 8.43.21.69 8.41.17.71 NO NA
3 104 103 8222 9238 8.43.21.69 8.41.17.72 NO NA
4 107 106 8224 9250 8.43.21.69 8.41.17.73 NO NA
5 108 109 8226 8374 198.51.100.1 192.0.2.1 NO NA
6 109 108 8228 9002 8.43.21.69 8.41.17.71 NO NA
7 112 111 8230 9240 8.43.21.69 8.41.17.72 NO NA
8 115 114 8232 9252 8.43.21.69 8.41.17.73 NO NA

Found 8 active RTP connections

For CUBE Media Proxy using SIPREC, both near-end and far-end streams are established with the same inbound INVITE, which includes the detail in 2 m-lines. The following example shows how the inbound RTP connections are established before creating the RTP connections for five forks.

This example shows SIPREC with 5 recorders. One inbound INVITE (both near-end or far-end streams).

Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 12
Port range not configured
Min Max Ports Ports

Media-Address Range Port Port Available Reserved In-use
Global Media Pool 8000 48198 19999 101 12
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>200</td>
<td>202</td>
<td>8108</td>
<td>6012</td>
<td>198.51.100.1</td>
<td>192.0.2.1</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>201</td>
<td>203</td>
<td>8110</td>
<td>6014</td>
<td>198.51.100.1</td>
<td>192.0.2.1</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>202</td>
<td>200</td>
<td>8112</td>
<td>6004</td>
<td>8.43.21.69</td>
<td>8.41.17.71</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>4</td>
<td>203</td>
<td>201</td>
<td>8114</td>
<td>8882</td>
<td>8.43.21.69</td>
<td>8.41.17.71</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>5</td>
<td>208</td>
<td>204</td>
<td>8116</td>
<td>6000</td>
<td>8.43.21.69</td>
<td>8.41.17.72</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>6</td>
<td>209</td>
<td>204</td>
<td>8118</td>
<td>8886</td>
<td>8.43.21.69</td>
<td>8.41.17.72</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>7</td>
<td>212</td>
<td>205</td>
<td>8120</td>
<td>6008</td>
<td>8.43.21.69</td>
<td>8.41.17.73</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>8</td>
<td>213</td>
<td>205</td>
<td>8122</td>
<td>9990</td>
<td>8.43.21.69</td>
<td>8.41.17.73</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>9</td>
<td>216</td>
<td>206</td>
<td>8124</td>
<td>6024</td>
<td>8.43.21.69</td>
<td>8.41.17.74</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>10</td>
<td>217</td>
<td>206</td>
<td>8126</td>
<td>9978</td>
<td>8.43.21.69</td>
<td>8.41.17.74</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>11</td>
<td>220</td>
<td>207</td>
<td>8128</td>
<td>6016</td>
<td>8.43.21.69</td>
<td>8.41.17.75</td>
<td>NO</td>
<td>NA</td>
</tr>
<tr>
<td>12</td>
<td>221</td>
<td>207</td>
<td>8130</td>
<td>9968</td>
<td>8.43.21.69</td>
<td>8.41.17.75</td>
<td>NO</td>
<td>NA</td>
</tr>
</tbody>
</table>

Found 12 active RTP connections

• show voip recmsp session

Displays active recording Media Service Provider (MSP) session information internal to CUBE Media Proxy.

Following is the sample output for CUBE Media Proxy using Unified CM NBR or SIPREC-Based CUBE Media Proxy:

Device# show voip recmsp session

RECMSP active sessions:

<table>
<thead>
<tr>
<th>MSP Call-ID</th>
<th>AnchorLeg Call-ID</th>
<th>ForkedLeg Call-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>99</td>
<td>107</td>
</tr>
<tr>
<td>104</td>
<td>99</td>
<td>111</td>
</tr>
<tr>
<td>105</td>
<td>99</td>
<td>115</td>
</tr>
<tr>
<td>106</td>
<td>99</td>
<td>119</td>
</tr>
</tbody>
</table>

Found 4 active sessions

• show voip recmsp session detail call-id call-id

 Displays detailed information about the recording MSP Call ID.

Example:

Following is the sample output for CUBE Media Proxy using Unified CM NBR:

Device# show voip recmsp session detail call-id 104

RECMSP active sessions:

Detailed Information

---------------------------
Recording MSP Leg Details:
Call ID: 103
GUID : 7C5946D38ECD

AnchorLeg Details:
Call ID: 100
Forking Stream type: voice-nearend
Participant: 10000

Non-anchor Leg Details:
Call ID: 101
Forking Stream type: voice-farend
Participant: 708090

Forked Leg Details:
Call ID: 104
In SIPREC-based CUBE Media Proxy, there are two voice near-end streams for the forked call leg.

Following is the sample output:

```
Device# show voip recmsp session detail call-id 208
RECMSP active sessions:
Detailed Information
-------------------------------
Recording MSP Leg Details:
Call ID: 204
GUID : C710812A808A

AnchorLeg Details:
Call ID: 200
Forking Stream type: voice-nearend
Participant: sipp

Non-anchor Leg Details:
Call ID: 202
Forking Stream type: voice-farend
Participant: 9876

Forked Leg Details:
Call ID: 208
Voice Near End Stream CallID 208
Stream State ACTIVE
Voice Near End Stream CallID 209
Stream State ACTIVE
Found 1 active sessions
```

• `show voip rtp forking`

Displays RTP media-forking connections.

**Example:**

Following is the sample output for CUBE Media Proxy using Unified CM NBR:

```
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 8.41.17.72, remote port 9238, local port 8222
    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
Fork 2
  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 8.41.17.73, remote port 9250, local port 8224
    codec g711ulaw, logical ssrc 0x53
    packets sent 29687, packets received 0
```
Following is the sample output for SIPREC-Based CUBE Media Proxy:

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 2
    remote ip 8.41.17.72, remote port 6000, local port 8116
    codec g711ulaw, logical ssrc 0x53
    packets sent 29687, packets received 0
    remote ip 8.41.17.72, remote port 8886, local port 8118
    codec g711ulaw, logical ssrc 0x53
    packets sent 1296, 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
Fork 2
  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 2
    remote ip 8.41.17.73, remote port 6008, local port 8120
    codec g711ulaw, logical ssrc 0x53
    packets sent 29687, packets received 0
remote ip 8.41.17.73, remote port 9990, local port 8122
codec g711ulaw, logical ssrc 0x53
packets sent 1296, 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

Fork 3
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 2
remote ip 8.41.17.74, remote port 6024, local port 8124
codec g711ulaw, logical ssrc 0x53
packets sent 29687, packets received 0
remote ip 8.41.17.74, remote port 9978, local port 8126
codec g711ulaw, logical ssrc 0x53
packets sent 1296, 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

Fork 4
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 2
remote ip 8.41.17.75, remote port 6016, local port 8128
codec g711ulaw, logical ssrc 0x53
packets sent 29687, packets received 0
remote ip 8.41.17.75, remote port 9968, local port 8130
codec g711ulaw, logical ssrc 0x53
packets sent 1296, 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

• show call active voice compact

Displays a compact version of voice CallsInProgress. An extra call leg is displayed for media forking.

Example:

Following is a sample using NBR:

Device# show call active voice compact
<callID> A/OFAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 8

100 101 104 107 108 109 112 115
ANS ORG ORG ORG ANS ORG ORG ORG
T644 T644 T643 T643 T642 T642 T641 T641
g711ulaw g711ulaw g711ulaw g711ulaw g711ulaw g711ulaw g711ulaw g711ulaw
VOIPVOIPVOIPVOIPVOIPVOIPVOIP
P10000 P708090 P708090 P708090 P10000 P708090 P708090 P708090
192.0.2.1:8372 8.41.17.71:9000 8.41.17.72:9238 8.41.17.73:9250 192.0.2.1:8374 8.41.17.71:9002 8.41.17.72:5240 8.41.17.72:5252
Following is a sample output using SIPREC:

Device# show call active voice compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 6
  200 ANS T644 g711ulaw VOIP P10000 192.0.2.1:8108
  202 ORG T644 g711ulaw VOIP P708090 8.41.17.71:8112
  208 ORG T643 g711ulaw VOIP P708090 8.41.17.72:8116
  212 ORG T643 g711ulaw VOIP P708090 8.41.17.73:8120
  216 ORG T643 g711ulaw VOIP P708090 8.41.17.74:8124
  220 ORG T643 g711ulaw VOIP P708090 8.41.17.75:8128

• show sip-ua calls

Displays active user agent client (UAC) and user agent server (UAS) information on SIP calls.

Example:

Following is the sample output for CUBE Media Proxy using Unified CM NBR:

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 : ab9f4823802156aaaa8d62e044aa2b96
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
Following is the sample output for SIPREC-based CUBE Media Proxy:

```
Device# show sip-ua calls
Total SIP call legs:6, User Agent Client:5, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID : C711BA13-7E9B11EA-8090D6ED-255EEFA088.43.21.69
  State of the call : STATE_ACTIVE (7)
  Substate of the call : SUBSTATE_NONE (0)
  Calling Number : sip:9876@8.41.17.71:8881
  Called Number : 9876
  Called URI : sip:9876@8.41.17.71:8881
  CC Call ID : 101
  Local UUID : eeabf35db3d25ca4b8276616d85d5d15
  Remote UUID : 8afa5ed7b8a052e29235bade4affc9e
  Source IP Address (Sig ) : 8.43.21.69
  Destn SIP Req Addr:Port : [8.41.17.71]:8881
  Destn SIP Resp Addr:Port: [8.41.17.71]:8881
  Destination Name : 8.41.17.71
  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 : 101
  Stream Type : voice+dtmf (1)
  Stream Media Addr Type : 1
  Negotiated Codec : g711ulaw (160 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: [8.43.21.69]:8112
  Media Dest IP Addr:Port : [8.41.17.71]:6005

Media Stream 2
  State of the stream : STREAM_ACTIVE
  Stream Call ID : 102
  Stream Type : voice+dtmf (1)
  Stream Media Addr Type : 1
  Negotiated Codec : g711ulaw (160 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: [8.43.21.69]:8114
  Media Dest IP Addr:Port : [8.41.17.71]:8883
  Mid-Call Re-Assocation Count: 0
  SRTP-RTP Re-Assocation DSP Query Count: 0
```

Options-Ping ENABLED:NO ACTIVE:NO

• show voip fpi calls
Displays the call (both inbound and outbound leg) information at the application level.

**Example:**

Following is the sample output for CUBE Media Proxy using Unified CM NBR:

```
Device# show voip fpi calls
Number of Calls : 1
------------- ------------ ------------ ------------
confID correlator AcallID BcallID state event
------------- ------------ ------------ ------------
1005 1 1019 1020 ALLOCATED DETAIL_STAT_RSP
```

As there are 2-m lines in the incoming invite to SIPREC-based CUBE Media Proxy, two FPI sessions are created. Following is the sample output:

```
Device# show voip fpi calls
Number of Calls : 2
------------- ------------ ------------ ------------
confID correlator AcallID BcallID state event
------------- ------------ ------------ ------------
42 13 102 100 ALLOCATED DETAIL_STAT_RSP
41 14 99 101 ALLOCATED DETAIL_STAT_RSP
```

**• 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 nonsecure. The "SIPREC" field in the command output is tagged Y for SIPREC-based recording session and N for Unified CM-based recording session.

**Example:**

```
Device# show media-proxy sessions
No. Call-ID Session-ID Dialpeer Tag (Y/N)
Inbound/Forked LocalUuid;RemoteUuid Tag (Y/N)
-------------------------------
1 36770/- a234a20672ce596d969c59ee9767f127; 3 N
```

**• 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:**

```
NBR:
Device# show media-proxy sessions summary
No Sessions Inbound/Forked Dialpeer-Tag IP:Port Total/Failed
-------------------------------
1 Forked 100 ipv4:8.41.17.71:5060 2/0
2 Forked 200 ipv4:8.41.17.72:5060 2/0
3 Forked 300 ipv4:8.41.17.73:5060 2/0
4 Inbound 5678
```
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</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Forked</td>
<td>100</td>
<td>ipv4:8.41.17.71:5060</td>
<td>1/0</td>
</tr>
<tr>
<td>2</td>
<td>Forked</td>
<td>200</td>
<td>ipv4:8.41.17.72:5060</td>
<td>1/0</td>
</tr>
<tr>
<td>3</td>
<td>Forked</td>
<td>300</td>
<td>ipv4:8.41.17.73:5060</td>
<td>1/0</td>
</tr>
<tr>
<td>4</td>
<td>Forked</td>
<td>400</td>
<td>ipv4:8.41.17.74:5060</td>
<td>1/0</td>
</tr>
<tr>
<td>5</td>
<td>Forked</td>
<td>500</td>
<td>ipv4:8.41.17.75:5060</td>
<td>1/0</td>
</tr>
<tr>
<td>6</td>
<td>Inbound</td>
<td>5678</td>
<td></td>
<td>1/0</td>
</tr>
</tbody>
</table>

**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 101
CC Call-ID: 100 Inbound-leg
Dur: 00:00:15 tx: 0/0 rx: 1484/296800 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 192.0.2.1:8372 Local-Addr: 192.0.2.1:8218 rtt:0ms pl:0/0ms
Dialpeer-Tag: 5678 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: ab9f4823802156aaaa8d62e04aaa2b96

CC Call-ID: 101 Forked-leg (Primary)
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.71:9000 Local-Addr: 8.43.21.69:8220 rtt:0ms pl:0/0ms
Dialpeer-Tag: 100 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 104 Forked-leg
Dur: 00:00:15 tx: 8.41.17.72:9238 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 200 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: dcd882f0876890b930f3427be7fa5f6

CC Call-ID: 107 Forked-leg
Dur: 00:00:15 tx: 1479/295800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.73:9250 Local-Addr: 8.43.21.69:8224 rtt:0ms pl:0/0ms
Dialpeer-Tag: 300 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: 8df0863a6434263f60e50124da649e6

**show media-proxy sessions session-id**

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: 100 Inbound-leg
Dur: 00:00:15 tx: 0/0 rx: 1484/296800 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 192.0.2.1:8372 Local-Addr: 192.0.2.1:8218 rtt:0ms pl:0/0ms
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 Cisco Unified Communications Manager.

Example:

Device# show media-proxy sessions metadata-session-id 696dd5d3f7755c6abdc438e93d01febf

CC Call-ID: 108 Inbound-leg
Dur: 00:00:46 tx: 0/0 rx: 3105/578880 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 192.0.2.1:8374 Local-Addr: 198.51.100.1:8226 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 1 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 528b282b045f0d099eaba3696c00de2 RemoteUUID: 4fd0836613424366fe00521d46ea16e3

Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5

CUBE Media Proxy

Verification of CUBE Media Proxy Configuration

Dialpeer-Tag: 5678 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: ab9f4823802156aaa8d62e04aaa2b96

LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: ab9f4823802156aaa8d62e04aaa2b96

LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: 8df0863a6434263f60e50124da7e2fa6

LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: dcdf882f0876890b930f3427be7fa5f6

LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: 8df0863a6434263f60e50124da7e2fa6

LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: ab9f4823802156aaa8d62e04aaa2b96

LocalUUID: ab9f4823802156aaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 104 Forked-leg
Dur: 00:00:15 tx: 1480/296000 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.71:9000 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 200 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 107 Forked-leg
Dur: 00:00:15 tx: 1479/295800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.72:9238 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 300 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 108 Forked-leg
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.71:9000 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 100 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 112 Forked-leg
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.72:9238 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 200 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 115 Forked-leg
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.72:9238 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 300 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 116 Forked-leg
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.72:9238 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 400 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9
Supported Features

Mid-Call Message Handling

CUBE Media Proxy using Unified CM NBR or SIPREC support midcall signaling events that involve RE-INVITEs from the initiator of the recording session (Unified CM or Cisco UBE) to the recorders. CUBE Media Proxy handles the RE-INVITEs that request a session refresh, change in SDP for media address, direction or codec, or change SRTP crypto suite/key.

For NBR solutions, CUBE Media Proxy sends status updates of a midcall event to Unified CM using SIP Info messages.

When CUBE Media Proxy establishes a new set of forked sessions, the first is referred to as the primary. Where a destination is configured as mandatory, the destination is always the primary. Where all destinations are optional, the first successfully created session is the primary.

Perform the following steps to handle midcall messages:

1. On receipt of a RE-INVITE, CUBE Media Proxy sends the RE-INVITE to the primary recorder.

2. If the primary destination responds to the RE-INVITE with a BYE, then:
   - If the primary is mandatory, the call and all forks are stopped by sending BYE to the destinations and originator.
   - If the primary is optional, the BYE is acknowledged, but not passed back to the originator. The primary session is maintained in a dormant state and further midcall updates are blocked for the remainder of the call.

3. For other responses, the message from the primary is sent to the originator (Unified CM or CUBE).

4. Where the RE-INVITE requests a change in SDP or SRTP and only if this is successfully acknowledged (200 OK) by the primary, the RE-INVITE is sent to the other destinations.

5. If any of the other destinations respond to the RE-INVITE with a failure, CUBE Media Proxy clears that fork by sending a BYE to that destination. The status of this failed session is provided to Unified CM in an INFO message in NBR configurations.

Secure Recording of Secure Calls and Nonsecure Calls

Secure Recording of Secure Calls

With CUBE Media Proxy using Unified CM NBR, it is possible to extend encrypted calls to forked destinations. In this scenario, call signaling is secured using TLS for each connection between CUBE Media Proxy and Unified CM and recorders. As SRTP passthrough is used for media flows, the cipher suite and encryption key negotiated between Unified CM and the primary destination is used for all forks.

Refer to Configuring SIP TLS to secure signaling on Unified CM and forked legs. SRTP configuration is only required for the Unified CM.

Secure Recording of Nonsecure Calls

Secure Recording of Nonsecure Calls

With CUBE Media Proxy using Unified CM NBR, it is possible to extend encrypted calls to forked destinations. In this scenario, call signaling is secured using TLS for each connection between CUBE Media Proxy and Unified CM and recorders. As SRTP passthrough is used for media flows, the cipher suite and encryption key negotiated between Unified CM and the primary destination is used for all forks.

Secure Recording of Nonsecure Calls

Secure Recording of Nonsecure Calls

With CUBE Media Proxy using Unified CM NBR, it is possible to extend encrypted calls to forked destinations. In this scenario, call signaling is secured using TLS for each connection between CUBE Media Proxy and Unified CM and recorders. As SRTP passthrough is used for media flows, the cipher suite and encryption key negotiated between Unified CM and the primary destination is used for all forks.

Secure Recording of Nonsecure Calls
From Cisco IOS XE Bengaluru 17.5.1a, CUBE Media Proxy used in NBR or SIPREC mode may be configured to secure specific forked sessions when the original call is not encrypted. In this case, the primary destination must be secured and is treated in the same way as a mandatory destination as described in the message handling section above. Refer to SIP TLS and SRTP-RTP internetworking

**Support for High Availability**

CUBE Media Proxy may be run on a high availability pair of platforms to ensure that calls and media forks are maintained if hardware failure. Call and forked session state is continuously synchronized between the platforms, ensuring that the standby can seemlessly take over media forwarding and call control if necessary.

High availability is available for Cisco Media Proxy configured for Unified CM NBR or SIPREC using either box-to-box or inbox redundancy options.

The following conditions apply when using CUBE Media Proxy high availability:

- Both Active and Standby platforms must have a common hardware and software configuration.
- Calls are synchronized by establishing a checkpoint with the standby on completion of each INVITE, REINVITE, UPDATE, or BYE message transaction.
- Connections that are not successfully established at the point of switchover are not maintained (as there is no checkpoint for the incomplete message transaction).
- In Unified CM NBR mode, checkpoint information includes call metadata, SRTP context and common session ID for all forked sessions. Checkpoints are created after message flows between a recorder and Unified CM are complete. For example, when an optional recorder sends a BYE, the checkpoint is created after CUBE Media Proxy receives the 200 OK response from Unified CM for the INFO message it sends.
- In SIPREC mode, checkpoint information includes common session ID, but not metadata.

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`
- `show media-proxy sessions`
- `show media-proxy sessions summary`
- `show sip-ua calls`

**Media Latch**

By default, CUBE Media Proxy using Unified CM NBR uses 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 match with the details in the SDP sent by the SIP User Agent. Packets without matching IP address and port are dropped.

In a typical SCCP-based BiB recording using Unified CM NBR CUBE Media Proxy, Unified CM 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. Unified CM then sends this SDP to the SCCP phone. The CUBE Media Proxy does not know
the BiB 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 BiB of the SCCP phone to SCCP phone, are different from the IP address
and port details in the packets that are sent from Unified CM to the CUBE Media Proxy.

Media Latching is enabled on Unified CM NBR 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 that flows through the CUBE Media Proxy, and works for
initial and midcall scenarios. Media Latching is enabled on the inbound leg (Unified CM 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 603
- Copying SIP Headers, on page 605
- Manipulate SIP Status-Line Header of SIP Responses, on page 611
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 603
- Example: Passing a Header Not Supported by CUBE, on page 603

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>
<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 other. 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>

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 45

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 605
- How to Copy SIP Header Fields to Another, on page 606
- Example: Copying the To Header into the SIP-Req-URI, on page 609

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>
<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 other. 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. Configure a SIP profile to copy the 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 outbound-dial-peer-tag voip
14. voice-class sip-profiles profile-id
15. exit

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></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>enters global configuration mode.</td>
</tr>
<tr>
<td>Step 3</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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice class sip-copylist 100</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| **Step 4** | Do one of the following:  
- `sip-header header-name`  
- `sip-header SIP-Req-URI` | Specifies the SIP header to be copied to the peer call leg.  
- `sip-req-uri`—Configures Cisco Unified Border Element (UBE) to send a SIP request Uniform Resource Identifier (URI) to the peer call leg.  
- `header-name`—Configures Cisco Unified Border Element (UBE) to send the header name specified to the peer call leg. |
| **Step 5** | `exit` | Exits voice class configuration mode. |
| **Step 6** | `dial-peer voice inbound-dial-peer-tag voip` | Enters the dial peer configuration mode for the specified inbound dial peer. |
| **Step 7** | `voice-class sip-copylist tag` | Applies the copy list to the dial-peer. |
| **Step 8** | `exit` | Exits to global configuration mode. |
| **Step 9** | `voice class sip-profiles profile-id` | Create a SIP Profile and enters voice class configuration mode. |
| **Step 10** | `{request | response} message peer-header sip header-to-copy copy header-value-to-match copy-variable` | Copies headers from the corresponding incoming dial peer into a copy variable. |
| **Step 11** | `{request | response} message {sip-header | sdp-header} header-to-modify modify header-value-to-match header-value-to-replace` | Modifies an outgoing SIP or SDP header using the copy variable defined in the previous step. |
| **Step 12** | `exit` | Exits to global configuration mode. |
| **Step 13** | `dial-peer voice outbound-dial-peer-tag voip` | Enters the dial peer configuration mode for the specified outbound dial peer. |
### 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>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
</tr>
<tr>
<td>Purpose</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><code>configure terminal</code></td>
</tr>
<tr>
<td>Purpose</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>voice class sip-profiles profile-id</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# voice class sip-profiles 10</td>
</tr>
<tr>
<td>Purpose</td>
<td>Creates a SIP profile and enters voice class configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>`{request</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-class)# request INVITE sip-header TO copy &quot;sip:(''&quot;''@&quot;&quot;) u01</td>
</tr>
<tr>
<td>Purpose</td>
<td>Copies the contents of the specified header from an outbound message into a copy variable.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>`{request</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Purpose</td>
<td>Modifies an outgoing SIP or SDP header using the copy variable defined in the previous step.</td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config-class)# request INVITE sip-header</td>
<td></td>
</tr>
<tr>
<td>SIP-Req-URI modify &quot;.<em>@(.</em>)&quot; &quot;INVITE sip:\u01@\1&quot;</td>
<td></td>
</tr>
</tbody>
</table>

### Step 6

**Example:**

```bash
Device(config-class)# end
```

Exits voice class configuration mode and enters privileged EXEC mode.

### 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 49: Incoming SIP Message**

```
INVITE sip:43565432A5@192.168.1.100:5060 SIP/2.0
From: <sip:027784200@A.eu;user=phone>
To: <sip:25555552@A.eu>
...
```

Given below is the SIP message that is required. Note that 43565432A5 has changed to 25555552 in the SIP INVITE.

**Figure 50: Modified SIP Message**

```
INVITE sip:25555552@192.168.1.100:5060 SIP/2.0
From: <sip:027784200@A.eu;user=phone>
To: <sip:25555552@A.eu>
...
```

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
Manipulate 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 611
- Copying Incoming SIP Response Status Line to Outgoing SIP Response, on page 612
- Modifying Status-Line Header of Outgoing SIP Response with User Defined Values, on page 615

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>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Profile Enhancements for SIP responses and error codes</td>
<td>15.4(1)T</td>
<td>This feature extends SIP profiles to allow the following:</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE</td>
<td>• Modification of the outgoing SIP response status line.</td>
</tr>
<tr>
<td></td>
<td>Release 3.12S</td>
<td>Previously, only modification of outgoing SIP requests and responses was possible.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Before applying a SIP profile to a response from CUBE, the response can be mapped to its corresponding request.</td>
</tr>
</tbody>
</table>
### Feature Information

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Support for conditional header manipulation of SIP headers | 15.1(3)T Cisco IOS XE Release 3.6S | This feature allows users to copy content from one header to the other. 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`

### Copying Incoming 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 51: 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

---

[Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5](#)
### 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><strong>Example:</strong></td>
<td>Device&gt; enable</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>voice class sip-copylist tag</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# voice class sip-copylist 1</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>sip-header SIP-StatusLine</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-class)# sip-header SIP-StatusLine</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>exit</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-class)# exit</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>dial-peer voice inbound-dial-peer-id voip</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config)# dial-peer voice 99 voip</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>voice-class sip copy-list list-id</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# voice-class sip copy-list 1</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>exit</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Device(config-dial-peer)# exit</td>
</tr>
</tbody>
</table>
## SIP Header Manipulation

### Copying Incoming SIP Response Status Line to Outgoing SIP Response

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 9</strong></td>
<td>voice class sip-profiles tag</td>
<td>Enables dial peer-based VoIP SIP profile configurations and enters voice class configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td>Device(config)# voice class sip-profiles 10</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>response response-code peer-header sip SIP-StatusLine copy match-pattern copy-variable</td>
<td>Copies responses from the corresponding incoming call leg into a copy variable.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></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>response response-code sip-header SIP-StatusLine modify match-pattern copy-variable</td>
<td>Modifies an outgoing response using the copy variable defined in the previous step.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></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>exit</td>
<td>Exits voice class configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></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

**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>&lt;br&gt;enable&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device&gt; enable</td>
<td>Enables privileged EXEC mode.&lt;br&gt;- Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong>&lt;br&gt;configure terminal&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device# configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
| **Step 3** | `voice class sip-profiles tag`  
**Example:**  
Device(config)# voice class sip-profiles 10 | Enables dial peer-based VoIP SIP profile configurations and enters voice class configuration mode. |
| **Step 4** | `response response-code [method method-type] sip-header SIP-StatusLine modify match-pattern replacement-pattern`  
**Example:**  
Modifying status line of a SIP header to a user-defined response type:  
Device(config-class)# response 404 sip-header SIP-StatusLine modify "404 Not Found" "404 MyError" | Modifies SIP status line of a SIP response with user-defined values. |
| **Step 5** | `exit`  
**Example:**  
Device(config-class)# exit | Exits voice class configuration mode. |

**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 619
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 619
• Restrictions for Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls, on page 620
• Symmetric and Asymmetric Calls, on page 620
• High Availability Checkpointing Support for Asymmetric Payload, on page 621
• How to Configure Dynamic Payload Type Passthrough for DTMF and Codec Packets for SIP-to-SIP Calls, on page 622
• Configuration Examples for Assymmetric Payload Interworking, on page 625

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 69: 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:

\[ m = \text{audio} <\text{media-port1}> \text{ RTP/AVP XXX m = video} <\text{media-port2}> \text{ RTP/AVP XXX} \]

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 53: 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><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 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 voice 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> asymmetric payload {dtmf</td>
<td>dynamic-codecs</td>
</tr>
</tbody>
</table>
## Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(conf-serv-sip)# asymmetric payload full</code></td>
<td>Note: The <code>dtmf</code> and <code>dynamic-codecs</code> keywords are internally mapped to the <code>full</code> keyword to provide asymmetric payload type support for audio and video codecs, DTMF, and NSEs.</td>
</tr>
</tbody>
</table>

### DETAILED STEPS

### 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>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>enable</strong></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><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>configure terminal</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>dial-peer voice</strong> <em>tag</em> <em>voip</em></td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Device(config)# dial-peer voice 77 voip</code></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>voice-class sip asymmetric payload</strong> { <code>dtmf</code></td>
<td><code>dynamic-codecs</code></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`Device(config)# voice-class sip asymmetric payload {dtmf</td>
<td>dynamic-codecs</td>
</tr>
</tbody>
</table>
Purpose

Device(config-dial-peer)# voice-class sip asymmetric payload full

Step 5

end

Example:

Device(config-dial-peer)# end

(Optional) Exits dial peer voice configuration mode and enters privileged EXEC mode.

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>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 show call active voice compact</td>
<td>(Optional) Displays a compact version of call information.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# show call active voice compact</td>
<td></td>
</tr>
<tr>
<td>Step 3 show call active voice</td>
<td>(Optional) Displays call information for voice calls in progress.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# show call active voice</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 cc sip all
- debug voip ccapi inout
• debug voip rtp

Use the following debug commands to troubleshoot HA Checkpointing for Asymmetric Payload:

• debug voip ccapi all
• debug voice high-availability all
• debug voip rtp error
• debug voip rtp in/out
• debug voip rtp packet
• debug voip rtp high-availability
• debug voip rtp function
• debug ccsip all

Use the following show commands to troubleshoot HA Checkpointing for Asymmetric Payload:

• show redundancy state
• show redundancy inter-device
• show standby brief
• show voice high-availability summary
• show voip rtp stats
• show voip rtp high-availability stats
• show voip rtp connection detail
• show call active voice brief
• show call active voice [summary]
• show call active video brief
• show call active video [summary]
• show align
• show memory debug leak

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
!```

Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5
Example: Asymmetric Payload Interworking—Interworking Configuration

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 629
- H.323-to-SIP Interworking on CUBE, on page 639
- H.323-to-H.323 Interworking on CUBE, on page 645
- SIP RFC 2782 Compliance with DNS SRV Queries, on page 659
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 629
- Prerequisites for Delayed-Offer to Early-Offer, on page 630
- Restrictions for Delayed-Offer to Early-Offer Media Flow-Around, on page 630
- Delayed-Offer to Early-Offer in Media Flow-Around Calls, on page 630
- MidCall Renegotiation Support for Delayed-Offer to Early-Offer Calls, on page 634
- High-Density Transcoding Calls in Delayed-Offer to Early-Offer, on page 636

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 70: Feature Information for Delayed-Offer to Early-Offer

<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>
</tbody>
</table>
## Prerequisites for Delayed-Offer to Early-Offer

Configure delayed-offer to early-offer in media flow-around mode.

## Restrictions for Delayed-Offer to Early-Offer Media 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 54: 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. The 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>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><strong>Example:</strong></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><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Configure conversion of a delayed offer to an early offer:</td>
</tr>
<tr>
<td>Configure conversion of a delayed offer to an early offer:</td>
<td></td>
</tr>
<tr>
<td>- In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td>voice-class sip early-offer forced</td>
<td></td>
</tr>
<tr>
<td>- In global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>early-offer forced</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>In dial-peer configuration mode:</td>
</tr>
<tr>
<td>Device (config) dial-peer voice 10 voip</td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer) voice-class sip</td>
<td></td>
</tr>
<tr>
<td>early-offer forced</td>
<td></td>
</tr>
<tr>
<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 tag profile</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 number 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>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 5</strong> codec codec profile</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# profile 1 aacld</td>
<td></td>
</tr>
<tr>
<td><strong>Purpose:</strong> Audio codec profile is applied on the dial peer.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> video codec codec profile</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# video codec h264 profile 2</td>
<td></td>
</tr>
<tr>
<td><strong>Purpose:</strong> Video codec profile is applied on the dial peer.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> voice-class sip early-offer forced</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device (config-dial-peer)# voice-class sip early-offer forced</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> end</td>
<td></td>
</tr>
<tr>
<td><strong>Purpose:</strong> Exits to privileged EXEC mode.</td>
<td></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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Purpose:</strong> Enables privileged EXEC mode. Enter your password if prompted.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Purpose:</strong> Enters global configuration mode.</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Device# configure terminal</th>
</tr>
</thead>
</table>

### Step 3

**medial flow-around**

**Example:**

```
Device(config-voi-serv)# media flow-around
```

| Enables 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).
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.

The early-offer forced renegotiate command triggers a delayed-offer RE-INVITE if the negotiated codecs are one of the following:

- aacld—Audio codec AAACL 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

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>Example:</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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice <em>id</em> 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>Example:</td>
<td></td>
</tr>
<tr>
<td>Device (config) dial-peer voice 10 voip</td>
<td></td>
</tr>
<tr>
<td>Device (config-dial-peer) media transcoder high-density</td>
<td></td>
</tr>
<tr>
<td>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 56: 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 57: 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></td>
<td></td>
</tr>
<tr>
<td><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><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>
</tbody>
</table>
## Configuring High-Density Transcoding

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

| **Step 4** media transcoder high-density | Enables media transcoder high-density for transcoding high-density media calls. |
| **Example:** | |
| Device(config-voi-serv)# media transcoder high-density | |

| **Step 5** sip | Enters SIP configuration mode. |
| **Example:** | |
| Device(config-voi-serv)# sip | |

| **Step 6** early offer-forced | Forcefully sends SIP EO invites on the Out-Leg. |
| **Example:** | |
| Device(config-voi-sip)# early offer-forced | |

| **Step 7** end | Exits the present configuration mode. |
| **Example:** | |
| Device(config-voi-sip)# end | |
This chapter describes how to configure H.323-to-SIP interworking in CUBE and lists the various features supported in this interworking model.

**Note**

H.323 protocol is no longer supported from Cisco IOS XE Bengaluru 17.6.1a onwards. Consider using SIP for multimedia applications.

- Prerequisites, on page 639
- Restrictions, on page 639
- H.323-to-SIP Basic Call Interworking, on page 640
- H.323-to-SIP Supplementary Features Interworking, on page 642
- H.323-to-SIP Codec Progress Indicator Interworking for Media Cut-Through, on page 643
- Configuring H.323-to-SIP Interworking, on page 643

**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-Offerto 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.

H.323-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;––––––––––––– H.323 (TCP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SIP (TCP)&lt;––––––––––––– H.323 (TCP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SIP (UDP)&lt;––––––––––––– H.323 (UDP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SIP (TCP)&lt;––––––––––––– H.323 (UDP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default SIP protocol is UDP. Default H.323 protocol is TCP.</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>Interworking between • H.323 Fast-Start and SIP early-media signaling • H.323 Slow-Start and SIP delayed-media signaling</td>
<td>12.3(11)T</td>
<td>• H.323 Fast Start&lt;—&gt;SIP Early Media • H.323 Slow Start&lt;—&gt;SIP Delay Media <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>
<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): • H.323 to H.323 with only one leg having RSVP • H.323 to H.323 with both legs having RSVP • H.323 to SIP with only one leg having RSVP • H.323 to SIP with both legs having RSVP</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) • Codec transparent and codec filtering is not supported • Cisco Fax Relay and T.38 Fax are not supported</td>
</tr>
<tr>
<td>Feature</td>
<td>Supported Release</td>
<td>Additional Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------</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>
<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 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>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support H.323-to-SIP Supplementary services for CUCM with MTP on the</td>
<td>12.3(11)T</td>
</tr>
<tr>
<td>H.323 Trunk.</td>
<td></td>
</tr>
</tbody>
</table>
**Feature** | **Release**
--- | ---
ILBC Codec Support | 12.3(11)T
Interworking between G.711 inband DTMF to RFC2833 | 12.3(11)T
VXML 3.x support | 12.3(11)T
SIP CDRs and H.323 CDRs Mapping | 12.3(11)T
Conference ID can be used to correlate H.323 and SIP Radius records. Conference ID is unique on both H.323 and SIP legs | 12.3(11)T
VXML support with SIP Notify | 12.4(11)T
Mapping ECS to ReINVITE and ECS to REFER on the Cisco CUBE | 12.4(20)T

### H.323-to-SIP Codec Progress Indicator Interworking for Media Cut-Through

OGW is the originating gateway and TGW is the terminating gateway.

**Table 71: SIP(OGW)—>IPIPGW—>H.323(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 72: H.323(OGW)—>IPIPGW—>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 H.323-to-SIP Interworking

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. allow-connections h.323 to sip
5. allow-connections sip to h.323
### 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.  
Example:  
```
Router> enable
``` |
| 2    | configure terminal | Enters global configuration mode.  
Example:  
```
Router# configure terminal
``` |
| 3    | voice service voip | Enters Global VoIP configuration mode.  
Example:  
```
Router(config)# voice service voip
``` |
| 4    | allow-connections h.323 to sip | Allows connections from a h.323 endpoint to a SIP endpoint.  
Example: |
| 5    | allow-connections sip to h.323 | Allows connections from a SIP endpoint to a H.323 endpoint.  
Example: |
| 6    | end               | Exits to privileged EXEC mode.  
Example:  
```
Router(conf-voi-serv)# end
``` |
This chapter describes how to configure and enable features for H.323-to-H.323 connections on CUBE.

Note

H.323 protocol is no longer supported from 17.6.1 onwards. Consider using SIP for multimedia applications.

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 645
- Prerequisites, on page 646
- Restrictions, on page 646
- Slow Start to Fast-Start Interworking, on page 646
- Call Failure Recovery (Rotary), on page 648
- Managing H.323 IP Group Call Capacities, on page 649
- Overlap Signaling, on page 654
- Verifying H.323-to-H.323 Interworking, on page 655
- Troubleshooting H.323-to-H.323 Interworking, on page 657

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 73: Feature Information for H.323-to-H.323 Interworking

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>---------------------</td>
</tr>
<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>Overlap Signaling for H.323-to-H.323 Connections on a Cisco Unified Border Element</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>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-nce to out of band is not supported when high density transcoder is enabled. Use normal transcoding for rtp-nce 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>
</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**
| 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.
<p>| Example: | Enables interworking between slow start and fast start. |</p>
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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)# h323</td>
<td></td>
</tr>
<tr>
<td>Device(conf-serv-h323)# call start interwork</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>In voice class configuration mode</td>
<td></td>
</tr>
<tr>
<td>!Configuring a Voice class with Fast Start and Slow Start Interworking</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice class h323 10</td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# call start interwork</td>
<td></td>
</tr>
<tr>
<td>!Applying the voice class to a dial peer.</td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 20 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class h323 10</td>
<td></td>
</tr>
<tr>
<td>Step 4 end</td>
<td>Exits to privileged EXEC mode.</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**
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 1    | enable           | Enables privileged EXEC mode.  
|      | Example:         |         |
|      | Device> enable   | • Enter your password if prompted. |
| 2    | configure terminal | Enters global configuration mode. |
|      | Example:         |         |
|      | Device# configure terminal |         |
| 3    | voice service voip | Enters VoIP voice-service configuration mode. |
|      | Example:         |         |
|      | Device(config)# voice service voip |         |
| 4    | h323             | Enters H.323 voice-service configuration mode. |
|      | Example:         |         |
|      | Device(conf-voi-serv)# h323 |         |
| 5    | emptycapability | Enables call failure recovery (TCS=0) without the need for identical codec configuration. |
|      | Example:         |         |
|      | Device(conf-serv-h323)# emptycapability |         |
| 6    | exit             | Exits the current mode. |
|      | Example:         |         |
|      | Router(conf-serv-h323)# exit |         |

## 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>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Router&gt; enable</td>
<td></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 voice service voip</td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td>Example: Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td>Step 4 h323</td>
<td>Enters H.323 voice-service configuration mode.</td>
</tr>
<tr>
<td>Example: Router(conf-voi-serv)# h323</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

#### Step 5
```
ip circuit max-calls maximum-calls
```
**Example:**
```
Router(config-serv-h323)# ip circuit max-calls 1500
```

**Purpose**
(Required only if reserved calls are to exceed 1000) Sets the maximum number of aggregate H.323 IP circuit carrier call legs.

**Note**
If you do not configure this value, the default maximum value is 1000 reserved call legs. You may need to configure a lower value to obtain overload behavior. You can also configure a higher value.

**Note**
After you set a maximum number of call legs for defined circuits, any aggregate capacity left over is available for default circuits. For example, if you specify 1000 as the maximum number of call legs and then reserve 200 call legs for defined circuits, 800 call legs are available for use by default circuits.

**Note**
CUBE prevents you from allocating all of the capacity to specified carriers; at least one available circuit is required, which can be the default.

#### Step 6
```
ip circuit carrier-id carrier-name [reserved-calls reserved]
```
**Example:**
```
Router(config-serv-h323)# ip circuit carrier-id AA reserved-calls 500
```

**Purpose**
(Optional) Defines an IP circuit using the specified name as the circuit ID.

**Note**
The `reserved` keyword for this command is optional. Using this keyword creates a specified maximum number of calls for that circuit ID. The default value is 200 call legs.

#### Step 7
```
ip circuit default only
```
**Example:**
```
Router(config-serv-h323)# ip circuit default only
```

**Purpose**
(Optional) Creates a single carrier to use all of the call capacity available to CUBE.

**Note**
If you use the `ip circuit default only` command, you cannot use the `ip circuit carrier-id` command to configure more circuits. Using the `ip circuit default only` command creates a single carrier using the default carrier name.

#### Step 8
```
exit
```
**Example:**
```
Router(config-serv-h323)# exit
```

**Purpose**
Exits the current mode.

### 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

Example: Configuration with Default Calls in Excess of 1000

Example: Default Carrier and Incoming Source Carrier with No Voice Source Group

Note In this example, 800 call legs are implicitly reserved for the default circuit.

Note A gatekeeper is required with carrier-id routing.
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
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.
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. showcrm
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 showcrm
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 ccap inout`
CHAPTER 51

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.

• Prerequisites SIP RFC 2782 Compliance with DNS SRV Queries, on page 659
• Information SIP RFC 2782 Compliance with DNS SRV Queries, on page 659
• How to Configure SIP-RFC 2782 Compliance with DNS SRV Queries, on page 660
• Configuring DNS Server Lookups , on page 661
• Verifying, on page 663
• Feature Information for SIP RFC 2782 Compliance with DNS SRV Queries, on page 663

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_transp rort." 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.
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>
<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>Router&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

**Step 2**

**configure terminal**

*Example:*

Router# configure terminal

**Purpose:** Enters global configuration mode.

**Step 3**

**interface type number**

*Example:*

Router(config)# interface gigabitethernet 0/0/0

**Purpose:** Configures an interface type and enters interface configuration mode.

**Step 4**

**sip-ua**

*Example:*

Router(config-if)# sip-ua

**Purpose:** Enters SIP UA configuration mode.

**Step 5**

**srv version {1 | 2}**

*Example:*

Router(config-sip-ua)# srv version 2

**Purpose:** Generates DNS SRV queries in either RFC 2782 or RFC 2052 format.

- **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**

*Example:*

Router(config-sip-ua)# exit

**Purpose:** Exits the current configuration mode.

### 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

! dial-peer voice 1 voip
```
session protocol sipv2
session transport tcp tls
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
!

From Cisco IOS XE Gibraltar Release 16.12.3 onwards, CUBE sends '_sips._tcp.' query when the transport is TLS. The '_sips._tcp.' query is independent of the URI scheme—sip or sips. 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
!

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
!

Note
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
debug ip domain
debug ccsip info
debug ccsip 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.

**Feature History Table Entry**

<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>12.2(8)T, 12.2(11)T, 12.2(15)T</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>
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`.

<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 667
- SRTP-RTP Interworking, on page 683
- SRTP-SRTP Pass-Through, on page 697
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 667
- Prerequisites for SRTP-SRTP Interworking, on page 668
- Restrictions for SRTP-SRTP Interworking, on page 668
- Information About SRTP-SRTP Interworking, on page 668
- How to Configure SRTP-SRTP Interworking, on page 670
- Configuration Examples, on page 678

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 76: Feature Information for SRTP-SRTP Interworking

<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 show sip-ua calls is modified to display local crypto key and remote cryto key.</td>
</tr>
<tr>
<td>Support for SRTP-SRTP interworking</td>
<td>Cisco IOS XE Everest 16.5.1b</td>
<td>This feature allows secure calls between two enterprises using different cipher suites. Supported cipher suites are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AEAD_AES_256_GCM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AEAD_AES_128_GCM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AES_CM_128_HMAC_SHA1_80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AES_CM_128_HMAC_SHA1_32</td>
</tr>
</tbody>
</table>
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 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 that are supported for SRTP-SRTP interworking with default preference order are 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 1–4, 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.
Anycall-flow wherein there is a switchover from RTP to SRTP on the same SIP call-leg requires the supplementary-service media-renegotiate command that is enabled in global or voice service VoIP configuration mode to ensure that there is 2-way audio.

Example call-flows:

- RTP-RTP flow switching to SRTP-RTP.
- Nonsecure MOH being played during secure call hold or resume.
- RTP-SRTP flow switching to SRTP-SRTP.

When supplementary services are invoked from the endpoints, 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 675.

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>• 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>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</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><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice <strong>tag</strong> voip</td>
<td>Specifies either the prefix or the full E.164 telephone number to be used for a dial peer string.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# dial-peer voice 201 voip</td>
<td>• In the example, the following parameters are set:</td>
</tr>
<tr>
<td></td>
<td>• Dial peer 201 is defined.</td>
</tr>
<tr>
<td></td>
<td>• VoIP is shown as the method of encapsulation.</td>
</tr>
<tr>
<td><strong>Step 4</strong> destination-pattern <strong>string</strong></td>
<td>Specifies a session protocol for calls between local and remote routers using the packet network.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# destination-pattern 5550111</td>
<td>• In the example, 5550111 is specified as the pattern for the telephone number.</td>
</tr>
<tr>
<td><strong>Step 5</strong> session protocol sipv2</td>
<td>Specifies a session protocol for calls between local and remote routers using the packet network.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# session protocol sipv2</td>
<td>• In the example, the <strong>sipv2</strong> keyword is configured so that the dial peer uses the SIP protocol.</td>
</tr>
<tr>
<td><strong>Step 6</strong> session target ipv4:<strong>destination-address</strong></td>
<td>Designates an IP address where calls will be sent.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# session target ipv4:10.13.25.102</td>
<td>• In the example, calls matching this outbound dial-peer will be sent to 10.13.25.102.</td>
</tr>
<tr>
<td><strong>Step 7</strong> incoming called-number <strong>string</strong></td>
<td>Specifies a digit string that can be matched by an incoming call to associate the call with a dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# incoming called-number 5550111</td>
<td>• In the example, 5550111 is specified as the pattern for the E.164 or private dialing plan telephone number.</td>
</tr>
<tr>
<td><strong>Step 8</strong> srtp</td>
<td>Specifies that SRTP is used to enable secure calls for the dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# srtp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> codec <strong>codec</strong></td>
<td>Specifies the voice coder rate of speech for the dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-dial-peer)# codec</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device(config-dial-peer)# codec g711ulaw</td>
<td>• In the example, G.711 mu-law at 64,000 bps, is specified as the voice coder rate for speech.</td>
</tr>
</tbody>
</table>

**Step 10**

Example:

Device(config-dial-peer)#end

Exits dial peer voice configuration mode.

**Step 11**

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 12**

Repeat Steps 4, 5, 6, and 7 to configure a second dial peer.

**Step 13**

Example:

Device(config-dial-peer)# srtp

Specifies that SRTP is used to enable secure calls for the dial peer.

**Step 14**

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 15**

Example:

Device(config-dial-peer)# exit

Exits dial peer voice configuration mode.

---

**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
## 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

### 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 srtp-crypto tag</td>
<td>Enters voice class configuration mode and assign an identification tag for a srtp-crypto voice class.</td>
</tr>
<tr>
<td>Example: Device(config)# voice class srtp-crypto 100</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> crypto preference cipher-suite</td>
<td>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.</td>
</tr>
<tr>
<td>Example: Device(config-class)# crypto 1 AEAD_AES_256_GCM</td>
<td>You can configure a maximum of four preferences.</td>
</tr>
<tr>
<td><strong>Step 5</strong> exit</td>
<td>Exists the present configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config-class)# exit</td>
<td></td>
</tr>
</tbody>
</table>

### Example

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 673.
Applying Crypto Suite Selection Preference (optional)

- In global configuration mode:
  - `voice service voice`
  - `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`

### 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><strong>Step 3</strong> Apply crypto suite selection preference</td>
<td>Assigns previously configured crypto-suite selection preference.</td>
</tr>
<tr>
<td>• In global configuration mode:</td>
<td>The <code>crypto-tag</code> maps to the tag created using the <code>voice class srtp-crypto</code> command available in global configuration mode.</td>
</tr>
<tr>
<td>• <code>voice service voice</code></td>
<td></td>
</tr>
<tr>
<td>• <code>sip</code></td>
<td></td>
</tr>
<tr>
<td>• <code>srtp-crypto crypto-tag</code></td>
<td></td>
</tr>
</tbody>
</table>
**Command or Action** | **Purpose**
--- | ---
**Example:**
In global configuration mode:
```
Device> enable
Device# configure terminal
Device(config)# voice service voice
Device(config-voi-serv)# sip
Device(config-serv-sip)# srtp-crypto 102
```

In voice class tenant configuration mode:
```
Device> enable
Device# configure terminal
Device(config)# voice class tenant 100
Device(config-serv-sip)# srtp-crypto 102
```

In dial-peer configuration mode:
```
Device> enable
Device# configure terminal
Device(config)# dial-peer voice 300 voip
Device(config-dial-peer)# voice-class sip srtp-crypto 102
```

**Step 4**
**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:
   - 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>
</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**  
  Enter one of the following commands:  
  • 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 | Enables call fallback to nonsecure mode. |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fallback</strong> command to support SRTP fallback with Cisco Unified Communications Manager</td>
<td></td>
</tr>
<tr>
<td>• In global VoIP SIP configuration mode</td>
<td></td>
</tr>
<tr>
<td>voice service voip</td>
<td></td>
</tr>
<tr>
<td>sip</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>srtp negotiate cisco</td>
<td></td>
</tr>
<tr>
<td>(Enable this CLI along with srtp fallback command to support SRTP fallback with Cisco Unified Communications Manager)</td>
<td></td>
</tr>
</tbody>
</table>

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 : 2e9094e33b815992a519f82abf8e09d2
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: [10.25.15.63]: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 : bTQqZXbgFJddA1hE9wJGV3aKxo5vFV+z21234tVb2
Remote Crypto Key : bTQqZXbgFJddA1hE9wJGV3aKxo5vFV+z9876tVb2
Mid-Call Re-Association Count: 0
SRTP-RTP Re-Association 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# config 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
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 683
- Prerequisites for SRTP-RTP Interworking, on page 684
- Restrictions for SRTP-RTP Interworking, on page 684
- Information About SRTP-RTP Interworking, on page 684
- How to Configure Support for SRTP-RTP Interworking, on page 688
- Configuration Examples for SRTP-RTP Interworking, on page 695

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 77: Feature Information for SRTP-RTP Interworking

<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>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<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 on the Cisco Integrated Services Router Generation 2 platform.
- Dial peer hunting to H323 or TDM 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 62: 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 Manger (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.

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 that is enabled in global or voice service VoIP configuration mode to ensure that there is 2-way audio.

Example call-flows:

- RTP-RTP flow switching to SRTP-RTP.
- Nonsecure MOH being played during secure call hold or resume.
- RTP-SRTP flow switching to SRTP-SRTP.

When supplementary services are invoked from the endpoints, 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 675.
How to Configure Support for SRTP-RTP Interworking

Configuring SRTP-RTP Interworking Support

<table>
<thead>
<tr>
<th>Note</th>
<th>From Cisco IOS XE Everest Release 16.5.1b onwards, the following crypto suites are enabled by default on the SRTP leg:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• AEAD_AES_256_GCM</td>
</tr>
<tr>
<td></td>
<td>• AEAD_AES_128_GCM</td>
</tr>
<tr>
<td></td>
<td>• AES_CM_128_HMAC_SHA1_80</td>
</tr>
<tr>
<td></td>
<td>• AES_CM_128_HMAC_SHA1_32</td>
</tr>
</tbody>
</table>

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>
<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>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>• In the example, the following parameters are set:</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>• Dial peer 201 is defined.</td>
</tr>
<tr>
<td><code>Device(config)# dial-peer voice 201 voip</code></td>
<td>• VoIP is shown as the method of encapsulation.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Specifies either the prefix or the full E.164 telephone number to be used for a dial peer string.</td>
</tr>
<tr>
<td><code>destination-pattern string</code></td>
<td>• In the example, 5550111 is specified as the pattern for the telephone number.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# destination-pattern 5550111</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Specifies a session protocol for calls between local and remote routers using the packet network.</td>
</tr>
<tr>
<td><code>session protocol sipv2</code></td>
<td>• In the example, the <code>sipv2</code> keyword is configured so that the dial peer uses the SIP protocol.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# session protocol sipv2</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Designates an IPv4 destination address where calls will be sent.</td>
</tr>
<tr>
<td><code>session target ipv4: destination-address</code></td>
<td>• In the example, calls matching this outbound dial-peer will be sent to 10.13.25.102.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# session target ipv4:10.13.25.102</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Specifies a digit string that can be matched by an incoming call to associate the call with a dial peer.</td>
</tr>
<tr>
<td><code>incoming called-number string</code></td>
<td>• In the example, 5550111 is specified as the pattern for the E.164 or private dialing plan telephone number.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# incoming called-number 5550111</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Specifies the voice coder rate of speech for the dial peer.</td>
</tr>
<tr>
<td><code>codec codec</code></td>
<td>• In the example, G.711 mu-law at 64,000 bps, is specified as the voice coder rate for speech.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Purpose Command or Action

<table>
<thead>
<tr>
<th>Step 9</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config-dial-peer)# codec g711ulaw</td>
<td>Exits dial peer voice configuration mode.</td>
</tr>
</tbody>
</table>

### Purpose Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config) # dial-peer voice 200 voip</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><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 200 voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• In the example, the following parameters are set:</td>
</tr>
<tr>
<td></td>
<td>• Dial peer 200 is defined.</td>
</tr>
<tr>
<td></td>
<td>• VoIP is shown as the method of encapsulation.</td>
</tr>
</tbody>
</table>

### Purpose Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device(config-dial-peer)# exit</td>
<td>Exits dial peer voice configuration mode.</td>
</tr>
</tbody>
</table>

### 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>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>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></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>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></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<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>• In dial-peer configuration mode:</td>
<td>• The default value is sha1-32.</td>
</tr>
</tbody>
</table>
| ```
    dial-peer voice tag voip
    voice-class sip srtp-auth {sha1-32 | sha1-80 | system}
    ``` | |
| **Example:** | |
| Device(config)# dial-peer voice 15 voip | |
| Device(config-dial-peer)# voice-class sip srtp-auth sha1-80 | |
| **Example:** | |
| Device(config)# voice service voip | |
| Device(conf-voi-serv)# sip | |
| Device(conf-serv-sip)# srtp-auth sha1-80 | |
### 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
     ```markdown
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
     ```markdown
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>Step</th>
<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 |  
Example:  
Device(config)# dial-peer voice 10 voip  
Device(config-dial-peer)# srtpe fallback |
| Step 3 | Enter one of the following commands:  
- In dial-peer configuration mode  
  
  dial-peer  
  voice  
  tag  
  voip  
  srtpe  
  fallback (for interworking with devices other than Cisco Unified Communications Manager)  
  or  
  voice-class sip srtp  
  negotiate cisco (Enable this CLI along with srtpe fallback command to support SRTP fallback with Cisco Unified Communications Manager)  
  
- In global VoIP SIP configuration mode  
  voice service voip  
  sip  
  srtpe  
  fallback (for interworking with devices other than Cisco Unified Communications Manager)  
  or  
  srtpe  
  negotiate cisco (Enable this CLI along with srtpe fallback command to support SRTP fallback with Cisco Unified Communications Manager)  
  | Enables call fallback to nonsecure mode.  
Example:  
Device(config)# dial-peer voice 10 voip  
Device(config-dial-peer)# srtpe fallback |
## Troubleshooting Tips

The following commands help in troubleshooting SRTP-RTP supplementary services support:

- `debug ccsip all`
- `debug voip ccapi 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:**

```console
Device(conf-voi-serv)# exit
```

<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)# dial-peer voice 10 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip srtp negotiate Cisco</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></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(conf-voi-serv)# srtp fallback</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></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(conf-voi-serv)# srtpe negotiate cisco</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>exit</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# exit</td>
<td></td>
</tr>
<tr>
<td>Exits present configuration mode and enters privileged EXEC mode.</td>
<td></td>
</tr>
</tbody>
</table>
Device> enable

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
unicast call-legs: 0
Total call-legs: 4
0 12:45.256 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 12:49:45.256 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

Note

Effective Cisco IOS XE Everest Releases 16.5.1b, \texttt{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 \texttt{voice class srtp-crypto} command to configure the preferred cipher-suites for the SRTP call leg (connection). For more information, see SRTP-SRTP Interworking.

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:

\begin{verbatim}
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
\end{verbatim}

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:

\begin{verbatim}
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
\end{verbatim}
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 697
- Information About SRTP-SRTP Pass-Through, on page 698
- Configure Pass-Through of Unsupported Crypto Suites for a Specific Dial Peer, on page 699
- Configure Pass-Through of Unsupported Crypto Suites Globally, on page 701
- Configuration Examples for SRTP-SRTP Pass-Through, on page 702

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>
<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.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.11S</td>
<td>The following commands were introduced or modified: show sip-ua srtp, srtp-auth and voice-class sip srtp-auth.</td>
</tr>
</tbody>
</table>
### Feature Information

<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 `srtp` command was modified to add `pass-thru` keyword. |

### 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
• 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></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>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>Step 3</td>
<td><strong>dial-peer voice tag voip</strong></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 201 voip</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><strong>destination-pattern string</strong></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# destination-pattern 5550111</td>
</tr>
<tr>
<td>Step 5</td>
<td><strong>session protocol sipv2</strong></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td>Step 6</td>
<td><strong>session target ipv4: destination-address</strong></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>Step 7</td>
<td><strong>incoming called-number string</strong></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>Step 8</td>
<td><strong>srtp pass-thru</strong></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# srtp pass-thru</td>
</tr>
<tr>
<td>Step 9</td>
<td><strong>codec codec</strong></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# codec g711ulaw</td>
</tr>
<tr>
<td>Step 10</td>
<td><strong>end</strong></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# end</td>
</tr>
</tbody>
</table>
### Purpose

**Command or Action**  
**dial-peer voice** tag voip  
*Example:*  
Device(config)# dial-peer voice 200 voip

**Step 11**  
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 12**  
Repeat Steps 4, 5, 6, and 7 to configure a second dial peer. --

**Step 13**  
**srtp pass-thru**  
*Example:*  
Device(config-dial-peer)# srtp pass-thru

**Step 14**  
**codec codec**  
*Example:*  
Device(config-dial-peer)# codec g711ulaw

**Step 15**  
**exit**  
*Example:*  
Device(config-dial-peer)# exit

### Configure Pass-Through of Unsupported Crypto Suites Globally

**SUMMARY STEPS**

1. enable  
2. configure terminal  
3. voice service voip  
4. srtp pass-thru  
5. end

**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. |
<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>configure terminal</td>
<td>Example:</td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td>voice service voip</td>
<td>Example:</td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enables transparent passthrough of all crypto suites globally.</td>
</tr>
<tr>
<td>srtp pass-thru</td>
<td>Example:</td>
</tr>
<tr>
<td>Device(config-dial-peer)# srtp pass-thru</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Exits dial peer voice configuration mode.</td>
</tr>
<tr>
<td>end</td>
<td>Example:</td>
</tr>
<tr>
<td>Device(config-dial-peer)#end</td>
<td></td>
</tr>
</tbody>
</table>

### Configuration Examples for SRTP-SRTP Pass-Through

**Example for SRTP=SRTP Pass-Through**

```plaintext
enable
configure 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**

```plaintext
enable
configure terminal
```
dial-peer voice 201 voip
destination-pattern 5550111
session protocol sipv2
session target ipv4:10.13.25.102
incoming called-number 5550111
srtp pass-thru
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 pass-thru
codec g711ulaw
end

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 on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms, on page 707
- High Availability on Cisco ASR 1000 Series Aggregation Services Routers, on page 729
- High Availability on Cisco CSR 1000V or C8000V Cloud Services Routers, on page 759
- High Availability on Cisco Integrated Services Routers (ISR-G2), on page 771
- DSP High Availability Support, on page 811
- Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 815
- CVP Survivability TCL support with High Availability, on page 829
High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms

The High Availability (HA) feature allows you to benefit from the failover capability of Cisco Unified Border Element (CUBE) on two routers, one active and one standby. When the active router goes down for any reason, the standby router takes over seamlessly, preserving and processing your calls.

Figure 63: Cisco CUBE High Availability

- About CUBE High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms, on page 707
- How to Configure CUBE High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms, on page 712
- Verify Your Configuration, on page 718
- Troubleshoot High Availability Issues, on page 726

About CUBE High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms

CUBE supports Box-to-box redundancy on Cisco 4000 Series Integrated Services Router (Cisco 4000 Series ISR), Cisco Catalyst 8300, 8200, and 8200-L Series Edge Platforms, and uses Redundancy Group Infrastructure to provide High Availability.

Box-to-Box Redundancy

Box-to-box redundancy enables configuring a pair of routers to act as back up for each other. In the router pair, the active router is determined based on the failover conditions. The router pair continuously exchange status messages. CUBE session information is checkpointed across the active and standby router. This enables
the standby router to immediately take over all CUBE call processing responsibilities when the active router becomes unavailable.

**Redundancy Group (RG) Infrastructure**

A group of redundant interfaces form a Redundancy Group. The active and standby routers are connected by a configurable control link and data synchronization link. The control link is used to communicate the redundancy state for each router. The data synchronization link is used to transfer stateful information to synchronize the stateful database for the calls and media flows. Each pair of redundant interfaces is configured with the same unique ID number, also known as the Redundancy Interface Identifier (RII).

A Virtual IP address (VIP) is configured on interfaces that connect to the external network. All signaling and media is sourced from and sent to the Virtual IP address. External devices such as Cisco Unified Communication Manager, uses VIP as the destination IP address for the calls traversing through Cisco UBE.

The following figure shows the redundancy group configured for a pair of routers with a single outgoing interface.

*Figure 64: Redundancy Group Configuration*

---

**Network Topology**

This section describes how to configure the following network topology. PSTN access uses an Active and Standby pair of routers in a SIP trunk deployment between a Cisco Unified Communications Manager (Unified CM) and a service provider SIP trunk.
In this topology, both Active and Standby routers have the same configuration and connects through a physical switch across same interfaces. The topology is mandatory for the CUBE High Availability (HA) to work. For example, the CUBE-1 and CUBE-2 interface toward WAN must terminate on the same switch. Use Multiple interfaces or subinterfaces on either LAN or WAN side. Also, one CUBE has a lower IP address across all three interfaces on the same CUBE palletform.

We recommend that you keep the following in mind when configuring this topology:

- Connect the redundancy group control and data interfaces in the CUBE HA pair to the same physical switch to avoid any latency in the network.

- The RG control and data interfaces of the CUBE HA pair can be connected through a back-to-back cable or using a switch as shown in figures **Network Topology with switch between active and standby routers** and **Network Topology with crossover cable between active and standby routers**. However, it is recommended to use Portchannel for the RG control and data interfaces for redundancy. A single connection using back-to-back cable or switch presents a single point of failure due to a faulty cable, port, or switch, resulting in an error state where both routers are Active.

- If the RG ID is the same for the two different CUBE HA pairs, keepalive interface for check-pointing the RG control and data, and traffic must be in a different subnet or VLAN.

**Note**

This recommendation is applicable only if you connect using a switch, not by back-to-back cables.

- You can configure a maximum of two redundancy groups. Hence, there can be only two Active and Standby pairs within the same network.
This recommendation is applicable only if you connect using a switch, not by back-to-back cables.

- Source all signaling and media from and to the virtual IP address.
- Always save the running configuration to avoid losing it due to router reload during a failover.
- Virtual Routing and Forwarding
  - Define Virtual Router Forwarding (VRF) in the same order on both Active and Standby routers for an accurate synchronization of data.
  - You can configure VRFs only on the traffic interface (SIP and RTP). Do not configure VRF on redundancy group control and data interface.
  - VRF configurations on both the Active and Standby router must be identical. VRF IDs checkpoints for the calls before and after switchover (includes VRF-based RTP port range).
- Manually copy the configurations from one router to the other.
- Replicating the configuration on the Standby router does not commit to the startup configuration; it is the running configuration. You must run the `write memory` command to commit the changes that are synchronized from the Active router on the Standby router.

Considerations and Restrictions

The following is a list of further considerations and restrictions you should know before configuring this topology:

Considerations

- Only active calls are checkpointed (Calls that are connected with 200 OK or ACK transaction completed).
- When you apply and save the configuration for the first time, the platform must be reloaded.
- For H.323, and TCP-based calls, media preservation is supported after the failover, but session signaling is not preserved.
- If you have Cisco Unified Customer Voice Portal (CVP) in your network, we recommend that you configure TCP session transport for the SIP trunk between CVP and CUBE.
- Upon failover, the previously active CUBE reloads by design.
- CUBE uses the virtual IP address to communicate Smart Licensing information.
- For SIP-SIP TLS calls, configure both the active and standby CUBE as trust points to a common external CA Server.
- TCP sessions are not preserved during the failover. Remote user agents are expected to reestablish TCP sessions (using port 5060) before sending subsequent messages.
- Call Admission Control (CAC) state is maintained through switchover. After Stateful Switchover, no calls are allowed if the CAC limit is reached before the switchover.
• Up to six multimedia lines in the SDP are checkpointed for CUBE high availability. From Cisco IOS XE Release 3.17 onwards, SDP Passthru (up to two m-lines) calls are also checkpointed.

• Survivability.tcl preservation is supported from Cisco IOS XE Release 3.17 onwards for Unified Customer Voice Portal (CVP) deployments.

• SRTP-RTP, SRTP-SRTP, and SRTP Passthru are supported.

Note
Redundancy control traffic that is exchanged between CUBE-1 and CUBE-2 is not secured natively and displays SRTP encryption keys in cleartext. If SRTP is used, you must secure this traffic by configuring a transport IPsec tunnel between the two interfaces that are used as the redundancy control link.

• Port channel is supported for both RG control data and traffic interfaces only from Cisco IOS XE 16.3.1 onwards.

Figure 67: Additional Supported Options for CUBE HA

• LTI-based transcoder call flow preservation is supported from Cisco IOS XE Release 3.15 onwards and requires the same DSP module capacity on both active and standby in the same slot or subslot.

• While deploying High Availability pair with Application Centric Infrastructure (ACI), perform one of the following:
  • Disable IP data plane learning on the ACI VRF.
    Refer to IP Data-plane Learning for details.
  • Use an intermediate Layer 3 switch between the High Availability pair and the ACI deployment. This Layer 3 switch prevents the ACI from directly learning the CUBE IP address and its associated MAC addresses.

Restrictions

• 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 (Cisco Unified SRST) or TDM Gateway colocation on CUBE HA is not supported.

• Routers connected through Metropolitan Area Network (MAN) Ethernet regardless of latency are not supported.

• Out-of-band DTMF (Notify or KPML) is not supported post switchover. Only rtp-nte to rtp-nte and voice-inband to voice-inband DTMF works after the switchover.

• Media-flow around and UC Services API (Cisco Unified Communications Manager Network-Based Recording) are not supported.

• You cannot terminate Wide Area Network (WAN) on CUBE directly or Data HA on either side. Both active and standby routers must be in the same Data Center and connected to the same physical switch.

• The Courtesy Callback (CCB) feature is not supported if a callback was registered with Cisco Unified Customer Voice Portal (CVP) and then a switchover was done on CUBE.

• You cannot configure a secondary IP address for the interfaces.

• If the redundancy group ID is same for the two different CUBE HA pairs, then the keepalive interface that is used for checkpointing RG control and data traffic must be in a different subnet or VLAN.

• Call Progress Analysis (CPA) calls (before transferred to the agent), SCCP-based media resources, Noise Reduction, Acoustic Shock Protection (ASP), and transcoding calls are not supported.

• The failover time for a Box-to-box application is higher than the Inbox application.

• One CUBE must have lower IPs across all three interfaces on the same CUBE platform. For instance, CUBE-1 must have lower IP addresses in Gig0/0/0 interface compared with CUBE-2 Gig0/0/0 interface.

• CUBE box-to-box high availability requires same priority and threshold to be configured on both CUBE-1 and CUBE-2.

---

**How to Configure CUBE High Availability on Cisco 4000 Series ISR and Cisco Catalyst 8000 Series Edge Platforms**

**Before You Begin**

• Use Cisco IOS-XE Release 3.11 and or later on both active and standby routers.

• Ensure that you have the required licenses for configuring high availability. For detailed information, see Cisco Unified Border Element Data Sheet.

• Connect the active and the standby router through a layer 2 connection for the control path.

• Configure the Network Time Protocol (NTP) or set the clock to be identical on both active and standby routers, to allow timestamps and call timers to match.

• The latency times must be minimal on all control and data links to prevent timeouts.

• Physically redundant links, such as Gigabit EtherChannel, must be used for the control and data paths.
Configure High Availability

SUMMARY STEPS

1. Configure the Redundancy Group (RG).
2. Configure interface tracking.
3. Configure the interfaces.
4. Configure SIP Binding.
5. (Optional) If H.323 calls are involved, enable H.323 binding.
6. Configure the Punt Policing feature.
7. Configure the RG group under voice service voip. This enables Box-to-box CUBE HA.
8. Configure the Media Inactivity timer.
9. Reload the router.
10. Configure the peer router.
11. Point the attached devices to the CUBE Virtual IP (VIP) address.

DETAILED STEPS

Step 1 Configure the Redundancy Group (RG).

a) Enter application redundancy mode.

Example:

Router>enable
Router#configure terminal
Router(config)#redundancy
Router(config-r)#mode none
Router(config-red)#application redundancy
Router(config-red-app)#group 1

b) Configure a name for the redundancy group.

Example:

Router(config-red-app-grp)#name cube-ha

where cube-ha is the name of the redundancy group.

c) Specify the initial priority and failover threshold for a redundancy group.

Example:

Router(config-red-app-grp)#priority 100 failover threshold 75

where 100 is the priority value and 75 is the threshold value. Both routers should have the same priority and threshold values.

d) Configure the timers for delay and reload.

Example:

Router(config-red-app-grp)#timers delay 30 reload 60

Delay timer which is the amount of time to delay the RG group’s initialization and role negotiation after the interface comes up.

Default: 30 seconds. Range is 0-10000 seconds.
Reload timer is the amount of time to delay RG group initialization and role-negotiation after a reload.
Default: 60 seconds. Range is 0-10000 seconds.

e) Configure the interface used to exchange keepalive and hello messages between the router pair.

Example:
Router(config-red-app-grp)#control GigabitEthernet0/0/2 protocol 1
where GigabitEthernet0/0/2 is the interface and protocol 1 is the protocol instance that is attached to the interface.

f) Configure the interface that is used for checkpointing of data traffic.

Example:
Router(config-red-app-grp)#data GigabitEthernet0/0/2

g) Configure RG group tracking.

Example:
Router(config-red-app-grp)#track 1 shutdown
Router(config-red-app-grp)#track 2 shutdown

h) Specify the protocol instance that will be attached to a control interface and enters redundancy application protocol configuration mode.

Example:
Router(config-red-app-grp)#protocol 1

i) Configure the two timers for hello time and hold time.

Example:
Router(config-red-app-grp)#timers hellotime 3 holdtime 10
hellotime—Interval between successive hello messages.
Default is 3 seconds. Range is 250 milliseconds-254 seconds.
holdtime—The interval between the receipt of a hello message and the presumption that the sending router has failed. This duration has to be greater than the hellotime.
Default is 10 seconds. Range is 750 milliseconds-255 seconds.
We recommend that you configure the holdtime timer that is configured to be at least 3 times the value of the hellotime timer.

Step 2
Configure interface tracking.
The track command is used in RG to track the voice traffic interface state so that the active router initiates switchover after the traffic interface is down.
Configure the following commands at the global level to track the status of the interface.

Example:
Router(config)#track 1 interface GigabitEthernet0/0/0 line-protocol
Router(config)#track 2 interface GigabitEthernet0/0/1 line-protocol

Step 3
Configure the interfaces.
a) Configure the redundancy interface identifier for the redundancy group.
Required for generating a Virtual MAC (VMAC) address. You must use the same rii ID value on the interface of each router (active and standby) that has the same Virtual IP address.

If there is more than one Box-to-box HA pair on the same LAN, each pair MUST have unique rii IDs on their respective interfaces (to prevent collision). show redundancy application group all must indicate the correct local and peer information.

Example:

Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#ip address 203.0.113.10 255.255.0.0
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 1

Router(config)#interface GigabitEthernet0/0/1
Router(config-if)#ip address 192.0.2.1 255.255.255.0
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 2

b) Associate the interface with the redundancy group created.

Example:

Router(config-if)#redundancy group 1 ip 203.0.113.12 exclusive
Router(config-if)#redundancy group 1 ip 192.0.2.3 exclusive

c) Configure interface for RG control and data.

Example:

Router(config)#interface GigabitEthernet0/0/2
Router(config-if)#ip address 198.51.100.1 255.255.255.0
Router(config-if)#media-type rj45
Router(config-if)#negotiation auto

Step 4 Configure SIP Binding.

Configure CUBE to bind SIP messages to the interface that is configured with a Virtual IP address (VIP) for the RG group employed.

Example:

Router(config)#dial-peer voice 1 voip
Router(config-dial-peer)#session protocol sipv2
Router(config-dial-peer)#incoming called-number 2000
Router(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/0/0
Router(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/0/0
Router(config-dial-peer)#codec g711ulaw
Router(config-dial-peer)#!

Router(config)#dial-peer voice 2 voip
Router(config-dial-peer)#destination-pattern 2000
Router(config-dial-peer)#session protocol sipv2
Router(config-dial-peer)#session target ipv4:203.0.113.13
Router(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/0/1
Router(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/0/1
Router(config-dial-peer)#codec g711ulaw

Step 5 (Optional) If H.323 calls are involved, enable H.323 binding.
Under the interface used by H.323, configure voip-bind with its source address equal to the interface’s VIP for the RG group employed.

**Example:**

```bash
Router#voice service voip
Router(config-voi-serv)#h323
Router(config-serv-h323)#call preserve limit-media-detection
Router(config-serv-h323)#no h225 timeout keepalive

Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#ip address 203.0.113.10 255.255.0.0
Router(config-if)#media-type rj45
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 1
Router(config-if)#redundancy group 1 ip 9.13.25.123 exclusive
Router(config-if)#h323-gateway voip interface
Router(config-if)#h323-gateway voip bind srcaddr 203.0.113.12

Router(config)#interface GigabitEthernet0/0/1
Router(config-if)#ip address 192.0.2.1 255.255.255.0
Router(config-if)#media-type rj45
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 2
Router(config-if)#redundancy group 1 ip 192.0.2.3 exclusive
Router(config-if)#h323-gateway voip interface
Router(config-if)#h323-gateway voip bind srcaddr 192.0.2.3
```

**Step 6**

Configure the Punt Policing feature.

SIP packets towards the virtual IP address and physical IP address match different punt-cause codes. The punt-rate of the virtual IP address with a punt-cause of 60, is lower than the punt-rate of the physical IP address.

To ensure that the behaviour of the SIP packets towards virtual and physical IP address remains the same, you must increase the punt-rate of the virtual IP address by using the `platform punt-policer` command in global configuration mode.

**Note** For Cisco IOS XE Releases 16.6.7, 16.9.4, 16.11.1, 16.12.1, 17.1.1 and later releases, you do not need to increase the punt-rate.

**Example:**

```bash
Router(config)#platform punt-policer 60 40000
```

In the preceding example, the punt-rate of the virtual IP address (punt-cause 60) is increased from the default value of 2000–40000.

The following table provides details of the fields of the CLI.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>platform punt-policer</code></td>
<td>Configures the Punt Policing feature.</td>
</tr>
<tr>
<td>60</td>
<td>punt-cause—Punt cause. Range is 1–107. Punt cause of the virtual interface is 60.</td>
</tr>
<tr>
<td>40000</td>
<td>punt-rate—Rate limit in packets per second. Range is 10–146484.</td>
</tr>
</tbody>
</table>
**Note**  The default punt rate value of the virtual IP address and the physical IP address varies with the router platform.

**Note**  The default and maximum setting are platform-specific. Default value is optimal for most deployments. Change the rate only when suggested by Cisco Support.

**Step 7**  Configure the RG group under **voice service voip**. This enables Box-to-box CUBE HA.

**Example:**

```
Router#voice service voip
Router(conf-voi-serv)#redundancy-group 1
```

**Step 8**  Configure the Media Inactivity timer.

The Media Inactivity Timer enables the active and standby router pair to monitor and disconnect calls if no Real-Time Protocol (RTP) packets are received within a configurable time period.

For the SIP calls, the switched over calls are cleared with signaling (as signaling information is preserved for switched calls).

The Media Inactivity Timer releases TCP-based and H.323-based calls. This is used to guard against any hung sessions resulting from the failover when a normal call disconnect does not clear the call.

You must configure the same duration for the Media Inactivity Timer on both routers. The default value is 30 seconds for SIP and H.323 calls. The sample configuration is as follows:

**Example:**

```
Router(config)#ip rtcp report interval 9000
Router(config)#gateway
Router(config-gateway)#media-inactivity-criteria all
Router(config-gateway)#timer receive-rtp 1200
Router(config-gateway)#timer receive-rtcp 5
```

SIP and H.323 call legs are cleared once the RTCP timer expires.

**Step 9**  Reload the router.

Once all the preceding configurations are completed, you must save the configurations, and reload the router.

**Example:**

```
Router>enable
Router#reload
```

**Step 10**  Configure the peer router.

Follow the preceding steps to configure the standby router. Make sure that you use the correct IP addresses.

**Step 11**  Point the attached devices to the CUBE Virtual IP (VIP) address.

The IP-PBX, Unified SIP Proxy, or service provider must route the calls to CUBE’s Virtual IP address.

HA configuration does not handle SIP and H.323 messages to the CUBE’s physical IP addresses.

For H.323 calls, you must disable the keepalive messages in Unified CM configuration.

- **a.** Go to **System** menu, and choose **Service Parameters**. At the bottom of the Service Parameters, enable **Advanced**.
- **b.** Set the **Allow TCP KeepAlives for H323** to False.
c. After this setting is saved, restart the CallManager Services.

Configuration Examples

Example: Control Interface Protocol Configuration

Router#configure terminal
Router(config)#redundancy
Router(config-red)#mode none
Router(config-red)#application redundancy
Router(config-red-app)#protocol 4
Router(config-red-app-prot)#name rg1
Router(config-red-app-prot)#timers hellotime 3 holdtime 10
Router(config-red-app-prot)#authentication text password

Example: Redundancy Group Protocol Configuration

Router#configure terminal
Router(config)#redundancy
Router(red)#application redundancy
Router(config-red-app)#protocol 1
Router(config-red-app-prtcl)#name RG1
Router(config-red-app-prtcl)#timers hellotime 1 holdtime 3
Router(config-red-app-prtcl)#end
Router#configure terminal
Router(config)#redundancy
Router(red)#application redundancy
Router(config-red-app)#protocol 2
Router(config-red-app-prtcl)#name RG1
Router(config-red-app-prtcl)#end

Example: Redundant Traffic Interface Configuration

Router#configure terminal
Router(config)#interface GigabitEthernet 0/0/2
Router(config-if)#ip address 198.51.100.1 255.0.0.0
Router(config-if)#ip nat outside
Router(config-if)#ip virtual-reassembly
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 200
Router(config-if)#redundancy group 1 ip 198.51.100.50 exclusive decrement 10

Verify Your Configuration

All configuration commands in this task are optional. You can use the show commands in any order.

SUMMARY STEPS

1. show redundancy application group [group-id | all]
2. show redundancy application transport {clients | group [group-id]}
3. `show redundancy application protocol {protocol-id | group [group-id]}`
4. `show redundancy application faults group [group-id]`
5. `show redundancy application if-mgr group [group-id]`
6. `show redundancy application control-interface group [group-id]`
7. `show redundancy application data-interface group [group-id]`

**DETAILED STEPS**

---

**Step 1**

`show redundancy application group [group-id | all]`

**Example:**

```
Router#show redundancy application group
Group ID       Group Name                State
-------------- -------------------- -------
1              Generic-Redundancy-1     STANDBY
2              Generic-Redundancy2     ACTIVE
```

The following example shows the details of redundancy application group 1:

```
Router#show redundancy application group 1
Group ID:1
Group Name:Generic-Redundancy-1

Administrative State: No Shutdown
Aggregate operational state : Up
My Role: STANDBY
Peer Role: ACTIVE
Peer Presence: Yes
Peer Comm: Yes
Peer Progression Started: Yes

RF Domain: btob-one
RF state: STANDBY HOT
Peer RF state: ACTIVE
```

The following example shows the details of redundancy application group 2:

```
Router#show redundancy application group 2
Group ID:2
Group Name:Generic-Redundancy2

Administrative State: No Shutdown
Aggregate operational state : Up
My Role: ACTIVE
Peer Role: STANDBY
Peer Presence: Yes
Peer Comm: Yes
Peer Progression Started: Yes

RF Domain: btob-two
RF state: ACTIVE
Peer RF state: STANDBY HOT
```

**Step 2**

`show redundancy application transport {clients | group [group-id]}`

**Example:**
The following example shows configuration details for the redundancy application transport group:

Router#show redundancy application transport client

<table>
<thead>
<tr>
<th>Client</th>
<th>Conn#</th>
<th>Priority</th>
<th>Interface</th>
<th>L3</th>
<th>L4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)RF</td>
<td>0</td>
<td>1</td>
<td>CTRL</td>
<td>IPV4</td>
<td>SCTP</td>
</tr>
<tr>
<td>(1)MCP_HA</td>
<td>1</td>
<td>1</td>
<td>DATA</td>
<td>IPV4</td>
<td>UDP_REL</td>
</tr>
<tr>
<td>(4)AR</td>
<td>0</td>
<td>1</td>
<td>ASYM</td>
<td>IPV4</td>
<td>UDP</td>
</tr>
<tr>
<td>(5)CF</td>
<td>0</td>
<td>1</td>
<td>DATA</td>
<td>IPV4</td>
<td>SCTP</td>
</tr>
</tbody>
</table>

The following examples show the configuration details of redundancy application transport group 1:

Router#show redundancy application transport group

Transport Information for RG (1)
Client = RF
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
0 0 1.1.1.1 59000 1.1.1.2 59000 CTRL IPV4 SCTP
Client = MCP_HA
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
1 1 9.9.9.2 53000 9.9.9.1 53000 DATA IPV4 UDP_REL
Client = AR
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
2 0 0.0.0.0 0 0.0.0.0 0 NONE_IN NONE_L3 NONE_L4
Client = CF
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
3 0 9.9.9.2 59001 9.9.9.1 59001 DATA IPV4 SCTP

Transport Information for RG (2)
Client = RF
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
8 0 1.1.1.1 59004 1.1.1.2 59004 CTRL IPV4 SCTP
Client = MCP_HA
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
9 1 9.9.9.2 53002 9.9.9.1 53002 DATA IPV4 UDP_REL
Client = AR
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
10 0 0.0.0.0 0 0.0.0.0 0 NONE_IN NONE_L3 NONE_L4
Client = CF
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
11 0 9.9.9.2 59005 9.9.9.1 59005 DATA IPV4 SCTP

The following example shows the configuration details of redundancy application transport group 2:

Router#show redundancy application transport group 1

Transport Information for RG (1)
Client = RF
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
0 0 1.1.1.1 59000 1.1.1.2 59000 CTRL IPV4 SCTP
Client = MCP_HA
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
1 1 9.9.9.2 53000 9.9.9.1 53000 DATA IPV4 UDP_REL
Client = AR
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
2 0 0.0.0.0 0 0.0.0.0 0 NONE_IN NONE_L3 NONE_L4
Client = CF
TI conn_id my_ip my_port peer_ip peer_port intf L3 L4
3 0 9.9.9.2 59001 9.9.9.1 59001 DATA IPV4 SCTP
Router# show redundancy application transport group 2
Transport Information for RG (2)
Client = RF
   TI conn_id my_ip   my_port peer_ip peer_port  intf  L3   L4
   8  0 1.1.1.1     59004 1.1.1.2   59004 CTRL IPV4 SCTP
Client = MCP_HA
   TI conn_id my_ip   my_port peer_ip peer_port  intf  L3   L4
   9  1 9.9.9.2     53002 9.9.9.1   53002 DATA IPV4 UDP_REL
Client = AR
   TI conn_id my_ip   my_port peer_ip peer_port  intf  L3   L4
  10  0 0.0.0.0     0 0.0.0.0     0  NONE_IN NONE_L3 NONE_L4
Client = CF
   TI conn_id my_ip   my_port peer_ip peer_port  intf  L3   L4
  11  0 9.9.9.2     59005 9.9.9.1   59005 DATA IPV4 SCTP

Step 3  show redundancy application protocol  \{protocol-id | group [group-id]\}

Example:
Router# show redundancy application protocol group

RG Protocol RG 1
-----------------
Role: Standby
Negotiation: Enabled
Priority: 50
Protocol state: Standby-hot
Ctrl Intf(s) state: Up
Active Peer: address 1.1.1.2, priority 150, intf Gi0/0/0
Standby Peer: Local
Log counters:
  role change to active: 0
  role change to standby: 1
  disable events: rg down state 1, rg shut 0
  ctrl intf events: up 2, down 1, admin_down 1
  reload events: local request 0, peer request 0

RG Media Context for RG 1
-------------------------
Ctx State: Standby
Protocol ID: 1
Media type: Default
Control Interface: GigabitEthernet0/0/0
Current Hello timer: 3000
Configured Hello timer: 3000, Hold timer: 10000
Peer Hello timer: 3000, Peer Hold timer: 10000
Stats:
  Pkts 117, Bytes 7254, HA Seq 0, Seq Number 117, Pkt Loss 0
  Authentication not configured
  Authentication Failure: 0
  Reload Peer: TX 0, RX 0
  Resign: TX 0, RX 0
  Active Peer: Present. Hold Timer: 10000
  Pkts 115, Bytes 3910, HA Seq 0, Seq Number 1453975, Pkt Loss 0

RG Protocol RG 2
-----------------
Role: Active
Negotiation: Enabled
Priority: 135
Protocol state: Active
Ctrl Intf(s) state: Up
Active Peer: Local
Standby Peer: address 1.1.1.2, priority 130, intf Gi0/0/0
Log counters:
role change to active: 1
role change to standby: 1
disable events: rg down state 1, rg shut 0
ctrl intf events: up 2, down 1, admin_down 1
reload events: local request 0, peer request 0

RG Media Context for RG 2
--------------------------
Ctx State: Active
Protocol ID: 2
Media type: Default
Control Interface: GigabitEthernet0/0/0
Current Hello timer: 3000
Configured Hello timer: 3000, Hold timer: 10000
Peer Hello timer: 3000, Peer Hold timer: 10000
Stats:
Pkts 118, Bytes 7316, HA Seq 0, Seq Number 118, Pkt Loss 0
Authentication not configured
Authentication Failure: 0
Reload Peer: TX 0, RX 0
Resign: TX 0, RX 1
Standby Peer: Present. Hold Timer: 10000
Pkts 102, Bytes 3468, HA Seq 0, Seq Number 1453977, Pkt Loss 0

The following example shows configuration details for the redundancy application protocol group 1:

Router#show redundancy application protocol group 1

RG Protocol RG 1
------------------
Role: Standby
Negotiation: Enabled
Priority: 50
Protocol state: Standby-hot
Ctrl Intf(s) state: Up
Active Peer: address 1.1.1.2, priority 150, intf Gi0/0/0
Standby Peer: Local
Log counters:
role change to active: 0
role change to standby: 1
disable events: rg down state 1, rg shut 0
ctrl intf events: up 2, down 1, admin_down 1
reload events: local request 0, peer request 0

RG Media Context for RG 1
--------------------------
Ctx State: Standby
Protocol ID: 1
Media type: Default
Control Interface: GigabitEthernet0/0/0
Current Hello timer: 3000
Configured Hello timer: 3000, Hold timer: 10000
Peer Hello timer: 3000, Peer Hold timer: 10000
Stats:
Pkts 120, Bytes 7440, HA Seq 0, Seq Number 120, Pkt Loss 0
Authentication not configured
Authentication Failure: 0
Reload Peer: TX 0, RX 0
Resign: TX 0, RX 0
Active Peer: Present. Hold Timer: 10000
The following example shows configuration details for the redundancy application protocol group 2:

```
Router# show redundancy application protocol group 2

RG Protocol RG 2
------------------
Role: Active
Negotiation: Enabled
Priority: 135
Protocol state: Active
Ctrl Intf(s) state: Up
Active Peer: Local
Standby Peer: address 1.1.1.2, priority 130, intf Gi0/0/0
Log counters:
role change to active: 1
role change to standby: 1
disable events: rg down state 1, rg shut 0
ctrl intf events: up 2, down 1, admin_down 1
reload events: local request 0, peer request 0

RG Media Context for RG 2
--------------------------
Ctx State: Active
Protocol ID: 2
Media type: Default
Control Interface: GigabitEthernet0/0/0
Current Hello timer: 3000
Configured Hello timer: 3000, Hold timer: 10000
Peer Hello timer: 3000, Peer Hold timer: 10000
Stats:
Pkts 123, Bytes 7626, HA Seq 0, Seq Number 123, Pkt Loss 0
Authentication not configured
Authentication Failure: 0
Reload Peer: TX 0, RX 0
Resign: TX 0, RX 1
Standby Peer: Present. Hold Timer: 10000
Pkts 107, Bytes 3638, HA Seq 0, Seq Number 1453982, Pkt Loss 0
```

The following example shows configuration details for the redundancy application protocol 1:

```
Router# show redundancy application protocol 1

Protocol id: 1, name: rg-protocol-1
Hello timer in msecs: 3000
Hold timer in msecs: 10000
```

Step 4  show redundancy application faults group [group-id]

Example:
```
Router# show redundancy application faults group

Faults states Group 1 info:
Runtime priority: [50]
RG Faults RG State: Up.
Total # of switchovers due to faults: 0
Total # of down/up state changes due to faults: 2
```
Faults states Group 2 info:
Runtime priority: [135]
RG Faults RG State: Up.
Total # of switchovers due to faults: 0
Total # of down/up state changes due to faults: 2

The following example shows configuration details specific to redundancy application faults group 1:

```
Router# show redundancy application faults group 1
```

Faults states Group 1 info:
Runtime priority: [50]
RG Faults RG State: Up.
Total # of switchovers due to faults: 0
Total # of down/up state changes due to faults: 2

The following example shows configuration details specific to redundancy application faults group 2:

```
Router# show redundancy application faults group 2
```

Faults states Group 2 info:
Runtime priority: [135]
RG Faults RG State: Up.
Total # of switchovers due to faults: 0
Total # of down/up state changes due to faults: 2

**Step 5**  
```
show redundancy application if-mgr group [group-id]
```

**Example:**

```
Router# show redundancy application if-mgr group
```

**RG ID: 1**

```
interface GigabitEthernet0/0/3.152
---------------------------------------
VMAC 0007.b421.4e21
VIP 55.1.1.255
Shut shut
Decrement 10
interface GigabitEthernet0/0/2.152
---------------------------------------
VMAC 0007.b421.5209
VIP 45.1.1.255
Shut shut
Decrement 10
```

**RG ID: 2**

```
interface GigabitEthernet0/0/3.166
---------------------------------------
VMAC 0007.b422.14d6
VIP 4.1.255.254
Shut no shut
Decrement 10
interface GigabitEthernet0/0/2.166
---------------------------------------
VMAC 0007.b422.0d06
VIP 3.1.255.254
```
The following examples shows configuration details for redundancy application interface manager group 1 and group 2:

Router# show redundancy application if-mgr group 1

RG ID: 1
---------
interface GigabitEthernet0/0/3.152
-------------------------------
VMAC 0007.e421.4e21
VIP 55.1.1.255
Shut shut
Decrement 10
interface GigabitEthernet0/0/2.152
-------------------------------
VMAC 0007.e421.5209
VIP 45.1.1.255
Shut shut
Decrement 10

Router# show redundancy application if-mgr group 2

RG ID: 2
---------
interface GigabitEthernet0/0/3.166
-------------------------------
VMAC 0007.e422.14d6
VIP 4.1.255.254
Shut no shut
Decrement 10
interface GigabitEthernet0/0/2.166
-------------------------------
VMAC 0007.e422.0d06
VIP 3.1.255.254
Shut no shut
Decrement 10

Step 6  show redundancy application control-interface  group [group-id]

Example:
Router# show redundancy application control-interface group

The control interface for rg[1] is GigabitEthernet0/0/0
Interface is Control interface associated with the following protocols: 2 1
Interface Neighbors:
Peer: 1.1.1.2 Active RGs: 1 Standby RGs: 2

The control interface for rg[2] is GigabitEthernet0/0/0
Interface is Control interface associated with the following protocols: 2 1
Interface Neighbors:
Peer: 1.1.1.2 Active RGs: 1 Standby RGs: 2

The following example shows configuration details of the redundancy application control-interface group 1:
Router# show redundancy application control-interface group 1
The control interface for rg[1] is GigabitEthernet0/0/0
Interface is Control interface associated with the following protocols: 2 1
Interface Neighbors:
Peer: 1.1.1.2 Active RGs: 1 Standby RGs: 2

The following example shows configuration details of the redundancy application control-interface group 2:

Router# show redundancy application control-interface group 2

The control interface for rg[2] is GigabitEthernet0/0/0
Interface is Control interface associated with the following protocols: 2 1
Interface Neighbors:
Peer: 1.1.1.2 Active RGs: 1 Standby RGs: 2

Step 7

show redundancy application data-interface group [group-id]

Example:

Router# show redundancy application data-interface group

The data interface for rg[1] is GigabitEthernet0/0/1
The data interface for rg[2] is GigabitEthernet0/0/1

The following examples show configuration details specific to redundancy application data-interface group 1 and group 2:

Router# show redundancy application data-interface group 1
The data interface for rg[1] is GigabitEthernet0/0/1

Router# show redundancy application data-interface group 2
The data interface for rg[2] is GigabitEthernet0/0/1

---

Troubleshoot High Availability Issues

Use the following show and debug commands to troubleshoot any issues:

- show redundancy application group all
- show redundancy application transport clients
- show redundancy client domain all | inc VOIP RG
- show voice high-availability summary
- show voip fpi stats
- debug voip rtp session
- debug voice high-availability all
- debug voip fpi all
- debug redundancy application group {config | faults | media | protocol | rii transport | vp}
Note

Do not turn on a large number of debugs on a system carrying high volume of active call traffic.
High Availability on Cisco ASR 1000 Series Aggregation Services Routers

The High Availability (HA) feature allows you to benefit from the failover capability of Cisco Unified Border Element (CUBE) on two routers, one active and one standby. When the active router goes down for any reason, the standby router takes over seamlessly, preserving and processing your calls.

Figure 68: Cisco CUBE High Availability

• About CUBE High Availability on Cisco ASR 1000 Series Routers, on page 729
• How to Configure CUBE High Availability on Cisco ASR 1000 Series Router, on page 736
• Verify Your Configuration, on page 749
• Troubleshoot High Availability Issues, on page 756

About CUBE High Availability on Cisco ASR 1000 Series Routers

CUBE supports two HA options on the Cisco ASR 1000 Series Aggregation Services Router:

• Box-to-box Redundancy
• Inbox Redundancy

The following table describes the Cisco ASR 1000 Series Router models supported for each redundancy type:
### Table 79: Redundancy Type, Supported Models, and Supported Cisco IOS XE Release

<table>
<thead>
<tr>
<th>Redundancy Type</th>
<th>Router Models</th>
<th>Supported Cisco IOS-XE Release</th>
</tr>
</thead>
</table>
| Box-to-box       | • Cisco ASR 1001-X Router  
                  | • Cisco ASR 1002-X Router   
                  | • Cisco ASR 1004 Router     
                  | • Cisco ASR 1006 Router (with a single RP and an ESP)  
                  | • Cisco ASR 1006-X Router (with a single RP and an ESP) | Cisco IOS XE Release 3.11 onwards |
| Inbox            | Cisco ASR 1006 Router | Cisco IOS XE Release 3.11 onwards |

**Note** Cisco ASR 1006 supports both Box-to-box and Inbox redundancy. You cannot switch between these two modes dynamically.

The following table provides details on the type of information that is preserved in different call types:

### Table 80: Call Preservation for Various Call Types

<table>
<thead>
<tr>
<th>Call Type</th>
<th>Transport Layer</th>
<th>Call Preservation After Switchover</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP-SIP</td>
<td>UDP</td>
<td>Both media and session signaling are preserved.</td>
</tr>
<tr>
<td>SIP-SIP</td>
<td>TCP/TLS</td>
<td>Both media and session signaling are preserved using port 5060.</td>
</tr>
<tr>
<td>SIP-H.323</td>
<td>TCP or UDP</td>
<td>Only media is preserved. Session signaling is not preserved.</td>
</tr>
<tr>
<td>H.323-H.323</td>
<td>TCP</td>
<td></td>
</tr>
</tbody>
</table>

### Inbox Redundancy

Inbox redundancy with Stateful Switchover (SSO) mechanism provides redundancy within the same device. Cisco ASR1006 supports the stateful failover from an active Enhanced Services Processor (ESP) to a standby and from an active Route Processor to a standby on the same box.
Box-to-Box Redundancy

Box-to-box redundancy enables configuring a pair of routers to act as back up for each other. In the router pair, the active router is determined based on the failover conditions. The router pair continuously exchange status messages. CUBE session information is checkpointed across the active and standby router. This enables the standby router to immediately take over all CUBE call processing responsibilities when the active router becomes unavailable.

Redundancy Group (RG) Infrastructure

A group of redundant interfaces form a Redundancy Group. The active and standby routers are connected by a configurable control link and data synchronization link. The control link is used to communicate the redundancy state for each router. The data synchronization link is used to transfer stateful information to synchronize the stateful database for the calls and media flows. Each pair of redundant interfaces is configured with the same unique ID number, also known as the Redundancy Interface Identifier (RII).

A Virtual IP address (VIP) is configured on interfaces that connect to the external network. All signaling and media is sourced from and sent to the Virtual IP address. External devices such as Cisco Unified Communication Manager, uses VIP as the destination IP address for the calls traversing through Cisco UBE.

The following figure shows the redundancy group configured for a pair of routers with a single outgoing interface.
PROTECTED Mode

The default failover redundancy behavior in a box-to-box HA pair is to reload the affected router to avoid out-of-sync conditions or Split brain. From release IOS XE 3.11 onwards, you can configure a Cisco ASR 1000 Series Router to transition into PROTECTED mode, which has the following features:

- Bulk sync request, Call checkpointing, and incoming call processing are disabled.
- You must manually reload a router in PROTECTED mode to come out of this state.

To enable the PROTECTED mode, use the `no redundancy-reload` command under `voice service voip`.

Network Topology

This section describes how to configure the following network topology. PSTN access uses an Active and Standby pair of routers in a SIP trunk deployment between a Cisco Unified Communications Manager (Unified CM) and a service provider SIP trunk.
In this topology, both Active and Standby routers have the same configuration and connects through a physical switch across same interfaces. The topology is mandatory for the CUBE High Availability (HA) to work. For example, the CUBE-1 and CUBE-2 interface toward WAN must terminate on the same switch. Use Multiple interfaces or subinterfaces on either LAN or WAN side. Also, one CUBE has a lower IP address across all three interfaces on the same CUBE platform.

We recommend that you keep the following in mind when configuring this topology:

- Connect the redundancy group control and data interfaces in the CUBE HA pair to the same physical switch to avoid any latency in the network.

- The RG control and data interfaces of the CUBE HA pair can be connected through a back-to-back cable or using a switch as shown in figures Network Topology with switch between active and standby routers and Network Topology with crossover cable between active and standby routers. However, it is recommended to use Portchannel for the RG control and data interfaces for redundancy. A single connection using back-to-back cable or switch presents a single point of failure due to a faulty cable, port, or switch, resulting in error state where both routers are Active.

- If the RG ID is the same for the two different CUBE HA pairs, keepalive interface for check-pointing the RG control and data, and traffic must be in a different subnet or VLAN.

Note  
This recommendation is applicable only if you connect using a switch, not by back-to-back cables.

- You can configure a maximum of two redundancy groups. Hence, there can be only two Active and Standby pairs within the same network.
Considerations and Restrictions

The following is a list of further considerations and restrictions you should know before configuring this topology:

Considerations

- Only active calls are checkpointed (Calls that are connected with 200 OK or ACK transaction completed).
- When you apply and save the configuration for the first time, the platform must be reloaded.
- For H.323, and TCP-based calls, media preservation is supported after the failover, but session signaling is not preserved.
- If you have Cisco Unified Customer Voice Portal (CVP) in your network, we recommend that you configure TCP session transport for the SIP trunk between CVP and CUBE.
- Upon failover, the previously active CUBE reloads by design.
- CUBE uses the virtual IP address to communicate Smart Licensing information.
- For SIP-SIP TLS calls, configure both the active and standby CUBE as trust points to a common external CA Server.
- TCP sessions are not preserved during the failover. Remote user agents are expected to reestablish TCP sessions (using port 5060) before sending subsequent messages.
- Call Admission Control (CAC) state is maintained through switchover. After Stateful Switchover, no calls are allowed if the CAC limit is reached before the switchover.
- Up to six multimedia lines in the SDP are checkpointed for CUBE high availability. From Cisco IOS XE Release 3.17 onwards, SDP Passthru (up to two m-lines) calls are also checkpointed.

- Survivability.tcl preservation is supported from Cisco IOS XE Release 3.17 onwards for Unified Customer Voice Portal (CVP) deployments.

- SRTP-RTP, SRTP-SRTP, and SRTP Passthru are supported.

**Note**

Redundancy control traffic that is exchanged between CUBE-1 and CUBE-2 is not secured natively and displays SRTP encryption keys in cleartext. If SRTP is used, you must secure this traffic by configuring a transport IPsec tunnel between the two interfaces that are used as the redundancy control link.

- Port channel is supported for both RG control data and traffic interfaces only from Cisco IOS XE 16.3.1 onwards.

*Figure 73: Additional Supported Options for CUBE HA*

- LTI-based transcoder call flow preservation is supported from Cisco IOS XE Release 3.15 onwards and requires the same DSP module capacity on both active and standby in the same slot or subslot.

- From release Cisco IOS-XE 3.11 onwards, upon failover, you can move the previously active CUBE to a PROTECTED state to avoid the reload.

- While deploying High Availability pair with Application Centric Infrastructure (ACI), perform one of the following:
  - Disable IP data plane learning on the VRF.
    Refer to IP Data-plane Learning for details.
  - Use an intermediate Layer 3 switch between the High Availability pair and the ACI deployment. This Layer 3 switch prevents the ACI from directly learning the CUBE IP address and its associated MAC addresses.

**Restrictions**

- 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 (Cisco Unified SRST) or TDM Gateway colocation on CUBE HA is not supported.

• Routers connected through Metropolitan Area Network (MAN) Ethernet regardless of latency are not supported.

• Out-of-band DTMF (Notify or KPML) is not supported post switchover. Only rtp-nte to rtp-nte and voice-inband to voice-inband DTMF works after the switchover.

• Media-flow around and UC Services API (Cisco Unified Communications Manager Network-Based Recording) are not supported.

• You cannot terminate Wide Area Network (WAN) on CUBE directly or Data HA on either side. Both active and standby routers must be in the same Data Center and connected to the same physical switch.

• The Courtesy Callback (CCB) feature is not supported if a callback was registered with Cisco Unified Customer Voice Portal (CVP) and then a switchover was done on CUBE.

• You cannot configure a secondary IP address for the interfaces.

• If the redundancy group ID is same for the two different CUBE HA pairs, then the keepalive interface that is used for checkpointing RG control and data traffic must be in a different subnet or VLAN.

• One CUBE must have lower IPs across all the three interfaces on the same CUBE platform. For instance, CUBE-1 must have lower IP addresses in Gig0/0/0 interface compared with CUBE-2 Gig0/0/0 interface.

• CUBE box-to-box high availability requires same priority and threshold to be configured on both CUBE-1 and CUBE-2.

How to Configure CUBE High Availability on Cisco ASR 1000 Series Router

Before You Begin

• Use the same hardware platform, including the cards and their positioning.

• Place both Active and Standby routers physically in the same location, which is connected to the same Ethernet LAN.

• If there are currently dual RPs or ESPs in the Cisco ASR 1006 Router, remove the extra RP or ESP and reload the router before configuring the redundancy mode.

• Use Cisco IOS-XE Release 3.11 and or later on both active and standby routers.

• Ensure that you have the required licenses for using CUBE in High Availability mode. For detailed information, see Cisco Unified Border Element Data Sheet. In addition to an ASR1000 platform license (Advanced IP or Advanced Enterprise) and CUBE session licenses, a Firewall/NAT Stateful Inter-Chassis Redundancy License (Part number: FLSASR1-FWNAT-R) is also required for Box-to-Box High Availability configurations.
Configure Inbox High Availability

Enable inbox redundancy.

Example:

Router>enable
Router#configure terminal
Router(config)#redundancy
Router(config-r)#mode sso
Router(config-r)#end
Router(config)#copy run start /* This is to save the configuration */

Configure Box-to-Box High Availability

SUMMARY STEPS

1. Disable inbox and software redundancy.
2. Configure the Redundancy Group (RG).
3. Configure interface tracking.
4. Configure the interfaces.
5. Configure SIP Binding.
6. (Optional) If H.323 calls are involved, enable H.323 binding.
7. Configure the Punt Policing feature.
8. Configure the RG group under voice service voip. This enables Box-to-box CUBE HA.
9. Configure the Media Inactivity timer.
10. Reload the router.
11. Configure the peer router.
12. Point the attached devices to the CUBE Virtual IP (VIP) address.

DETAILED STEPS

Step 1

Disable inbox and software redundancy.

a) Disable software redundancy.

Example:

Disable software redundancy:

Router>enable
Router#configure terminal
Router(config)#redundancy
Router(config-r)#mode none

Example:

Disable the inbox redundancy if you are using ASR1006 router:
Router>enable
Router#configure terminal
Router(config)#redundancy
Router(config-r)#mode rpr

b) Save the running configuration to a text file in the bootflash.

   Example:

   Router>enable
   Router#copy running-configuration bootflash:<filename>

   In the preceding command, provide a name of your preference for `<filename>`.

c) Force the router to go into ROMMON mode upon next reload and erase the existing configuration from the NVRAM:

   Example:

   Router>enable
   Router#configure terminal
   Router(config)#config-register 0x0
   Router(config)#write erase

d) Reload the router.

   Example:

   Router>enable
   Router#reload

e) At ROMMON prompt, reset the IOSXE_Dual_IOS variable to disable the software redundancy.

   Example:

   rommon1>IOSXE_Dual_IOS=0
   rommon2>sync

f) Boot the image from the bootflash or harddisk, or from the network.

   Example:

   rommon1>boot bootflash:isr4400-universalk9.03.13.02.S.154-3.S2-ext.SPA.bin

g) When the router is up, reapply the old configuration by copying the configuration file to the running-configuration.

   Example:

   Router>enable
   Router#copy bootflash:sampleconfig running-configuration

h) Change the config register back to a nonzero value.

   Example:

   Router>enable
   Router#config-register 0x2102

Step 2

Configure the Redundancy Group (RG).

a) Enter application redundancy mode.

   Example:
Router>enable
Router#configure terminal
Router(config)#redundancy
Router(config-r)#mode none
Router(config-red)#application redundancy
Router(config-red-app)#group 1

b) Configure a name for the redundancy group.

Example:
Router(config-red-app-grp)#name cube-ha

where cube-ha is the name of the redundancy group.

c) Specify the initial priority and failover threshold for a redundancy group.

Example:
Router(config-red-app-grp)#priority 100 failover threshold 75

where 100 is the priority value and 75 is the threshold value. Both routers should have the same priority and threshold values.

d) Configure the timers for delay and reload.

Example:
Router(config-red-app-grp)#timers delay 30 reload 60

Delay timer which is the amount of time to delay the RG group’s initialization and role negotiation after the interface comes up.

Default: 30 seconds. Range is 0-10000 seconds.

Reload timer is the amount of time to delay RG group initialization and role-negotiation after a reload.

Default: 60 seconds. Range is 0-10000 seconds.

e) Configure the interface used to exchange keepalive and hello messages between the router pair.

Example:
Router(config-red-app-grp)#control GigabitEthernet0/0/2 protocol 1

where GigabitEthernet0/0/2 is the interface and protocol 1 is the protocol instance that is attached to the interface.

f) Configure the interface that is used for checkpointing of data traffic.

Example:
Router(config-red-app-grp)#data GigabitEthernet0/0/2

g) Configure RG group tracking.

Example:
Router(config-red-app-grp)#track 1 shutdown
Router(config-red-app-grp)#track 2 shutdown

If you want protected mode, enter the following command:

Router(config-red-app-grp)#track 3 shutdown

h) Specify the protocol instance that will be attached to a control interface and enters redundancy application protocol configuration mode.
Example:
Router(config-red-app-grp)#protocol 1

i) Configure the two timers for hellotime and holdtime.

Example:
Router(config-red-app-grp)#timers hellotime 3 holdtime 10

hellotime—Interval between successive hello messages.
Default is 3 seconds. Range is 250 milliseconds-254 seconds.

holdtime—The interval between the receipt of a hello message and the presumption that the sending router has failed. This duration has to be greater than the hellotime.
Default is 10 seconds. Range is 750 milliseconds-255 seconds.

We recommend that you configure the holdtime timer that is configured to be at least 3 times the value of the hellotimer timer.

Step 3
Configure interface tracking.

The `track` command is used in RG to track the voice traffic interface state so that the active router initiates switchover after the traffic interface is down.

Configure the following commands at the global level to track the status of the interface.

Router(config)#track 1 interface GigabitEthernet0/0/0 line-protocol
Router(config)#track 2 interface GigabitEthernet0/0/1 line-protocol

If you want protected mode, enter the following command:
Router(config)#track 3 interface GigabitEthernet0/0/2 line-protocol

Step 4
Configure the interfaces.

a) Configure the redundancy interface identifier for the redundancy group.

Required for generating a Virtual MAC (VMAC) address. You must use the same rii ID value on the interface of each router (active and standby) that has the same Virtual IP address.

If there is more than one Box-to-box HA pair on the same LAN, each pair MUST have unique rii IDs on their respective interfaces (to prevent collision). `show redundancy application group all` must indicate the correct local and peer information.

Example:

Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#ip address 203.0.113.10 255.255.0.0
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 1

Router(config)#interface GigabitEthernet0/0/1
Router(config-if)#ip address 192.0.2.1 255.255.255.0
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 2

b) Associate the interface with the redundancy group created.

Example:
c) Configure interface for RG control and data.

Example:

```plaintext
Router(config)#interface GigabitEthernet0/0/2
Router(config-if)#ip address 198.51.100.1 255.255.255.0
Router(config-if)#media-type rj45
Router(config-if)#negotiation auto
```

Step 5

Configure SIP Binding.

Configure CUBE to bind SIP messages to the interface that is configured with a Virtual IP address (VIP) for the RG group employed.

Example:

```plaintext
Router(config)#dial-peer voice 1 voip
Router(config-dial-peer)#session protocol sipv2
Router(config-dial-peer)#incoming called-number 2000
Router(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/0/0
Router(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/0/0
Router(config-dial-peer)#codec g711ulaw
Router(config-dial-peer)#

Router(config)#dial-peer voice 2 voip
Router(config-dial-peer)#destination-pattern 2000
Router(config-dial-peer)#session protocol sipv2
Router(config-dial-peer)#session target ipv4:203.0.113.13
Router(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/0/1
Router(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/0/1
Router(config-dial-peer)#codec g711ulaw
```

Step 6

(Optional) If H.323 calls are involved, enable H.323 binding.

Under the interface used by H.323, configure voip-bind with its source address equal to the interface’s VIP for the RG group employed.

Example:

```plaintext
Router#voice service voip
Router(conf-voi-serv)#h323
Router(conf-serv-h323)#call preserve limit-media-detection
Router(conf-serv-h323)#no h225 timeout keepalive

Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#ip address 203.0.113.10 255.255.0.0
Router(config-if)#media-type rj45
Router(config-if)#negotiation auto
Router(config-if)#redundancy rii 1
Router(config-if)#redundancy group 1 ip 9.13.25.123 exclusive
Router(config-if)#h323-gateway voip interface
Router(config-if)#h323-gateway voip bind srcaddr 203.0.113.12

Router(config)#interface GigabitEthernet0/0/1
Router(config-if)#ip address 192.0.2.1 255.255.255.0
Router(config-if)#media-type rj45
Router(config-if)#negotiation auto
```
Step 7 Configure the Punt Policing feature.

SIP packets towards the virtual IP address and physical IP address match different punt-cause codes. The punt-rate of the virtual IP address with a punt-cause of 60, is lower than the punt-rate of the physical IP address.

To ensure that the behaviour of the SIP packets towards virtual and physical IP address remains the same, you must increase the punt-rate of the virtual IP address by using the `platform punt-policer` command in global configuration mode.

**Note** For Cisco IOS XE Releases 16.6.7, 16.9.4, 16.11.1, 16.12.1, 17.1.1 and later releases, you do not need to increase the punt-rate.

**Example:**

```
Router(config)#platform punt-policer 60 40000
```

In the preceding example, the punt-rate of the virtual IP address (punt-cause 60) is increased from the default value of 2000–40000.

The following table provides details of the fields of the CLI.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>platform punt-policer</code></td>
<td>Configure the Punt Policing feature.</td>
</tr>
<tr>
<td>60</td>
<td><code>punt-cause</code>—Punt cause. Range is 1–107. Punt cause of the virtual interface is 60.</td>
</tr>
<tr>
<td>40000</td>
<td><code>punt-rate</code>—Rate limit in packets per second. Range is 10–146484.</td>
</tr>
</tbody>
</table>

**Note** The default punt rate value of the virtual IP address and the physical IP address varies with the router platform.

**Note** The default and maximum setting are platform-specific. Default value is optimal for most deployments. Change the rate only when suggested by Cisco Support.

Step 8 Configure the RG group under `voice service voip`. This enables Box-to-box CUBE HA.

**Example:**

```
Router#voice service voip
Router(conf-voi-serv)#redundancy-group 1
```

For enabling protected mode:

```
Router#voice service voip
Router(conf-voi-serv)#no redundancy-reload
```

Step 9 Configure the Media Inactivity timer.

The Media Inactivity Timer enables the active and standby router pair to monitor and disconnect calls if no Real-Time Protocol (RTP) packets are received within a configurable time period.
For the SIP calls, the switched over calls are cleared with signaling (as signaling information is preserved for switched calls).

The Media Inactivity Timer releases TCP-based and H.323-based calls. This is used to guard against any hung sessions resulting from the failover when a normal call disconnect does not clear the call.

You must configure the same duration for the Media Inactivity Timer on both routers. The default value is 30 seconds for SIP and H.323 calls. The sample configuration is as follows:

**Example:**

```
Router(config)#ip rtcp report interval 9000
Router(config)#gateway
Router(config-gateway)#media-inactivity-criteria all
Router(config-gateway)#timer receive-rtp 1200
Router(config-gateway)#timer receive-rtcp 5
```

SIP and H.323 call legs are cleared once the RTCP timer expires.

**Step 10**

Reload the router.

Once all the preceding configurations are completed, you must save the configurations, and reload the router.

**Example:**

```
Router>enable
Router#reload
```

**Step 11**

Configure the peer router.

Follow the preceding steps to configure the standby router. Make sure that you use the correct IP addresses.

**Step 12**

Point the attached devices to the CUBE Virtual IP (VIP) address.

The IP-PBX, Unified SIP Proxy, or service provider must route the calls to CUBE’s Virtual IP address.

HA configuration does not handle SIP and H.323 messages to the CUBE’s physical IP addresses.

For H.323 calls, you must disable the keepalive messages in Unified CM configuration.

a. Go to **System** menu, and choose **Service Parameters**. At the bottom of the Service Parameters, enable **Advanced**.

b. Set the **Allow TCP KeepAlives for H323** to False.

c. After this setting is saved, restart the CallManager Services.

### Configuration Examples

The following sample configuration assumes interfaces Gig0/0/0 is used for incoming calls, and Gig0/0/1 is used for outgoing calls, and Gig0/0/2 is used for redundancy.

**Active Router Configurations**

```
Router1# show run
Building configuration...
Current configuration : 3082 bytes
```
! Last configuration change at 21:33:13 UTC Sun Sep 19 2010
!
version 15.1
service timestamps debug datetime msec
service timestamps log datetime msec
!
hostname b2bred2
!
!
boot-start-marker

boot system flash bootflash:asr1000rp2-adventerprisek9.BLD_MCP_DEV_LATEST_201008
24_091509.bin
boot-end-marker
!
!
!

vrf definition Mgmt-intf
!

exit-address-family
!
address-family ipv6
!
exit-address-family
!
!
logging buffered 777777777
no logging console
enable secret 5 $1$kan3$QsGBuVkgGDZgBlg41SrsW
!
no aaa new-model
!
!
ip source-route
!
!
multilink bundle-name authenticated
!
!
voice service voip
media bulk-stats
allow-connections h323 to h323
allow-connections h323 to sip
allow-connections sip to h323
allow-connections sip to sip
redundancy-group 1
h323
emptycapability

call preserve limit-media-detection
no h225 timeout keepalive
h245 passthru tcsnonstd-passthru
sip
early-offer forced
midcall-signaling passthru
!
!
voice iec syslog
!
!
track 1 interface GigabitEthernet0/0/0 line-protocol
track 2 interface GigabitEthernet0/0/1 line-protocol
!
!
redundancy
mode none
application redundancy

! group 1
name voice-b2bha
priority 100 failover threshold 75
timers delay 30 reload 60
control GigabitEthernet0/0/2 protocol 1
data GigabitEthernet0/0/2
track 1 shutdown
track 2 shutdown
protocol 1
timers hello-time 3 holdtime 10
!
!
ip ftp username bhks
ip ftp password bhks
!
interface GigabitEthernet0/0/0
ip address 203.0.113.10 255.255.255.0
media-type rj45
negotiation auto
no mop enabled
redundancy rii 1
redundancy group 1 ip 203.0.113.12 exclusive
h323-gateway voip interface
h323-gateway voip bind srcaddr 203.0.113.12
!
interface GigabitEthernet0/0/1
ip address 192.0.2.1 255.255.255.0
media-type rj45
negotiation auto
redundancy rii 2
redundancy group 1 ip 192.0.2.3 exclusive
h323-gateway voip interface
h323-gateway voip bind srcaddr 192.0.2.3
!
interface GigabitEthernet0/0/2
ip address 198.51.100.1 255.255.255.0
media-type rj45
negotiation auto
!
interface GigabitEthernet0
vrf forwarding Mgmt-intf
no ip address
negotiation auto
!
no ip http server
no ip http secure-server
ip rtcp report interval 9000
ip route 0.0.0.0 0.0.0.0 9.44.0.1
!
logging esm config
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
!
control-plane
!
!
dial-peer voice 10 voip
destination-pattern 140854.....
session protocol sipv2
session target ipv4:y.y.y.y
voice-class sip bind control source-interface GigabitEthernet0/0/1
voice-class sip bind media source-interface GigabitEthernet0/0/1
codec g711ulaw
no vad
!
dial-peer voice 20 voip
session protocol sipv2
session target ipv4:203.0.113.13
incoming called-number 140854....
voice-class sip bind control source-interface GigabitEthernet0/0/0
voice-class sip bind media source-interface GigabitEthernet0/0/0
codec g711ulaw
no vad
!
gateway
media-inactivity-criteria all
!timer receive-rtcp 5
timer receive-rtp 1200
!
line con 0
exec-timeout 0 0
stopbits 1
line vty 0 4
no login
!
exception data-corruption buffer truncate
end

Standby Router Configurations

Router2#sh run
Building configuration...
Current configuration : 2606 bytes
!
! Last configuration change at 21:34:07 UTC Sun Sep 19 2010
!
version 15.1
service timestamps debug datetime msec
service timestamps log datetime msec
!
hostname b2bred1
!
boot-start-marker
boot system flash bootflash:asr1000rp2-adventerprisek9.BLD_MCP_DEV_LATEST_201008
24_091509.bin
boot-end-marker
!
!vrf definition Mgmt-intf
!
address-family ipv4
exit-address-family
!
address-family ipv6
exit-address-family
!
logging buffered 77777777
no logging console
!
no aaa new-model
!

ip source-route
!
!!
multilink bundle-name authenticated
!
!
voice service voip
media bulk-stats
allow-connections h323 to h323
allow-connections h323 to sip
allow-connections sip to h323
allow-connections sip to sip
redundancy-group 1
h323
emptycapability
call preserve limit-media-detection
no h225 timeout keepalive
h245 passthru tcsnonstd-passthru
sip
early-offer forced
midcall-signaling passthru
!
!
voice iec syslog
!
!
track 1 interface GigabitEthernet0/0/0 line-protocol
track 2 interface GigabitEthernet0/0/1 line-protocol
!
!
redundancy
mode none
application redundancy
  group 1
    name voice-b2bha
    priority 100 failover threshold 75
    timers delay 30 reload 60
    control GigabitEthernet0/0/2 protocol 1
    data GigabitEthernet0/0/2
    track 1 shutdown
    track 2 shutdown
    protocol 1
    timers hellotime 3 holdtime 10
!
!
ip ftp username bhks
ip ftp password bhks
!
!
interface GigabitEthernet0/0/0
  ip address 203.0.113.11 255.255.255.0
  media-type rj45
  negotiation auto
  redundancy rii 1
  redundancy group 1 ip 203.0.113.12 exclusive
  h323-gateway voip interface
  h323-gateway voip bind srcaddr 203.0.113.12
!
interface GigabitEthernet0/0/1
  ip address 192.0.2.2 255.255.255.0
  media-type rj45
negotiation auto
redundancy ril 2
redundancy group 1 ip 192.0.2.3 exclusive
h323-gateway voip interface
h323-gateway voip bind srcaddr 192.0.2.3
interface GigabitEthernet0/0/2
   ip address 198.51.100.2 255.255.255.0
   media-type rj45
   negotiation auto
   !
interface GigabitEthernet0
   vrf forwarding Mgmt-intf
   no ip address
   shutdown
   negotiation auto
   !
no ip http server
no ip http secure-server
ip rtcp report interval 9000
ip route 0.0.0.0 0.0.0.0 9.44.0.1
!
logging esm config
!
control-plane
!
dial-peer voice 10 voip
destination-pattern 140854........
session protocol sipv2
session target ipv4:y.y.y.y
voice-class sip bind control source-interface GigabitEthernet0/0/1
voice-class sip bind media source-interface GigabitEthernet0/0/1
codec g711ulaw
no vad
!
dial-peer voice 20 voip
session protocol sipv2
session target ipv4:203.0.113.13
incoming called-number 140854........
voice-class sip bind control source-interface GigabitEthernet0/0/0
voice-class sip bind media source-interface GigabitEthernet0/0/0
codec g711ulaw
no vad
!
gateway
media-inactivity-criteria all
timer receive-rtcp 5
timer receive-rtp 1200
!
line con 0
   exec-timeout 0 0
   stopbits 1
line vty 0 4
   no login
!
exception data-corruption buffer truncate
end
Verify Your Configuration

Verify Redundancy State on Active and Standby Routers

Use the `show redundancy application group all` command to display the redundancy inter-device states.

**Step 1  Active Router:**

**Example:**

```
Router#show redundancy application group all
```

Faults states Group 1 info:
Runtime priority: [100]
    RG Faults RG State: Up.
    Total # of switchovers due to faults: 0
    Total # of down/up state changes due to faults: 2
Group ID:1
Group Name:voice-b2bha

Administrative State: No Shutdown
Aggregate operational state : Up
My Role: ACTIVE
Peer Role: STANDBY
Peer Presence: Yes
Peer Comm: Yes
Peer Progression Started: Yes

RF Domain: btob-one
    RF state: ACTIVE
    Peer RF state: STANDBY HOT

RG Protocol RG 1
------------------
Role: Active
Negotiation: Enabled
Priority: 100
Protocol state: Active
Ctrl Intf(s) state: Up
Active Peer: Local
Standby Peer: address 203.0.113.11, priority 100, intf Gi0/0/2

Log counters:
role change to active: 1
role change to standby: 0
disable events: rg down state 1, rg shut 0
ctrl intf events: up 1, down 2, admin_down 1
reload events: local request 0, peer Request 0

RG Media Context for RG 1
--------------------------
Ctx State: Active
Protocol ID: 1
Media type: Default
Control Interface: GigabitEthernet0/0/2
Configured Hello timer: 3000
Current Hello timer: 3000, Hold timer: 10000
Peer Hello timer: 3000, Peer Hold timer: 10000
Stats:
Pkts 27719, Bytes 1718578, HA Seq 0, Seq Number 27719, Pkt Loss
Step 2  
**Standby Router:**

**Example:**

Router#show redundancy application group all

```plaintext
Faults states Group 1 info:
  Runtime priority: [100]
    RG Faults RG State: Up.
    Total # of switchovers due to faults: 0
    Total # of down/up state changes due to faults: 2
Group ID:1
Group Name:voice-b2bha

Administrative State: No Shutdown
Aggregate operational state : Up
My Role: STANDBY
Peer Role: ACTIVE
Peer Presence: Yes
Peer Comm: Yes
Peer Progression Started: Yes

RF Domain: btob-one
  RF state: STANDBY HOT
  Peer RF state: ACTIVE

RG Protocol RG 1
------------------
  Role: Standby
  Negotiation: Enabled
  Priority: 100
  Protocol state: Standby-hot
  Ctrl Intf(s) state: Up
  Active Peer: address 203.0.113.10, priority 100, intf Gi0/0/2
  Standby Peer: Local
  Log counters:
    role change to active: 0
    role change to standby: 1
    disable events: rg down state 1, rg shut 0
    ctrl intf events: up 1, down 2, admin down 1
    reload events: local request 0, peer request 0

RG Media Context for RG 1
--------------------------
  Ctx State: Standby
  Protocol ID: 1
  Media type: Default
  Control Interface: GigabitEthernet0/0/2
  Current Hello timer: 3000
  Configured Hello timer: 3000, Hold timer: 10000
  Peer Hello timer: 3000, Peer Hold timer: 10000
  Stats:
    Pkts 27832, Bytes 1725584, HA Seq 0, Seq Number 27832, Pkt Loss 0
```

0
  Authentication not configured
  Authentication Failure: 0
  Reload Peer: TX 0, RX 0
  Resign: TX 0, RX 0
  Standby Peer: Present. Hold Timer: 10000
  Pkts 27700, Bytes 941800, HA Seq 0, Seq Number 27708, Pkt Loss 0
Verify Call State After Switchover

Use the `show voice high-availability summary` command to verify the following:

- The checkpointing of calls on the standby router after a switchover
- The media-inactivity count on the active router when the calls are over
- Native and non-native (preserved) calls when both call types are present
- Presence of leaked RTP, HA, SPI sessions

Active Router

Router# `show voice high-availability summary`

-------- HA Message Sizes --------
SCCPAPP Data Size: 412
SIPSPI Data Size: 4260
H323SPI Data Size: 2164
RTSPI Data Size: 861
CCAPI Data Size: 188
VOIPRTP Data Size: 158
HA Data Size: 68
Total Data Size: 4842

-------- Voice HA DB INFO --------
Number of calls in HA DB: 0
Number of calls in HA sync pending DB: 0
Number of current SWMTP calls with HA: 0

First a few entries in HA DB:

First a few entries in Sync Pending DB:

-------- Voice HA Process INFO --------
Active process current tick: 92663
Active process number of tick events pending: 0
Active process number of tick events processed: 0

-------- Voice HA RF INFO --------
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]:
RF Domain: 0x2 [RG: 1]
Voice HA Client Name: VOIP RG CLIENT
Voice HA RF Client ID: 4054
Voice HA RF Client SEQ: 418
My current RF state ACTIVE (13)
Peer current RF state STANDBY HOT (8)
Current VOIP HA state [LOCAL / PEER]: [ACTIVE (13) / STANDBY HOT (8)]

Voice HA Active and Standby are in sync.
System has experienced switchover.

--------- Voice HA CF INFO ---------
Voice HA CF for RG(1):
local ip = 9.13.25.190; remote ip = 9.13.25.191
local port = 4026; remote port = 4025
CF setup done: TRUE
Role is Active. Client side stats:
Received checkpointing requests: 0
Wrote to sockets: 0
Checkpoint buffer in use: 0
Pending transmit events: 0

--------- Voice HA COUNTERS ---------
Total number of checkpoint requests sent (Active): 0
Total APP DATA sent on Active: 0
Total CREATE sent on Active: 0
Total MODIFY sent on Active: 0
Total DELETE sent on Active: 0
Total number of checkpoint requested received (Standby): 0
Total APP DATA received on Standby: 0
Total CREATE received on Standby: 0
Total MODIFY received on Standby: 0
Total DELETE received on Standby: 0
Media Inactivity event count: 0
Max Media Up time since Call Create: 0 msecs
Queue Failed for MEDIA EVENT - move entry 2 sync pending db: 0
Queue Failed for CREATE - move entry to sync pending db: 0
Queue Failed for MODIFY - move entry to sync pending db: 0
Queue Failed for DELETE - move entry to sync pending db: 0
No Entry Found when processing Tick Queue Event: 0
Entry Deleted - never checkpointed :0
Added Element to Multi Delete List: 0
Standby received Delete as part of Multi-Delete Message: 0
Active Sent Multi Delete Message to Standby: 0
Standby Callback Invoked by CF: 0
Standby Callback Invoked by CF - Negotiation Message: 0
Standby Callback Invoked by CF - No Mag Header: 0
Standby Callback Invoked by CF - ISSU Xform Fail: 0
Standby Callback Invoked by CF - malloc VOIP Buffer fail: 0
Standby Callback Invoked by CF - enqueue to voip ha fail: 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Checkpoint overflow: 0
HA DB element pool overrun count: 0
HA DB aux element pool overrun count: 0
HA DB insertion failure count: 0
HA DB deletion failure count: 0
Tick event pool overrun count: 0
Tick event queue overrun count: 0
Checkpoint send failure count - ISSU Transform Failure: 0
Checkpoint send failure count - CF failed: 0
Checkpoint get buffer failure count: 0
Checkpoint Received IPC Flow ON from CF: 0
Checkpoint Received IPC Flow OFF from CF: 0

Standby Router

Router#show voice high-availability summary

========= HA Message Sizes ======
SCCP APP Data Size: 412
SIP-SPI Data Size: 4260
H323-SPI Data Size: 2164
RTSPI Data Size: 861
CCAPI Data Size: 188
VOIPRTP Data Size: 158
HA Data Size: 68
Total Data Size: 4842

======== Voice HA DB INFO ========
Number of calls in HA DB: 0
Number of calls in HA sync pending DB: 0
Number of current SWMTP calls with HA: 0
First a few entries in HA DB:
First a few entries in Sync Pending DB:

-------- Voice HA Process INFO --------
Active process current tick: 46846
Active process number of tick events pending: 0
Active process number of tick events processed: 0

----------- Voice HA RF INFO -----------
FUNCTIONING RF DOMAIN: 0x2
RF Domain: 0x0
Voice HA Client Name: VOIP RF CLIENT
Voice HA Client ID: 1345
Voice HA 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 Client ID: 4054
Voice HA Client SEQ: 418
My current RF state STANDBY HOT (8)
Peer current RF state ACTIVE (13)
Current VOIP HA state [LOCAL / PEER] :
[STANDBY HOT (8) / ACTIVE (13)]

RF Domain: 0x2
Voice HA Standby is not available.
System has not experienced switchover.

-------- Voice HA CF INFO --------
Voice HA CF for RG(1):
  local ip = 203.0.113.10; remote ip = 203.0.113.11
  local port = 4025; remote port = 4026
Verify SIP IP Address Bindings

Use the `show sip-ua status` command to verify SIP IP address bindings.

```
Router# 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
```
Verify Current CPU Use

Use the `show process cpu history` to verify the CPU utilization percentage at regular intervals.

Check CPU utilization before performing a switchover and proceed with a forced failover only when the CPU utilization is less than 70%. You can also use `show process cpu sorted` command repeatedly to know the CPU utilization for a particular process.

Force a Manual Failover for Testing

Box-to-box redundancy on the Cisco ASR 1000 Series Router platform supports full stateful switchover of calls. This means the media (RTP) and signaling information of the calls is preserved.

You can expect that switchovers occurring in real environments, where there is a constant mixture of calls in transient (call setup or being modified) and established state, result in some dropped calls during a failover.

To check that your configuration is correct, you can force a manual switchover.

A switchover involves the active router reloading, while the standby router takes over and becomes the new active router, processing incoming calls and maintaining the media streams and signaling information for calls until they are complete. The new active router continues to act as such until another switchover occurs. There is no pre-emption mechanism on Box-to-box redundancy.

Before you begin

Before you start a manual switchover, take note of the following:

- Monitor the CPU utilization % on the active and standby router pair. The active router has the higher CPU utilization as it is actively handling the calls, while the standby router shows little CPU utilization.
- Ensure that you perform a manual switchover when the CPU utilization of the active router is not more than 70%.
- Use the `show voip rtp connection` command to make sure that existing calls across the active and standby router pair are in sync.

You can achieve manual switchovers in various ways:

Procedure

- Initiate the manual switchover by using the CLI `redundancy application reload group RG ID self` on the active router.
- Reload of the active router
Troubleshoot High Availability Issues

Use the following show and debug commands to troubleshoot High Availability issues:

- `show redundancy application group all`
- `show redundancy application transport clients`
- `show redundancy client domain all | inc VOIP RG`
- `show voice high-availability summary`
- `show voip fpi stats`
- `debug voip rtp session`
- `debug voice high-availability all`
- `debug voip fpi all`
- `debug redundancy application group {config | faults | media | protocol | rii transport | vp}`

---

**Note**

On every switchover after reload, you must enable the debugs on the new standby router.

**Note**

Do not turn on many debugs on a system carrying high volume of active call traffic.

**Troubleshooting Tips**

- Check for proper HA states on both the active and standby router in the output of the show commands, like `show redundancy application group`.
- Perform incoming and outgoing ping tests with the VIPs employed.
- In the presence of active calls, look for the use of any physical interface’s IP address in the output of `show voip rtp connections` on both the active and standby routers. VIP must be used in both the show outputs and the debugs.
- In the output of `show voip rtp connection | inc Found` and `show call active voice compact | inc Total` on both the active and standby routers, check for any large number of mismatched calls.
- To debug problems, enable the corresponding debug options:
  - VoIP RTP
  - VoIP FPI
  - VoIP HA
• SPIs (SIP, H.323, SCCPAPP)
Chapter 57

High Availability on Cisco CSR 1000V or C8000V Cloud Services Routers

Note
Cisco Cloud Services Router 1000V Series (CSR 1000V) is no longer supported from Cisco IOS XE Bengaluru 17.4.1a onwards. If you are using CSR 1000V, you have to upgrade to Cisco Catalyst 8000V Edge Software (Catalyst 8000V). For End-of-Life information on CSR 1000V, see End-of-Sale and End-of-Life Announcement for the Select Cisco CSR 1000v Licenses.

The High Availability (HA) feature allows you to benefit from the failover capability of Cisco Unified Border Element (CUBE) on two routers, one active and one standby. When the active router goes down for any reason, the standby router takes over seamlessly, preserving and processing your calls.

Figure 74: Cisco CUBE High Availability

- About vCUBE High Availability on CSR 1000V or C8000V Cloud Services Routers, on page 759
- How to Configure vCUBE High Availability on Cisco CSR 1000v or C8000V, on page 766
- Troubleshoot High Availability Issues, on page 769

About vCUBE High Availability on CSR 1000V or C8000V Cloud Services Routers

Cisco Unified Border Element running on Cisco CSR 1000v Series Cloud Services Router and C8000V is called Virtual CUBE (vCUBE). vCUBE leverages Redundancy Group (RG) Infrastructure to provide high availability. HA is configured between two vCUBE Cisco CSR 1000v or C8000V instances running on either the same host or across different hosts that are connected through the same switch.

You can configure vCUBE on Cisco CSR 1000v or C8000V running on virtualized hosts listed in the Cisco Unified Border Element Data Sheet.
Box-to-Box Redundancy

Box-to-box redundancy enables configuring a pair of routers to act as back up for each other. In the router pair, the active router is determined based on the failover conditions. The router pair continuously exchange status messages. CUBE session information is checkpointed across the active and standby router. This enables the standby router to immediately take over all CUBE call processing responsibilities when the active router becomes unavailable.

Redundancy Group (RG) Infrastructure

A group of redundant interfaces form a Redundancy Group. The active and standby routers are connected by a configurable control link and data synchronization link. The control link is used to communicate the redundancy state for each router. The data synchronization link is used to transfer stateful information to synchronize the stateful database for the calls and media flows. Each pair of redundant interfaces is configured with the same unique ID number, also known as the Redundancy Interface Identifier (RII).

A Virtual IP address (VIP) is configured on interfaces that connect to the external network. All signaling and media is sourced from and sent to the Virtual IP address. External devices such as Cisco Unified Communication Manager, uses VIP as the destination IP address for the calls traversing through Cisco UBE.

The following figure shows the redundancy group configured for a pair of routers with a single outgoing interface.

Figure 75: Redundancy Group Configuration
Network Topology

Figure 76: Virtual CUBE High Availability
We recommend that you keep the following in mind when enabling this topology:

- Connect the Cisco CSR 1000v or C8000V running on the server to the virtual switch within the virtualized host. Then connect the virtual switches to external switches using the physical host interfaces. The virtual switch routes the traffic internally between the virtual machines and also connects the external networks.

- Configure high-availability connectivity using redundancy on virtual switch to avoid checkpointing failures.

In a scenario where the physical switch is down and there is no redundancy configured on virtual switch, the active router continues to process calls as it tracks only the status of virtual switch (which is up). At the same time, the standby router assumes the role of active router as it does not receive keepalive messages from the active router through the physical switch. Hence checkpointing fails. To avoid such scenarios, we recommend you to configure high availability connectivity using redundancy on virtual switch.

- Do not track the switches that are used to connect non-networking end devices or LAN, to determine uplink failures.
• Connect the redundancy group control and data interfaces in the CUBE HA pair to the same physical switch to avoid any latency in the network.

• The RG control and data interfaces of the CUBE HA pair can be connected through a back-to-back cable or using a switch. However, it is recommended to use Portchannel for the RG control and data interfaces for redundancy. A single connection using back-to-back cable or switch presents a single point of failure due to a faulty cable, port, or switch, resulting in error state where both routers are Active.

• If the RG ID is the same for the two different CUBE HA pairs, keepalive interface for check-pointing the RG control and data, and traffic must be in a different subnet or VLAN.

Note
This recommendation is applicable only if you connect using a switch, not by back-to-back cables.

• You can configure a maximum of two redundancy groups. Hence, there can be only two Active and Standby pairs within the same network.

Note
This recommendation is applicable only if you connect using a switch, not by back-to-back cables.

• Source all signaling and media from and to the virtual IP address.

• Always save the running configuration to avoid losing it due to router reload during a failover.

• Virtual Routing and Forwarding
  • Define Virtual Router Forwarding (VRF) in the same order on both Active and Standby routers for an accurate synchronization of data.
  • You can configure VRFs only on the traffic interface (SIP and RTP). Do not configure VRF on redundancy group control and data interface.
  • VRF configurations on both the Active and Standby router must be identical. VRF IDs checkpoints for the calls before and after switchover (includes VRF-based RTP port range).

• Manually copy the configurations from one router to the other.

• Replicating the configuration on the Standby router does not commit to the startup configuration; it is the running configuration. You must run the `write memory` command to commit the changes that are synchronized from the Active router on the Standby router.

Considerations and Restrictions

The following is a list of further considerations and restrictions you should know before configuring this topology:
Considerations

Before you configure High Availability feature on Cisco UBE, understand its behavior, and also know the Cisco IOS XE Software version that is required for supporting the call processing features to work seamlessly after switchover.

- Only active calls are checkpointed (Calls that are connected with 200 OK or ACK transaction completed).
- When you apply and save the configuration for the first time, the platform must be reloaded.
- For H.323, and TCP-based calls, media preservation is supported after the failover, but session signaling is not preserved.
- If you have Cisco Unified Customer Voice Portal (CVP) in your network, we recommend that you configure TCP session transport for the SIP trunk between CVP and CUBE.
- Upon failover, the previously active CUBE reloads by design.
- CUBE uses the virtual IP address to communicate Smart Licensing information.
- For SIP-SIP TLS calls, configure both the active and standby CUBE as trust points to a common external CA Server.
- TCP sessions are not preserved during the failover. Remote user agents are expected to reestablish TCP sessions (using port 5060) before sending subsequent messages.
- Call Admission Control (CAC) state is maintained through switchover. After Stateful Switchover, no calls are allowed if the CAC limit is reached before the switchover.
- Up to six multimedia lines in the SDP are checkpointed for CUBE high availability. From Cisco IOS XE Release 3.17 onwards, SDP Passthru (up to two m-lines) calls are also checkpointed.
- Survivability.tcl preservation is supported from Cisco IOS XE Release 3.17 onwards for Unified Customer Voice Portal (CVP) deployments.
- SRTP-RTP, SRTP-SRTP, and SRTP Passthru are supported.

**Note**

Redundancy control traffic that is exchanged between CUBE-1 and CUBE-2 is not secured natively and displays SRTP encryption keys in cleartext. If SRTP is used, you must secure this traffic by configuring a transport IPsec tunnel between the two interfaces that are used as the redundancy control link.

- Port channel is supported for both RG control data and traffic interfaces only from Cisco IOS XE 16.3.1 onwards.
• While deploying High Availability pair with Application Centric Infrastructure (ACI), perform one of the following:
  
  • Disable IP data plane learning on the VRF. Refer to IP Data-plane Learning for details.
  
  • Use an intermediate Layer 3 switch between the High Availability pair and the ACI deployment. This Layer 3 switch prevents the ACI from directly learning the CUBE IP address and its associated MAC addresses.

Restrictions

- Geographic stateful switchover is not supported.
- Calls in the transient state at the time of switchover are not preserved.
- 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 (Cisco Unified SRST) or TDM Gateway colocation on CUBE HA is not supported.
- Routers connected through Metropolitan Area Network (MAN) Ethernet regardless of latency are not supported.
- Out-of-band DTMF (Notify or KPML) is not supported post switchover. Only rtp-nte to rtp-nte and voice-inband to voice-inband DTMF works after the switchover.
- Media-flow around and UC Services API (Cisco Unified Communications Manager Network-Based Recording) are not supported.
- You cannot terminate Wide Area Network (WAN) on CUBE directly or Data HA on either side. Both active and standby routers must be in the same Data Center and connected to the same physical switch.
- The Courtesy Callback (CCB) feature is not supported if a callback was registered with Cisco Unified Customer Voice Portal (CVP) and then a switchover was done on CUBE.
How to Configure vCUBE High Availability on Cisco CSR 1000v or C8000V

Before You Begin

- Use Cisco IOS-XE Release 3.11 and or later on both active and standby routers.
- Ensure that you have the required licenses for configuring high availability. For detailed information, see Cisco Unified Border Element Data Sheet.

Configure High Availability

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the Redundancy Group.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Configure the interfaces.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Configure SIP Binding.</td>
<td>Configure CUBE to bind SIP messages to the interface that is configured with a virtual IP address (VIP) for the Redundancy Group employed.</td>
</tr>
</tbody>
</table>

Example:

Router(config)#dial-peer voice 1 voip
Router(config-dial-peer)#session protocol sipv2
Router(config-dial-peer)#incoming called-number 2000
Router(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/0/0
Router(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/0/0
Router(config-dial-peer)#codec g711ulaw
Router(config-dial-peer)!

Router(config)#dial-peer voice 2 voip
Router(config-dial-peer)#destination-pattern 2000
Router(config-dial-peer)#session protocol sipv2
Router(config-dial-peer)#session target ipv4:203.0.113.13
Router(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/0/1
Router(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/0/1
Router(config-dial-peer)#codec g711ulaw
### Command or Action

**Step 4** Configure the Punt Policing feature.

**Example:**
```
Router(config)#platform punt-policer 60 40000
```

In the preceding example, the punt-rate of the virtual IP address (punt-cause 60) is increased from the default value of 2000–40000.

**Purpose**

SIP packets toward the virtual IP address and physical IP address match different punt-cause codes. The punt-rate of the virtual IP address with a punt-cause of 60, is lower than the punt-rate of the physical IP address.

To ensure that the behaviour of the SIP packets toward virtual and physical IP address remains the same, you must increase the punt-rate of the virtual IP address by using the `platform punt-police` command in global configuration mode.

**Note** For Cisco IOS XE Releases 16.6.7, 16.9.4, 16.11.1, 16.12.1, 17.1.1 and later releases, you do not need to increase the punt-rate.

The following table provides details of the fields of the CLI.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>platform punt-police</code></td>
<td>Configures the Punt Policing feature.</td>
</tr>
<tr>
<td><code>punt-cause</code></td>
<td>Punt cause. Range is 1–107. Punt cause of the virtual interface is 60.</td>
</tr>
<tr>
<td><code>punt-rate</code></td>
<td>Rate limit in packets per second. Range is 10–146484.</td>
</tr>
</tbody>
</table>

**Note** The default punt rate value of the virtual IP address and the physical IP address varies with the router platform.

**Note** The default and maximum setting are platform-specific. Default value is optimal for most deployments. Change the rate only when suggested by Cisco Support.

**Step 5** Configure the Redundancy Group under `voice service voip`. This Redundancy Group creation enables Box-to-Box CUBE high availability.

**Example:**
```
Router#voice service voip
Router(conf-voi-serv)#no redundancy-reload
```

**Example:**
```
Router#voice service voip
Router(conf-voi-serv)#redundancy-group 1
```

**Purpose**

For enabling protected mode:
```
Router#voice service voip
Router(conf-voi-serv)#no redundancy-reload
```

**Step 6** Configure the Media Inactivity timer.

**Example:**
```
Router(config)#platform punt-police
Router(config)#punt-cause 60 40000
```

**Purpose**

The Media Inactivity Timer enables the Active and Standby router pair to monitor and disconnect calls if no Real-Time Protocol (RTP) packets are received within a configurable time period.
### Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router(config)#ip rtcp report interval 9000</td>
<td>For the SIP calls, the switched over calls are cleared with signaling (as signaling information is preserved for switched calls). The Media Inactivity Timer releases TCP-based calls. This is used to guard against any hung sessions resulting from the failover when a normal call disconnect does not clear the call. You must configure the same duration for the Media Inactivity Timer on both routers. The default value is 30 seconds for SIP calls. The sample configuration is as follows: SIP call legs are cleared once the RTCP timer expires.</td>
</tr>
<tr>
<td>Router(config)#gateway</td>
<td></td>
</tr>
<tr>
<td>Router(config-gateway)#media-inactivity-criteria all</td>
<td></td>
</tr>
<tr>
<td>Router(config-gateway)#timer receive-rtp 1200</td>
<td></td>
</tr>
<tr>
<td>Router(config-gateway)#timer receive-rtcp 5</td>
<td></td>
</tr>
</tbody>
</table>

### Step 7

Reload the router.

**Example:**

Router>enable
Router#relaod

### Step 8

Configure the peer router.

Follow the preceding steps to configure the standby router. Make sure that you use the correct IP addresses.

### Step 9

Point the attached devices to the CUBE Virtual IP (VIP) address.

The IP-PBX, Unified SIP Proxy, or service provider must route the calls to CUBE’s virtual IP address.

High availability configuration does not handle SIP messages to the CUBE’s physical IP addresses.

### Configuration Example

**Active Router:**

```
voice service voip
no ip address trusted authenticate
allow-connections sip to sip
redundancy-group 1
sip
 bind control source-interface GigabitEthernet1
 bind media source-interface GigabitEthernet1
redundancy
application redundancy
group 1
 name cube_b2b_ha_1
 priority 125 failover threshold 75
timers delay 30 reload 60
control GigabitEthernet2 protocol 1
data GigabitEthernet2
track 1 shutdown
protocol 1
 name cube_b2b_ha_1
 authentication text sol_ha1
```
track 1 interface GigabitEthernet1 line-protocol
!
interface GigabitEthernet1
  ip address 192.0.2.1 255.255.255.0
  negotiation auto
  no mop enabled
  no mop sysid
  redundancy rii 102
  redundancy group 1 ip 192.0.2.3 exclusive
!
interface GigabitEthernet2
  ip address 198.51.100.1 255.255.255.0
  negotiation auto
  no mop enabled
  no mop sysid

Standby Router:
  voice service voip
  no ip address trusted authenticate
  allow-connections sip to sip
  redundancy-group 2
  sip
    bind control source-interface GigabitEthernet1
    bind media source-interface GigabitEthernet1
!
redundancy
  application redundancy
  group 2
    name cube_b2b_ha_1
    priority 100 failover threshold 75
    timers delay 30 reload 60
    control GigabitEthernet2 protocol 1
    data GigabitEthernet2
    track 1 shutdown
    protocol 1
    name cube_b2b_ha_1
    authentication text sol_ha1
!
track 1 interface GigabitEthernet1 line-protocol
!
interface GigabitEthernet1
  ip address 192.0.2.2 255.255.255.0
  negotiation auto
  no mop enabled
  no mop sysid
  redundancy rii 102
  redundancy group 2 ip 192.0.2.3 exclusive
!
interface GigabitEthernet2
  ip address 198.51.100.2 255.255.255.0
  negotiation auto
  no mop enabled
  no mop sysid

Troubleshoot High Availability Issues

Use the following show and debug commands to troubleshoot High Availability issues:

- show redundancy application group all
- show redundancy application transport clients
Troubleshoot High Availability Issues

- show redundancy client domain all | inc VOIP RG
- show voice high-availability summary
- show voip fpi stats
- debug voip rtp session
- debug voice high-availability all
- debug voip fpi all
- debug redundancy application group {config | faults | media | protocol | rii transport | vp}

Note
Do not turn on a large number of debugs on a system carrying high volume of active call traffic.
High Availability on Cisco Integrated Services Routers (ISR-G2)

The High Availability (HA) feature allows you to benefit from the failover capability of Cisco Unified Border Element (CUBE) on two routers, one active and one standby. When the active router goes down for any reason, the standby router takes over seamlessly, preserving and processing your calls.

Figure 79: Cisco CUBE High Availability

- About CUBE High Availability on Cisco ISR-G2, on page 771
- How to Configure CUBE High Availability on Cisco ISR-G2, on page 791
- Verify Your Configurations, on page 805
- Troubleshoot High Availability Issues, on page 808

About CUBE High Availability on Cisco ISR-G2

CUBE supports Box-to-box redundancy on Cisco Integrated Services Router Generation 2 Router (ISR-G2) and uses Hot Standby Routing Protocol (HSRP) technology to provide High Availability.

Box-to-Box Redundancy

Box-to-box redundancy enables configuring a pair of routers to act as back up for each other. In the router pair, active router is determined based on the failover conditions. The router pair continuously exchange status messages. Cisco UBE session information is checkpointed across the active and standby router. This enables the standby router to immediately take over all Cisco UBE call processing responsibilities when the active router becomes unavailable.
Hot Standby Router Protocol (HSRP)

Hot Standby Router Protocol (HSRP) technology provides high network availability by not relying on any single router for routing IP traffic from hosts on the network.

By sharing an IP address and a MAC (Layer 2) address, two or more routers consist a virtual router group that is called a Standby group or HSRP group. This HSRP group acts as a single virtual router to hosts on the LAN. HSRP is used to select an active router and a standby router in an HSRP group. The active router forwards packets that the host sends to the virtual router group. Active and standby routers continually exchange periodic HSRP messages once the protocol has completed the router selection process.

HSRP monitors both the inside and outside interfaces. If any of the interfaces go down, the whole router is considered down and the standby router takes over the responsibilities of the active router.

The RTP streams of established calls are checkpointed between the active and standby routers through the HSRP protocol. Therefore the media streams of established calls are preserved over the HSRP failover from the active to the standby routers. Calls in the transient state (calls that are not established yet, or are in the process of being modified with transfer or hold function) at the time of failover are disconnected.

---

Note

For redundant solutions that use HSRP, CDRs are only generated by the active router.

HSRP Features

- **Preemption**—The HSRP preemption feature enables the router with the highest priority to immediately become the active router. Priority is determined as follows.
  1. Priority value that you configure.
  2. IP address.

  In each case, higher value is of a greater priority.

- **Preempt Delay**—The preempt delay feature allows you to delay the preemption for a configurable time period. Preempt delay allows the router to populate its routing table before becoming the active router.

- **Interface Tracking**—Allows you to specify details of another interface on the router of the HSRP group. Interface tracking helps to monitor the change in the HSRP priority of a given HSRP group.

Network Topology

This section describes how to configure the following dual-attached and single-attached network topology. The dual-attached network topology is the most common configuration, in which an active and standby pair of routers is used in a SIP trunk deployment between a Cisco Unified Communications Manager (Unified CM) and a service provider (SP) SIP trunk for PSTN access. It is also possible to configure CUBE HSRP Box-to-box redundancy with a single-attached network topology.
In these topologies, both active and standby routers have the same configuration and both platforms are connected through a physical switch across similar interfaces. This is required for Cisco UBE HA to work. For example, the CUBE-1 and CUBE-2 interface towards WAN must terminate on the same switch. Multiple interfaces or sub-interfaces can be used on either LAN or WAN side. Also, one Cisco UBE has a lower IP address across all three interfaces on the same Cisco UBE platform. This criteria decides the HSRP active state.

We recommend that you keep the following in mind when configuring these topologies:

- Configure all interfaces of an HSRP group with the same priority.
- The active and standby router pair, and interface combination on a particular LAN must have a unique HSRP group number.

Configure CUBE High Availability Using HSRP

Before you begin

It is recommended that you have the knowledge of the following topics:

- How to configure and use Cisco IOS® Voice.
- How to configure and use CUBE.
- How HSRP high availability works on general router platforms.

Components used:
• Minimum software release of CUBE 8.5 (Cisco IOS Release 15.1.2T), implemented on a Cisco 2900 or 3900 Series Integrated Service Router Generation 2 (ISR G2).

• Two identical ISR G2s equipped with the UC Technology Package license (SL-29-UC-K9 or SL-39-UC-K9) installed, 1G DRAM memory, and Cisco IOS Software Release 15.1.2T or later.

• Both routers must be physically located on the same Ethernet LAN.

• The CUBE configuration of both routers is identical and must be manually copied from one router to the other.

• SIP-SIP call flows.

SUMMARY STEPS

1. Enable CUBE and CUBE Redundancy.
2. Enable HSRP.
3. Configure HSRP Communication Transport.
4. Configure HSRP on Interfaces.
5. Configure HSRP Timers.
6. Configure Media Inactivity Timer.
7. Configure SIP Binding to HSRP Address
8. Reload Routers.
9. Point Attached Softswitches to CUBE HSRP Virtual Address

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 CUBE and CUBE Redundancy.</td>
<td>Enables CUBE on both routers. Also, enables CUBE redundancy and call check pointing on both routers.</td>
</tr>
</tbody>
</table>
| | Example:  
voice service voip  
mode border-element  
allow-connections sip to sip | |
| | Example:  
voice service voip  
redundancy | |
| Step 2 | Enable HSRP. | Enables router redundancy schemes on both routers, where: |
| | Example:  
redundancy inter-device  
scheme standby SB | • Scheme — redundancy state tracking scheme. |
| |  | • Standby — enable standby (HSRP) state tracking scheme. |
| |  | • SB — the HSRP standby group name. |
| Step 3 | Configure HSRP Communication Transport. | Enables the HSRP Inter-Device Communication Transport. |
| | Example:  
Active Configuration: | • ipc zone default — Configures the Inter-Device Communication Protocol (IPC) and enters IPC zone configuration mode. Use this command to initiate the |
### Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipc zone default</code></td>
<td>communication link between the Active and Standby devices.</td>
</tr>
<tr>
<td><code>association 1</code></td>
<td>• association 1 — Configures an association between the two devices and enters the IPC association configuration mode. Under this, configure the details of the association such as transport protocol, local port, local IP address, remote port and remote IP address. Valid association IDs range from 1 to 255. There are no default association IDs.</td>
</tr>
<tr>
<td><code>no shutdown</code></td>
<td>• no shutdown — Restarts a disabled association and its associated transport protocol. For any changes to the transport protocol parameters, this association must be shut down.</td>
</tr>
<tr>
<td><code>protocol sctp</code></td>
<td>• protocol sctp — Configures SCTP as the transport protocol for this association and enables SCTP protocol configuration mode.</td>
</tr>
<tr>
<td><code>local-port port_num</code></td>
<td>• local-port port_num — Defines the local SCTP port number to use in order to communicate with the redundant peer.</td>
</tr>
<tr>
<td><code>local-ip ip_addr</code></td>
<td>• remote-port port_num — Defines the remote SCTP port number to use in order to communicate with the redundant peer.</td>
</tr>
<tr>
<td><code>remote-port port_num</code></td>
<td>• remote-ip ip_addr — Defines the remote IP address of the peer router used to communicate with the local device. All remote IP addresses must point to the same device.</td>
</tr>
<tr>
<td><code>remote-ip ip_addr</code></td>
<td><strong>Note</strong> The local-port and the remote-port must be set to 5000 on the Active and Standby routers.</td>
</tr>
</tbody>
</table>

**Standby Configuration:**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipc zone default</code></td>
<td><strong>Exit from the local-sctp prompts to configure the remote Stream Control Transmission Protocol (SCTP) parameters as follows:</strong></td>
</tr>
<tr>
<td><code>association 1</code></td>
<td>XFR-2(config)#ipc zone default</td>
</tr>
<tr>
<td><code>no shutdown</code></td>
<td>XFR-2(config-ipczone)#association 1</td>
</tr>
<tr>
<td><code>protocol sctp</code></td>
<td>XFR-2(config-ipczone-assoc)#no shutdown</td>
</tr>
<tr>
<td><code>local-port port_num</code></td>
<td>XFR-2(config-ipczone-assoc)# protocol sctp</td>
</tr>
<tr>
<td><code>local-ip ip_addr</code></td>
<td>Pod4-3925(config-ipc-protocol-sctp)#local-port 5000</td>
</tr>
<tr>
<td><code>remote-port port_num</code></td>
<td>XFR-2(config-ipc-protocol-sctp)#local-ip 10.10.24.13</td>
</tr>
<tr>
<td><code>remote-ip ip_addr</code></td>
<td>XFR-2(config-ipc-protocol-sctp)#remote-port 5000</td>
</tr>
<tr>
<td><code>standby delay minimum 30 reload 60 standby version 2</code></td>
<td>XFR-2(config-ipc-remote-sctp)#remote-ip 10.10.24.14</td>
</tr>
</tbody>
</table>

**Step 4** Configure HSRP on Interfaces.

**Example:**

**Active Configuration:**

```plaintext
interface GigabitEthernet0/0
  ip address 10.10.25.14 255.255.255.0
duplex auto
keepalive
speed auto
standby delay minimum 30 reload 60
standby version 2
```

**Configure HSRP Inter-Device Communication Transport.**

- **0 / 6** — Defines the Standby Group Number.
- **keepalive** — Enables keepalive for HSRP in order to monitor up/down events.
- **standby delay** — Delays HSRP initialization until the physical interface is up.
### Command or Action

- standby 0 ip 10.10.25.1  
- standby 0 preempt  
- standby 0 priority 50  
- standby 0 track 2 decrement 10  
- standby 0 name SB  

```  
!  
interface GigabitEthernet0/1  
ip address 10.10.24.14 255.255.255.0  
duplex auto  
speed auto  
media-type rj45  
standby delay minimum 30 reload 60  
standby version 2  
standby 6 ip 10.10.24.1  
standby 6 priority 50  
standby 6 track 1 decrement 10  
standby 6 ip 10.10.24.1  
standby 6 priority 50  
standby 6 name SB  
standby 6 track 2 decrement 10  

!  
interface GigabitEthernet0/0  
ip address 10.10.25.13 255.255.255.0  
duplex auto  
speed auto  
media-type rj45  
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 name SB  
standby 0 track 2 decrement 10  
```

### Purpose

- **minimum**—Defines the minimum time in seconds to delay HSRP group initialization after an interface comes up. This minimum delay period applies to all subsequent interface events.

- **reload**—Defines the time period to delay after the router is reloaded.

- **standby x ip**—Defines the virtual IPv4 IP address shared between the Active and Standby devices. This command enables the HSRP on the interface.

- **standby x preempt**—Allows the router to become the active router when the priority is higher than all other HSRP-configured routers in the hot standby group. If you do not use the **standby preempt** command in the configuration for a router, that router does not become the active router, even if the priority is higher than all other routers.

- **standby x priority**—Defines the Hot Standby priority used in order to choose the active router. It ranges from 1 to 255 where 1 denotes the lowest priority and 255 the highest priority. In cases where the standby priority is the same, the device with the higher IP address assumes the role of the Active router.

- **standby x name**—Defines the name of the standby group which matches the scheme defined in Step 2 (SB). For multiple HSRP groups, the same standby name is used as only one standby scheme is allowed in the configurations.

- **standby 6 track 1 decrement 10**—Defines priority tracking.

In order to avoid race conditions when a router boots up and an interface comes up to establish contact (Hello) between the Active and Standby routers, it is recommended to configure this:

```
!  
interface GigabitEthernet0/0  
standby delay minimum 30 reload 60  
```

### Step 5

Configure HSRP Timers.

There are two important HSRP timers:

- **Hello Timer**: The interval between successive HSRP Hello messages from a given router. This timer can be configured in seconds or milliseconds under the HSRP interface. The default value is 3 seconds.

- **Hold Timer**: The interval between the receipt of a Hello message and the presumption that the sending
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>router has failed. This time can be configured in seconds or milliseconds under the HSRP interface. The default value is 8 seconds.</td>
</tr>
<tr>
<td></td>
<td>The HSRP Hello and Hold Timers are set to their default values. Therefore, they do not show up explicitly in the configurations. The recommended values for the Hello/Hold Timers are the default values.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> If you should use non-default values, you must configure each router to use the same Hello time and Hold timer values.</td>
</tr>
<tr>
<td></td>
<td>The Hello and Hold timers can be configured under the HSRP interface with this CLI:</td>
</tr>
<tr>
<td></td>
<td><code>Router(config-if)#standby 0 timers</code></td>
</tr>
<tr>
<td></td>
<td><code>?</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;1-254&gt;</code> Hello interval in seconds</td>
</tr>
<tr>
<td></td>
<td><code>msec</code> Specify hello interval in milliseconds</td>
</tr>
<tr>
<td></td>
<td><code>Router (config-if)#standby 0 timers 2 ?</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;3-255&gt;</code> Hold time in seconds</td>
</tr>
<tr>
<td></td>
<td><code>msec</code> Specify hold interval in milliseconds</td>
</tr>
<tr>
<td></td>
<td><code>Router(config-if)#standby 0 timers 2 msec 40</code></td>
</tr>
<tr>
<td></td>
<td>In the previous configuration, the Hello timer is set to 2 seconds and the Hold timer to 40 milliseconds.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> You can lower the timer settings to speed up failover or preemption. However, in order to avoid increased CPU use and unnecessary standby state flapping, it is recommended not to set the Hello timer at less than 1 second, and the Hold timer at less than 4 seconds.</td>
</tr>
</tbody>
</table>

**Step 6** Configure Media Inactivity Timer.  

The Media Inactivity Timer enables the Active/Standby router pair to monitor and disconnect calls if no Real-Time Protocol (RTP) packets are received within a configurable time period.  

When RTP packets for a call are not received by the Active/Standby router, the SIP Media Inactivity Timer releases the session. This is used to guard against any hung sessions that might have resulted from the failover in the event that a normal call disconnect does not clear the call.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The same duration for the Media Inactivity Timer must be configured on both routers. The default value is 28 seconds. This timer is configured as follows:</td>
</tr>
<tr>
<td>ip rtcp report interval 3000</td>
<td>Note</td>
</tr>
<tr>
<td>gateway</td>
<td>The media inactivity detection timer is defined with two CLI commands. One command configures the Real-time Transport Control Protocol (RTCP) report interval, and another defines the multiplying factor M (this also identifies the mode of detection with Cisco IOS Release 12.4(4)T). The controlling mechanism is accomplished through the configuration of application CLI.</td>
</tr>
<tr>
<td>media-inactivity-criteria all</td>
<td>Media inactive timer = M * ip rtcp report interval</td>
</tr>
<tr>
<td>timer receive-rtp 86400</td>
<td>The inactivity detection is supported in two modes based on which timer multiplying factor configuration (M factor) is used:</td>
</tr>
<tr>
<td>timer receive-rtcp</td>
<td>timer media-inactive: This mode is available in Cisco IOS Release 12.4(4)T, where detection is based on DSP statistics (it uses RTP-only mechanism; packets sent or received are considered active). If both directions are absent, it is considered inactive. This timer is enabled or disabled with the use of application CLI, which can also be used in order to control notification.</td>
</tr>
</tbody>
</table>

**Step 7** Configure SIP Binding to HSRP Address

Configures the CUBE SIP messaging in order to use the HSRP virtual address in SIP messaging:

```bash
dial-peer voice 100 voip
description to-SIP
voice-class sip bind control source-interface GigabitEthernet0/0
```
Once HSRP is configured under the physical interface and the `bind` command has been issued, calls to the physical IP address will fail. This is because the SIP listening socket is now bound to the virtual IP address but the signaling packets use the physical IP address, and therefore cannot be handled.

Once all the above configurations have been completed, the redundancy `show` output is as follows:

```
XFR-2#show redundancy inter-device
Redundancy inter-device state:
RF_INTERDEV_STATE_INIT
Pending Scheme: Standby (Will not take effect until next reload)
  Pending Groupname: b2bha
  Scheme: <NOT CONFIGURED>
  Peer present: UNKNOWN
  Security: Not configured
```

When you reload the router, the HSRP configuration is enabled as follows:

**Active Router Configuration:**

```
XFR-2#show redundancy inter-device
Redundancy inter-device state:
RF_INTERDEV_STATE_ACT
  Scheme: Standby
    Groupname: b2bha Group State: Active
    Peer present: RF_INTERDEV_PEER_COMM
    Security: Not configured
```

**Standby Router Configuration:**

```
CUBE_XFR#show redundancy inter-device
Redundancy inter-device state:
RF_INTERDEV_STATE_STDBY
  Scheme: Standby
    Groupname: b2bha Group State: Standby
    Peer present: RF_INTERDEV_PEER_COMM
    Security: Not configured
```

The CUCM, IP-PBX, SIP proxy or SP SBCs or SP softswitches that route calls to CUBE must use the HSRP virtual address in their SIP messaging. SIP messages to the CUBE physical IP addresses are not handled with an HSRP configuration.

---

**Step 8** Reload Routers.

**Step 9** Point Attached Softswitches to CUBE HSRP Virtual Address
Example

Sample Configurations for Dual-Attached CUBE HSRP Redundancy

In these configurations, the HSRP Hello and Hold timers use their default values of 3 and 8 seconds respectively, and are not shown explicitly in the CLI output.

Active Configuration:

```plaintext
ipc zone default
    association 1
        no shutdown
        protocol sctp
            local-port 5000
            local-ip 10.10.24.14
            remote-port 5000
            remote-ip 10.10.24.13

voice service voip
    mode border-element
    allow-connections sip to sip
    redundancy
!
redundancy inter-device
    scheme standby SB
!
redundancy
!
interface GigabitEthernet0/0
    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
!
interface GigabitEthernet0/1
    ip address 10.10.24.14 255.255.255.0
    duplex auto
    media-type rj45
    speed auto
    standby delay minimum 30 reload 60
    standby version 2
    standby 6 ip 10.10.24.1
    standby 6 priority 50
    standby 6 track 1 decrement 10
!
ip rtcp report interval 3000
!
track 1 interface GigabitEthernet0/0 line-protocol
!
track 2 interface GigabitEthernet0/1 line-protocol
!
dial-peer voice 100 voip
    description to-SIP
    destination-pattern 9T
```
session protocol sipv2
session target ipv4:x.x.x.x
voice-class sip bind control source-interface GigabitEthernet0/0
voice-class sip bind media source-interface GigabitEthernet0/0
!
dial-peer voice 200 voip
description to-CUCM
destination-pattern 555....
session protocol sipv2
session target ipv4:y.y.y.y
voice-class sip bind control source-interface GigabitEthernet0/1
voice-class sip bind media source-interface GigabitEthernet0/1
!
gateway
  media-inactivity-criteria all
timer receive-rtcp 5
timer receive-rtp 1200

Standby Configuration:

ipc zone default
association 1
  no shutdown
protocol sctp
  local-port 5000
  local-ip 10.10.24.13
  remote-port 5000
  remote-ip 10.10.24.14
!
voice service voip
  mode border-element
  allow-connections sip to sip
  redundancy

redundancy inter-device
  scheme standby SB

redundancy

interface GigabitEthernet0/0
  ip address 10.10.25.13 255.255.255.0
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 name SB
standby 0 track 2 decrement 10
!
interface GigabitEthernet0/1
  ip address 10.10.24.13 255.255.255.0
duplex auto
speed auto
media-type rj45
standby delay minimum 30 reload 60
standby version 2
standby 6 ip 10.10.24.1
standby 6 priority 50
standby 6 preempt
standby 6 track 1 decrement 10
ip rtcp report interval 3000
!
track 1 interface GigabitEthernet0/0 line-protocol
!
track 2 interface GigabitEthernet0/1 line-protocol
!
dial-peer voice 100 voip
   description to-SIP
   destination-pattern 9T
   session protocol sipv2
   session target ipv4:x.x.x.x
   voice-class sip bind control source-interface GigabitEthernet0/0
   voice-class sip bind media source-interface GigabitEthernet0/0
!
dial-peer voice 200 voip
   description to-CUCM
   destination-pattern 555....
   session protocol sipv2
   session target ipv4:y.y.y.y
   voice-class sip bind control source-interface GigabitEthernet0/1
   voice-class sip bind media source-interface GigabitEthernet0/1
!
gateway
   media-inactivity-criteria all
   timer receive-rtcp 5
   timer receive-rtp 1200

Sample Configuration for Single-Attached CUBE HSRP Redundancy

While a dual-attached CUBE is the most common configuration, especially for SP SIP trunk connections, it is also possible to configure CUBE HSRP box-to-box redundancy with a single-attached CUBE deployment as given in this section.

Active Router Configuration:

ipc zone default
   association 1
   no shutdown
   protocol sctp
      local-port 5000
      local-ip 1.2.175.8
      remote-port 5000
      remote-ip 1.2.175.12
!
voice service voip
   mode border-element
   allow-connections sip to sip
   redundancy
   sip
      bind control source-interface GigabitEthernet0/0
      bind media source-interface GigabitEthernet0/0
!
redundancy inter-device
   scheme standby SB
!
redundancy
!
interface GigabitEthernet0/0
   ip address 1.2.175.8 255.255.0.0
   duplex auto
   speed auto
   keepalive
   standby delay minimum 30 reload 60
standby version 2
standby 0 ip 1.2.175.100
standby 0 preempt
standby 0 priority 50
standby 0 name SB
standby 0 track 1 decrement 10

! ip rtcp report interval 3000
!
dial-peer voice 5 voip
  description to-SIP-application
  destination-pattern 9T
  session protocol sipv2
  session target ipv4:x.x.x.x
!
dial-peer voice 9 voip
  description to-CUCM
  destination-pattern 555....
  session protocol sipv2
  session target ipv4:y.y.y.y
!
gateway
  media-inactivity-criteria all
  timer receive-rtcp 5
  timer receive-rtp 1200

Standby Router Configuration:

ipc zone default
  association 1
  no shutdown
  protocol sctp
  local-port 5000
  local-ip 1.2.175.12
  remote-port 5000
  remote-ip 1.2.175.8
!
voice service voip
  mode border-element
  allow-connections sip to sip
  redundancy
  sip
    bind control source-interface GigabitEthernet0/0
    bind media source-interface GigabitEthernet0/0
!
redundancy inter-device
  scheme standby SB
!
redundancy
!
interface GigabitEthernet0/0
  ip address 1.2.175.12 255.255.0.0
  duplex auto
  speed auto
  standby delay minimum 30 reload 60
  standby version 2
  standby 0 ip 1.2.175.100
  standby 0 priority 50
  standby 0 preempt
  standby 0 name SB
  standby 0 track 1 decrement 10
!
ip rtcp report interval 3000
Verify Redundancy State

SUMMARY STEPS

1. Use the `show redundancy inter-Router` and `show redundancy state` commands to verify the redundancy state.

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> Use the <code>show redundancy inter-Router</code> and <code>show redundancy state</code> commands to verify the redundancy state.</td>
<td>The following are sample outputs for the commands <code>show redundancy inter-Router</code> and <code>show redundancy state</code> before inter-router configuration:</td>
</tr>
</tbody>
</table>

```
XFR-2#show redundancy inter-Router
Redundancy inter-Router state:
RF_INTERDEV_STATE_PNC_NO_HSRP
Scheme: Standby
Groupname: b2bha Group State: Init
Protocol: <NOT CONFIGURED>

XFR-2#show redundancy states
my state = 3 -NEGOTIATION
peer state = 1 -DISABLED
Mode = Simplex
Unit ID = 0

Maintenance Mode = Disabled
Manual Swact = disabled (system is simplex (no peer unit))
Communications = Down Reason: Simplex mode

client count = 14
client_notification_TMR = 30000 milliseconds
RF debug mask = 0x0
```

The following is a sample output for the command `show redundancy inter-Router` after the inter-router configuration and before router reload:
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>XFR-2#show redundancy inter-Router</td>
<td>Redundancy inter-Router state: RF_INTERDEV_STATE_INIT Pending Scheme: Standby (Will not take effect until next reload) Pending Groupname: b2bha Scheme: &lt;NOT CONFIGURED&gt; Peer present: UNKNOWN Security: Not configured</td>
</tr>
<tr>
<td>CUBE_XFR#show redundancy inter-Router</td>
<td>Redundancy inter-Router state: RF_INTERDEV_STATE_PNC_NO_HSRP Scheme: Standby Groupname: b2bha Group State: Init Peer present: UNKNOWN Security: Not configured</td>
</tr>
<tr>
<td>CUBE_XFR#show redundancy states</td>
<td>my state = 3 -NEGOTIATION peer state = 13 -ACTIVE Mode = Duplex Unit ID = 0 Maintenance Mode = Disabled Manual Swact = disabled (this unit is still initializing) Communications = Up client count = 14 client_notification_TMR = 30000 milliseconds RF debug mask = 0x0</td>
</tr>
<tr>
<td>CUBE_XFR#show redundancy inter-Router</td>
<td>Redundancy inter-Router state: RF_INTERDEV_STATE_ACT Scheme: Standby Groupname: b2bha Group State: Active Peer present: RF_INTERDEV_PEER_NO_COMM Security: Not configured XFR-2#show redundancy states</td>
</tr>
</tbody>
</table>

The following are sample outputs for the commands **show redundancy inter-Router** and **show redundancy state** after the router reload:

The following are sample outputs for the commands **show redundancy inter-Router** and **show redundancy state** during a switchover:
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Swact = disabled (system is simplex (no peer unit))</td>
<td></td>
</tr>
<tr>
<td>Communications = Up</td>
<td></td>
</tr>
<tr>
<td>client count = 14</td>
<td></td>
</tr>
<tr>
<td>client_notification_TMR = 30000 milliseconds</td>
<td></td>
</tr>
<tr>
<td>RF debug mask = 0x0</td>
<td></td>
</tr>
</tbody>
</table>

The following are sample outputs for the commands `show redundancy inter-Router` and `show redundancy state` after a switch over, but before the routers exchange Hello status messages:

```
CUBE_XFR#show redundancy inter-Router
Redundancy inter-Router state:
RF_INTERDEV_STATE_ACT
Scheme: Standby
Groupname: b2bha Group State: Active
Peer present: RF_INTERDEV_PEER_NO_COMM
Security: Not configured
```

```
XFR-2#show redundancy inter-Router
Redundancy inter-Router state:
RF_INTERDEV_STATE_HSRP_STDBY_PNC
Scheme: Standby
Groupname: b2bha Group State: Standby
Peer present: RF_INTERDEV_PEER_NO_COMM
Security: Not configured
```

The following are sample outputs for the commands `show redundancy inter-Router` and `show redundancy state` after the exchange of Hello status messages:

```
XFR-2#show redundancy inter-Router
Redundancy inter-Router state:
RF_INTERDEV_STATE_ACT
Scheme: Standby
Groupname: b2bha Group State: Active
Peer present: RF_INTERDEV_PEER_COMM
Security: Not configured
```

```
XFR-2#show redundancy state
my state = 13 -ACTIVE
peer state = 8 -STANDBY HOT
Mode = Duplex
Unit ID = 0
Maintenance Mode = Disabled
Manual Swact = disabled (peer unit not yet in terminal standby state)
```

```
Communications = Up
client count = 14
client_notification_TMR = 30000 milliseconds
RF debug mask = 0x0
```

```
CUBE_XFR#show redundancy inter-Router
```
Verify Call State After a Switchover

Use the `show voice high-availability summary` command to verify the following:

- The checkpointing of calls on the standby router after a switchover
- The media-inactivity count on the active router when the calls are over
- To check for native and nonnative (for example, preserved) calls when both types of calls are present
- To identify the presence of leaked RTP, HA, SPI sessions

**Verify checkpointing of calls on the standby router after a switchover**

In this example, 800 calls were checkpointed from active to standby after the switchover.

```
Router#show voice high-availability summary

---------- Voice HA DB INFO ----------
Number of calls in HA DB: 0
Number of calls in HA sync pending DB: 0
Number of calls in HA preserved session DB: 0

-----------------------------
First a few entries in HA DB:
-----------------------------

-----------------------------
First a few entries in Sync Pending DB:
-----------------------------

-----------------------------

---------- Voice HA Process INFO ----------
Active process current tick: 3100
Active process number of tick events pending: 0
```
Active process number of tick events processed: 0
voice service voip is configured to have redundancy

======== Voice HA RF INFO ========
Voice HA RF Client Name: VOIP RF CLIENT
Voice HA RF Client ID: 1345
My current rf state STANDBY NOT
Peer current rf state ACTIVE
Voice HA Standby is not available.
System has not experienced switchover.

======== Voice HA CF INFO ========
Voice HA CF Client Name: CHKPT VOIP SYMPHONY
Voice HA CF Client ID: 252
Voice HA CF Client Status: Peer NOT READY; TP flow ON.

======== Voice HA COUNTERS ========
Total number of checkpoint requests sent (Active): 0
Total number of checkpoint requested received (Standby): 971
Total CREATE received on Standby: 800
Total MODIFY received on Standby: 0
Total DELETE received on Standby: 800
Media Inactivity event count: 0
Checkpoint CREATE overflow: 0
Checkpoint MODIFY overflow: 0
Checkpoint DELETE overflow: 0
HA DB elememt pool overun count: 0
HA DB aux element pool overun count: 0
HA DB insertion failure count: 0
HA DB deletion failure count: 0
Tick event pool overun count: 0
Tick event queue overun count: 0
Checkpoint send failure count: 0
Checkpoint get buffer failure count: 0

Verify the media-inactivity count on the active router when the calls are over

In this example, 800 calls are cleared by the media-inactivity timer.

Router# show voice high-availability summary

======== Voice HA DB INFO ========
Number of calls in HA DB: 0
Number of calls in HA sync pending DB: 0
Number of calls in HA preserved session DB: 0

First a few entries in HA DB:

First a few entries in Sync Pending DB:

======== Voice HA Process INFO ========
Active process current tick: 4213
Active process number of tick events pending: 0
Active process number of tick events processed: 0
voice service voip is configured to have redundancy

======== Voice HA RF INFO ========
Voice HA RF Client Name: VOIP RF CLIENT  
Voice HA RF Client ID: 1345  
My current rf state ACTIVE  
Peer current rf state STANDBY HOT  
Voice HA Active and Standby are in sync.  
System has experienced switchover.

--------- Voice HA CF INFO ---------  
Voice HA CF Client Name: CHKPT VOIP SYMPHONY  
Voice HA CF Client ID: 252  
Voice HA CF Client Status: Peer READY; TP flow ON.

--------- Voice HA COUNTERS ---------  
Total number of checkpoint requests sent (Active): 971  
Total number of checkpoint requested received (Standby): 800  
Total CREATE received on Standby: 800  
Total MODIFY received on Standby: 0  
Total DELETE received on Standby: 0  
Media Inactivity event count: 800

- Checkpoint CREATE overflow: 0  
- Checkpoint MODIFY overflow: 0  
- Checkpoint DELETE overflow: 0  
- HA DB element pool overrun count: 0  
- HA DB aux element pool overrun count: 0  
- HA DB insertion failure count: 0  
- HA DB deletion failure count: 0  
- Tick event pool overrun count: 0  
- Tick event queue overrun count: 0  
- Checkpoint send failure count: 0  
   Checkpoint get buffer failure count: 0

Verify native and non-native (preserved) calls when both are present

The numbers of calls on the system are shown as follows:

- Total number of calls = "Number of calls in HA DB" + "Number of calls in HA sync pending DB". This is 100 + 50 = 150 in the example output below.

- Total number of preserved (nonnative) calls = "Number of calls in HA preserved session DB". This is 70 in the example output below.

- Total number of native calls (calls set up since the failover and therefore not preserved over the failover) is the difference in the previous two numbers. In this example, it is 150 - 70 = 80.

Router# show voice high-availability summary

--------- Voice HA DB INFO ---------  
Number of calls in HA DB: 100  
Number of calls in HA sync pending DB: 50  
Number of calls in HA preserved session DB: 70

Identify the presence of leaked RTP, HA, SPI Sessions

The total number of preserved (non-native) calls cleared by Media Inactivity is equal to the total CREATE received on standby router minus total DELETE received on standby router. Compare this number with the Media Inactivity event count and the number of media down events, as shown in the output of the show voip fpi stats command.

Router# show voice high-availability summary

--------- Voice HA DB INFO ---------
Number of calls in HA DB: 0
Number of calls in HA sync pending DB: 0
Number of calls in HA preserved session DB: 0

-------- Voice HA COUNTERS --------
Total number of checkpoint requests sent (Active): 971
Total number of checkpoint requested received (Standby): 800
Total CREATE received on Standby: 800
Total MODIFY received on Standby: 0
Total DELETE received on Standby: 0
Media Inactivity event count: 800

Considerations and Restrictions

The following is a list of further considerations and restrictions you should know before configuring this topology:

Considerations

• There are slight differences in the HSRP configuration between active and standby routers.
• Configuration synchronization between the active and standby router is manual.
• HSRP virtual addresses support only IPv4 addressing.
• Only active calls are checkpointed (Calls that are connected with 200 OK or ACK transaction completed).
• Upon failover, the previously active CUBE reloads by design.
• Multiple traffic (SIP/RTP) interfaces require Preemption and Interface Tracking.
• In High Availability deployments, CUBE uses a primary IP address to communicate the Smart Licensing information.
• Box-to-box redundancy configuration supports only SIP-SIP calls flows, the SIP transport can be either UDP-UDP or UDP-TCP.
• Port channel interfaces are supported only from Cisco IOS Release 15.6(3)M onwards.

Figure 82: Additional Supported Options for CUBE HA
Restrictions

- 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.
- Calls that involve supplementary services such as transcoding, DTMF-interworking, IVR, SIP-TLS, RSVP, STUN, RTP-SRTP conversion, or fax/modem features are not preserved during the failover.
- Box-to-box redundancy configuration supports multiple HSRP groups per router, but only a single HSRP group per physical interface.
- Loopback addresses with HSRP are not supported, the SIP bind command must use the HSRP virtual IP address.
- No support for media-flow around or UC Services API (Cisco Unified Communications Manager - Network-Based Recording).
- WANs cannot terminate directly on the CUBE or on data HSRP on either sides.
- Call Progress Analysis (CPA) calls (before to being transferred to the agent), SCCP-based media resources, Noise Reduction, Acoustic Shock Protection (ASP), and transrating calls are not checkpointed.
- Courtesy Callback (CCB) feature is not supported if a callback was registered with Cisco Unified Customer Voice Portal (CVP) and then a switchover was done on CUBE.

How to Configure CUBE High Availability on Cisco ISR-G2

Before You Begin

- Two identical ISR-G2s equipped with the UC Technology Package license (SL-29-UC-K9 or SL-39-UC-K9) installed, 1G DRAM memory, and Cisco IOS Software release 15.1.2T or later.
- Ensure that you have the required licenses for configuring High Availability. For detailed information, see Cisco Unified Border Element Data Sheet.

Configure High Availability

SUMMARY STEPS

1. Define the redundancy scheme.
2. Enable CUBE and CUBE redundancy.
3. Configure Inter-process Communication (IPC) protocol at the HSRP interface.
4. (Optional) Configure Virtual Route Forwarding (VRF) on the platform.
5. Configure HSRP on the interfaces.
6. Configure Interface Tracking.
7. Bind traffic to the respective interfaces.
8. Configure Media Inactivity feature.
9. Reload the routers.
10. Point the attached devices to the CUBE HSRP Virtual IP (VIP) address.

DETAILED STEPS

**Step 1** Define the redundancy scheme.

**Example:**

```
Router(config)#redundancy inter-device
Router(config-red-interdevice)#scheme standby SB
```

The following table provides details of the CLIs used in the configuration.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheme</td>
<td>Redundancy state tracking scheme</td>
</tr>
<tr>
<td>standby</td>
<td>Enables standby (HSRP) state tracking scheme</td>
</tr>
<tr>
<td>SB</td>
<td>HSRP standby group name</td>
</tr>
</tbody>
</table>

The router enters the interdevice configuration mode and names the redundancy scheme that is used between the two routers. The CLIs listed in the preceding example create interdependency between the CUBE redundancy and HSRP.

**Step 2** Enable CUBE and CUBE redundancy.

**Example:**

Enable CUBE on both routers

```
Router(config)#voice service voip
Router(config-voi-serv)#mode border-element
Router(config-voi-serv)#allow-connections sip to sip
```

Enables CUBE on the router and allows connections between the specific type of endpoints in a VoIP network.

**Example:**

Enable the CUBE redundancy and call checkpointing on both routers

```
Router(config)#voice service voip
Router(config-voi-serv)#redundancy
```

**Step 3** Configure Inter-process Communication (IPC) protocol at the HSRP interface.

**Example:**

Active CUBE configuration

```
CUBE-1(config)#ipc zone default
CUBE-1(config-ipzone)#association 1
CUBE-1(config-ipzone-assoc)#no shutdown
CUBE-1(config-ipzone-assoc)#protocol sctp
CUBE-1(config-ipc-protocol-sctp)#local-port 5000
```
Example:
Standby CUBE configuration

CUBE-2(config)#ipc zone default
CUBE-2(config-ipzone)#association 1
CUBE-2(config-ipzone-assoc)#no shutdown
CUBE-2(config-ipzone-assoc)#protocol sctp
CUBE-2(config-ip-protocol-sctp)#local-port 5000
CUBE-2(config-ip-local-sctp)#local-ip 203.0.113.11
CUBE-2(config-ip-local-sctp)#remote-port 5000
CUBE-2(config-ip-remote-sctp)#remote-ip 203.0.113.10

The following table provides details of the CLIs used in the configuration.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipc zone default</td>
<td>Configures the Inter-process Communication Protocol (IPC) and enters IPC zone configuration mode. Use this command to initiate the communication link between the active and standby routers.</td>
</tr>
<tr>
<td>association 1</td>
<td>Configures an association between the two routers and enters the IPC association configuration mode. Under this, configure the details of the association such as the transport protocol, local port, local IP address, remote port, and remote IP address. Valid association IDs range 1–255. There are no default association IDs.</td>
</tr>
<tr>
<td>no shutdown</td>
<td>Restarts a disabled association and the associated transport protocol. For any changes to the transport protocol parameters, you must shut down the association.</td>
</tr>
<tr>
<td>protocol sctp</td>
<td>Configures Stream Control Transmission Protocol (SCTP) as the transport protocol for the association and enables SCTP protocol configuration mode.</td>
</tr>
<tr>
<td>local-port port_num</td>
<td>Defines the local SCTP port number for communication with the redundant peer.</td>
</tr>
<tr>
<td>local-ip ip_addr</td>
<td>Defines the local router's IP address for communication with the redundant peer. The local IP address must match the remote IP address on the redundant router.</td>
</tr>
<tr>
<td>remote-port port_num</td>
<td>Defines the remote SCTP port number for communication with the redundant peer.</td>
</tr>
<tr>
<td>remote-ip ip_addr</td>
<td>Defines the remote IP address for communication with the redundant peer. All remote IP addresses must point to the same router.</td>
</tr>
</tbody>
</table>
Allows the active CUBE to communicate with the standby CUBE about the state of the calls. Configuration must be applied on the LAN side.

**Note**  The local-port and the remote-port must be set to 5000 on the active and standby routers.

### Step 4

(Optional) Configure Virtual Route Forwarding (VRF) on the platform.

**Example:**

VRF configuration on active and standby CUBE

```plaintext
Router(config)#ip vrf LAN-VRF
Router(config)#rd 1:1
Router(config)#ip vrf WAN-VRF
Router(config)#rd 1:1
```

The following table provides details of the CLIs used in the configuration.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip vrf</td>
<td>vrf-name</td>
</tr>
<tr>
<td></td>
<td>Creates a VRF with the specified name.</td>
</tr>
<tr>
<td></td>
<td>Space is not allowed in the VRF name.</td>
</tr>
<tr>
<td>rd</td>
<td>route-distinguisher</td>
</tr>
<tr>
<td></td>
<td>Creates a VRF table by specifying a route distinguisher.</td>
</tr>
<tr>
<td></td>
<td>Enter either an AS number and an arbitrary number (xxx:y) or an IP address and arbitrary number (A.B.C.D:y).</td>
</tr>
</tbody>
</table>

CUBE High Availability with HSRP supports VRF. Traffic interfaces (SIP/RTP) can have VRFs configured. VRF IDs are checkpointed for the calls before and after the switchover. VRF configurations including VRF-based RTP port range, must be identical on both active and standby routers.

### Step 5

Configure HSRP on the interfaces.

a) Configure the inside interface.

**Example:**

Active CUBE configuration

```plaintext
CUBE-1(config)#interface GigabitEthernet0/0
CUBE-1(config-if)#description "Enterprise LAN"
CUBE-1(config-if)#ip vrf forwarding LAN-VRF
CUBE-1(config-if)#ip address 203.0.113.10 255.255.255.0
CUBE-1(config-if)#standby version 2
CUBE-1(config-if)#standby 1 ip 203.0.113.12
CUBE-1(config-if)#standby delay minimum 30 reload 60
CUBE-1(config-if)#standby 1 track 2 decrement 10
CUBE-1(config-if)#standby 1 track 3 decrement 10
CUBE-1(config-if)#standby 1 priority 50
```

**Example:**

Standby CUBE configuration

```plaintext
CUBE-2(config)#interface GigabitEthernet0/0
CUBE-2(config-if)#description "Enterprise LAN"
CUBE-2(config-if)#ip vrf forwarding LAN-VRF
CUBE-2(config-if)#ip address 203.0.113.11 255.255.255.0
CUBE-2(config-if)#standby version 2
```
b) Configure the outside interface.

**Example:**

Active CUBE configuration

```
CUBE-1(config)#interface GigabitEthernet0/1
CUBE-1(config-if)#description "Enterprise WAN"
CUBE-1(config-if)#ip vrf forwarding WAN-VRF
CUBE-1(config-if)#ip address 192.0.2.1 255.255.255.0
CUBE-1(config-if)#standby version 2
CUBE-1(config-if)#standby 10 ip 192.0.2.5
CUBE-1(config-if)#standby delay minimum 30 reload 60
CUBE-1(config-if)#standby 10 preempt
CUBE-1(config-if)#standby 10 track 2 decrement 10
CUBE-1(config-if)#standby 10 track 3 decrement 10
CUBE-1(config-if)#standby 10 priority 50
```

Example:

Standby CUBE configuration

```
CUBE-2(config)#interface GigabitEthernet0/1
CUBE-2(config-if)#description "Enterprise WAN"
CUBE-2(config-if)#ip vrf forwarding WAN-VRF
CUBE-2(config-if)#ip address 192.0.2.2 255.255.255.0
CUBE-2(config-if)#standby version 2
CUBE-2(config-if)#standby 10 ip 192.0.2.5
CUBE-2(config-if)#standby delay minimum 30 reload 60
CUBE-2(config-if)#standby 10 preempt
CUBE-2(config-if)#standby 10 track 2 decrement 10
CUBE-2(config-if)#standby 10 track 3 decrement 10
CUBE-2(config-if)#standby 10 priority 50
```

c) Configure the HSRP interface (between the active and standby CUBE).

**Example:**

Active CUBE configuration

```
CUBE-1(config)#interface GigabitEthernet0/2
CUBE-1(config-if)#description "HSRP Interface"
CUBE-1(config-if)#ip address 198.51.100.1 255.255.255.0
CUBE-1(config-if)#standby version 2
CUBE-1(config-if)#standby 100 ip 198.51.100.5
CUBE-1(config-if)#standby delay minimum 30 reload 60
CUBE-1(config-if)#standby 100 preempt
CUBE-1(config-if)#standby 100 track 2 decrement 10
CUBE-1(config-if)#standby 100 track 3 decrement 10
CUBE-1(config-if)#standby 100 priority 50
```
Example:

Standby CUBE configuration

CUBE-2(config)#interface GigabitEthernet0/2
CUBE-2(config-if)#description "HSRP Interface"
CUBE-2(config-if)#ip address 198.51.100.2 255.255.255.0
CUBE-2(config-if)#standby version 2
CUBE-2(config-if)#standby 100 ip 198.51.100.5
CUBE-2(config-if)#standby delay minimum 30 reload 60
CUBE-2(config-if)#standby 100 preempt
CUBE-2(config-if)#standby 100 name SB
CUBE-2(config-if)#standby 100 track 2 decrement 10
CUBE-2(config-if)#standby 100 track 3 decrement 10
CUBE-2(config-if)#standby 100 priority 50

Note  

`ip vrf forwarding vrf-name` is applicable only if you have configured VRF.

The HSRP interface cannot have VRFs associated with it. For a CUBE deployment that has VRFs configured for SIP/RTP interfaces, you must have minimum of three interfaces. Otherwise, you can use any of the LAN interfaces as an HSRP interface.

The following table provides details of the CLIs used in the configuration.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>interface type number</code></td>
<td>Configures an interface type and enters the interface configuration mode.</td>
</tr>
<tr>
<td><code>ip address ip_address subnet_mask</code></td>
<td>Configures an IP address for an interface.</td>
</tr>
<tr>
<td>`standby version{1</td>
<td>2}`</td>
</tr>
<tr>
<td><code>standby [group-number] ip [ip_address]</code></td>
<td>Activates HSRP.</td>
</tr>
<tr>
<td></td>
<td>• If you do not configure a group number, the default group number is 0. The group number range is 0–255 for HSRP version 1 and 0–4095 for HSRP version 2.</td>
</tr>
<tr>
<td></td>
<td>• The value for the <code>ip_address</code> argument is the virtual IP address of the virtual device. For HSRP to elect a designated device, you must configure the virtual IP address for at least one of the devices in the group; it can be learned on the other devices in the group.</td>
</tr>
</tbody>
</table>
### Option Description

**standby delay minimum** `min-seconds reload reload-seconds`  
Configures the delay period before the initialization of HSRP group.  
- The `min-seconds` value is the minimum time (in seconds) to delay the HSRP group initialization after an interface comes up. This minimum delay period applies to all subsequent interface events.  
- The `reload-seconds` value is the time period to delay after the device has reloaded. This delay period applies only to the first interface-up event after the device has reloaded.  

**Note**  
The recommended `min-seconds` value is 30 and the recommended `reload-seconds` value is 60.

**standby group-number preempt**  
Allows the router to become the active router when the priority is higher than all other HSRP-configured routers in the HSRP group. If you do not use the `standby preempt` command in the configuration for a router, that router does not become the active router, even if the priority is higher than all other routers.

**standby group-number track track-process-number decrement value**  
Configures HSRP to track a device and change the HSRP priority on the basis of the state of the device. Decrement value specifies the value by which the HSRP priority of the tracked device is decremented (or incremented) when the device goes down (or becomes available).

**standby x priority**  
Defines the Hot Standby priority that is used in choosing the active router. The range is 1–255, where 1 denotes the lowest priority and 255 the highest priority.  

**Note**  
In cases where the standby priority is the same, the device with the higher IP address assumes the role of the active router.

**ip vrf forwarding vrf-name**  
Associates the specified VRF with the interface.

### Step 6  
Configure Interface Tracking.

#### Example:

**Active and standby CUBE configuration**

```
Router(config)#track 1 interface Gig0/0 line-protocol
Router(config)#track 2 interface Gig0/1 line-protocol
Router(config)#track 3 interface Gig0/2 line-protocol
```

Create a tracking list to track the line-protocol state of an interface.

The following table provides details of the CLIs used in the configuration.
Step 7

Bind traffic to the respective interfaces.

a) Bind traffic that is destined to the outside (Service Provider (SP) SIP trunk) to the outside physical interface.

Example:

Active and standby CUBE configuration

Router(config)#dial-peer voice 100 voip
Router(config-dial-peer)#description TO SERVICE PROVIDER
Router(config-dial-peer)#destination-pattern 9T
Router(config-dial-peer)#session protocol sipv2
Router(config-dial-peer)#session target ipv4:y.y.y.y
Router(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/1
Router(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/1

b) Bind traffic that is destined to the inside (Unified CM or IP PBX) to the inside physical interface.

Example:

Active and standby CUBE configuration

CUBE(config)#dial-peer voice 200 voip
CUBE(config-dial-peer)#description TO CUCM
CUBE(config-dial-peer)#destination-pattern 555...
CUBE(config-dial-peer)#session protocol sipv2
CUBE(config-dial-peer)#session target ipv4:203.0.113.1
CUBE(config-dial-peer)#voice-class sip bind control source-interface GigabitEthernet0/0
CUBE(config-dial-peer)#voice-class sip bind media source-interface GigabitEthernet0/0

Binding the traffic to the respective interfaces ensures that all RTP and SIP packets are created with the virtual IP associated with the respective physical interface.

The following table provides details of the CLIs used in the configuration.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>track object-number interface interface-id line-protocol</td>
<td>Enters tracking configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• The object-number identifies the tracked object and the range is 1–500.</td>
</tr>
<tr>
<td></td>
<td>• The interface-id represents the interface that is tracked.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dial-peer voice number voip</td>
<td>Defines a local dial peer.</td>
</tr>
<tr>
<td></td>
<td>• The number argument identifies the dial peer. Valid entries are 1–2147483647.</td>
</tr>
<tr>
<td>description string</td>
<td>Provides a description for the dial-peer group.</td>
</tr>
<tr>
<td>destination-pattern string</td>
<td>Defines the phone number that identifies the destination pattern that is associated with the dial-peer.</td>
</tr>
</tbody>
</table>
### Option Description

<table>
<thead>
<tr>
<th>session-protocol sipv2</th>
<th>Configures SIP as the session protocol type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>session target ip-address</td>
<td>Configures the network address of the remote router to which you want to send a call once a local voice-network dial peer is matched.</td>
</tr>
<tr>
<td>voice-class sip bind control source-interface interface-id</td>
<td>Sets a source interface for signaling and media packets. The binding applies to the specified interfaces only.</td>
</tr>
<tr>
<td>voice-class sip bind media source-interface interface-id</td>
<td>将一个源接口用于信号和媒体包。该绑定仅适用于指定的接口。</td>
</tr>
<tr>
<td>• control</td>
<td>Binds signaling packets.</td>
</tr>
<tr>
<td>• binds</td>
<td>Binds media packets.</td>
</tr>
<tr>
<td>• source-interface interface-id</td>
<td>Type of interface and its ID.</td>
</tr>
</tbody>
</table>

### Step 8
Configure Media Inactivity feature.

**Example:**
Active and standby CUBE configuration

```
CUBE(config)#ip rtcp report interval 3000
CUBE(config)#gateway
CUBE(config-gateway)#media-inactivity-criteria all
timer receive-rtcp 5
timer receive-rtp 86400
```

The following table provides details of the CLIs used in the configuration.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip rtcp report interval time in milliseconds</td>
<td>Configures the average reporting interval between subsequent RTCP report transmissions.</td>
</tr>
<tr>
<td>gateway</td>
<td>Enters the gateway configuration mode.</td>
</tr>
<tr>
<td>media-inactivity-criteria all</td>
<td>Specifies the use of both RTCP and RTP for detecting the silence on a voice call.</td>
</tr>
<tr>
<td>timer receive-rtcp timer</td>
<td>Enable the Real-Time Control Protocol (RTCP) timer and configures a multiplication factor for the RTCP timer interval for Session Initiation Protocol (SIP) or H.323.</td>
</tr>
<tr>
<td>• timer</td>
<td>Multiples of the RTCP report transmission interval. Range is 0–1000. Default value is 0. Recommended value is 5.</td>
</tr>
</tbody>
</table>

The Media Inactivity Timer enables the active/standby router pair to monitor and disconnect calls, if the router pair does not receive Real-Time Protocol (RTP) packets within a configurable time period.

When the active or the standby router does not receive RTP packets for a call, the SIP Media Inactivity Timer releases the session. The Media Inactivity Timer guards against any hung sessions resulting from the failover when a normal call disconnect does not clear the call.
You must configure the same duration for the Media Inactivity Timer on both routers.

**Step 9**
Reload the routers.

After completing all the preceding configuration steps, save and reload both the active and standby router.

**Step 10**
Point the attached devices to the CUBE HSRP Virtual IP (VIP) address.

The IP-PBX, Cisco Unified SIP Proxy, or service provider must route the calls to CUBE’s virtual IP address. This HA configuration does not handle SIP/H.323 messages to CUBE’s physical IP addresses.

---

**Configuration Examples**

**Example Configuration for Dual-Attached CUBE HSRP Redundancy**

This section provides sample configurations for both the active and standby CUBE routers. In these configurations, the HSRP Hello and Hold timers use their default values of 3 and 8 seconds respectively, and are not shown explicitly in the CLI output.

**Active Router Configuration**

```plaintext
ipc zone default
association 1
  no shutdown
  protocol sctp
    local-port 5000
    local-ip 203.0.113.10
    remote-port 5000
    remote-ip 203.0.113.11

voice service voip
  mode border-element
  allow-connections sip to sip
  redundancy

redundancy inter-device
  scheme standby SB

redundancy

interface GigabitEthernet0/0
  ip address 203.0.113.10 255.255.255.0
  standby version 2
  standby 1 ip 203.0.113.12
  standby delay minimum 30 reload 60
  standby 1 preempt
  standby 1 track 2 decrement 10
  standby 1 track 3 decrement 10
  standby 1 priority 50

interface GigabitEthernet0/1
  ip address 192.0.2.1 255.255.255.0
  standby version 2
  standby 10 ip 192.0.2.5
  standby delay minimum 30 reload 60
  standby 10 preempt
```
standby 10 track 2 decrement 10
standby 10 track 3 decrement 10
standby 10 priority 50

! interface GigabitEthernet0/2
  ip address 198.51.100.1 255.255.255.0
  standby version 2
  standby 10 ip 198.51.100.5
  standby delay minimum 30 reload 60
  standby 100 preempt
  standby 100 name SB
  standby 100 track 2 decrement 10
  standby 100 track 3 decrement 10
  standby 100 priority 50

!
track 1 interface Gig0/0 line-protocol
track 2 interface Gig0/1 line-protocol
track 3 interface Gig0/2 line-protocol
!
dial-peer voice 100 voip
  description TO SERVICE PROVIDER
  destination-pattern 9T
  session protocol sipv2
  session target ipv4:y.y.y.y
  voice-class sip bind control source-interface GigabitEthernet0/1
  voice-class sip bind media source-interface GigabitEthernet0/1
!
dial-peer voice 200 voip
  description TO CUCM
  destination-pattern 555....
  session protocol sipv2
  session target ipv4:203.0.113.1
  voice-class sip bind control source-interface GigabitEthernet0/0
  voice-class sip bind media source-interface GigabitEthernet0/0
!
ip rtcp report interval 3000
!
gateway
  media-inactivity-criteria all
  timer receive-rtcp 5
  timer receive-rtp 86400

Standby Router Configuration

ipc zone default
  association 1
    no shutdown
    protocol sctp
      local-port 5000
      local-ip 203.0.113.11
      remote-port 5000
      remote-ip 203.0.113.10
!
voice service voip
  mode border-element
  allow-connections sip to sip
  redundancy
!
redundancy inter-device
  scheme standby SB
!
redundancy
!interface GigabitEthernet0/0
 ip address 203.0.113.11 255.255.255.0
 standby version 2
 standby 1 ip 203.0.113.12
 standby delay minimum 30 reload 60
 standby 1 preempt
 standby 1 track 2 decrement 10
 standby 1 track 3 decrement 10
 standby 1 priority 50

! interface GigabitEthernet0/1
 ip address 192.0.2.2 255.255.255.0
 standby version 2
 standby 10 ip 192.0.2.5
 standby delay minimum 30 reload 60
 standby 10 preempt
 standby 10 track 2 decrement 10
 standby 10 track 3 decrement 10
 standby 10 priority 50

! interface GigabitEthernet0/2
 ip address 198.51.100.2 255.255.255.0
 standby version 2
 standby 100 ip 198.51.100.5
 standby delay minimum 30 reload 60
 standby 100 preempt
 standby 100 name SB
 standby 100 track 2 decrement 10
 standby 100 track 3 decrement 10
 standby 100 priority 50

! track 1 interface Gig0/0 line-protocol
 track 2 interface Gig0/1 line-protocol
 track 3 interface Gig0/2 line-protocol

! dial-peer voice 100 voip
 description TO SERVICE PROVIDER
 destination-pattern 9T
 session protocol sipv2
 session target ipv4:y.y.y.y
 voice-class sip bind control source-interface GigabitEthernet0/1
 voice-class sip bind media source-interface GigabitEthernet0/1

! dial-peer voice 200 voip
 description TO CUCM
 destination-pattern 555....
 session protocol sipv2
 session target ipv4:203.0.113.1
 voice-class sip bind control source-interface GigabitEthernet0/0
 voice-class sip bind media source-interface GigabitEthernet0/0

! ip rtcp report interval 3000
!
gateway
 media-inactivity-criteria all
 timer receive-rtcp 5
 timer receive-rtp 86400
Example Configuration for Single-Attached CUBE HSRP Redundancy

Although a dual-attached CUBE is the most common configuration, especially for SP SIP trunk connections, it is also possible to configure CUBE HSRP box-to-box redundancy with a single-attached CUBE deployment. The sample configurations for both the active and standby CUBE routers are as follows:

**Active Router Configuration**

```
ipc zone default
   association 1
      no shutdown
   protocol sctp
      local-port 5000
      local-ip 203.0.113.10
      remote-port 5000
      remote-ip 203.0.113.11
!
voice service voip
   mode border-element
   allow-connections sip to sip
   redundancy
      sip
         bind control source-interface GigabitEthernet0/0
         bind media source-interface GigabitEthernet0/0
!
redundancy inter-device
   scheme standby SB
!
redundancy
!
interface GigabitEthernet0/0
   ip address 203.0.113.10 255.255.0.0
   duplex auto
   speed auto
   keepalive
   standby delay minimum 30 reload 60
   standby version 2
   standby 0 ip 203.0.113.12
   standby 0 preempt
   standby 0 priority 50
   standby 0 name SB
   standby 0 track 1 decrement 10
!
ip rtcp report interval 3000
!
dial-peer voice 5 voip
   description to-SIP-application
   destination-pattern 9T
   session protocol sipv2
   session target ipv4:x.x.x.x
!
dial-peer voice 9 voip
   description to-CUCM
   destination-pattern 555....
   session protocol sipv2
   session target ipv4:y.y.y.y
!
gateway
   media-inactivity-criteria all
   timer receive-rtcp 5
   timer receive-rtp 86400
```
Standby Router Configuration

ipc zone default
    association 1
    no shutdown
    protocol sctp
    local-port 5000
    local-ip 203.0.113.11
    remote-port 5000
    remote-ip 203.0.113.10

voice service voip
    mode border-element
    allow-connections sip to sip
    redundancy
    sip
    bind control source-interface GigabitEthernet0/0
    bind media source-interface GigabitEthernet0/0

redundancy inter-device
    scheme standby SB

redundancy

interface GigabitEthernet0/0
    ip address 203.0.113.11 255.255.0.0
    duplex auto
    speed auto
    standby delay minimum 30 reload 60
    standby version 2
    standby 0 ip 203.0.113.12
    standby 0 priority 50
    standby 0 preempt
    standby 0 name SB
    standby 0 track 1 decrement 10

ip rtcp report interval 3000

dial-peer voice 5 voip
    description to-SIP-application
    destination-pattern 9T
    session protocol sipv2
    session target ipv4:x.x.x.x

dial-peer voice 9 voip
    description to-CUCM
    destination-pattern 555....
    session protocol sipv2
    session target ipv4:y.y.y.y

gateway
    media-inactivity-criteria all
    timer receive-rtcp 5
    timer receive-rtp 86400
Verify Your Configurations

Verify SIP IP Address Bindings

Use the `show sip-ua status` command to verify the SIP binding status.

```
Router#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
Snapshot of SIP listen sockets : 2
Local Address     Listen Port     Secure Listen Port
                   ===========     ===============
192.0.2.1          5060            5061
192.0.2.1          5060            5061
SIP early-media for 180 responses with SDP: ENABLED
SIP max-forwards : 70
```

Verify Current CPU Use

Use the `show process cpu history` command to verify the CPU utilization percentage at regular intervals.

Check CPU utilization before performing a switchover and proceed with a forced failover only when the CPU utilization is less than 70%. The `show process cpu sorted` command can also be used repeatedly to understand the CPU utilization for a particular process.

Verify the Call Processing During a Switchover

Use the `show sip-ua statistics` command to verify the call drops during the switchover by checking the number of BYE messages. Calls in progress during the switchover are dropped. Only established calls are preserved.

Use the `show interface accounting` command to verify the media path confirmation during a switchover.

```
Router#show interfaces g0/0 accounting
```

```
GigabitEthernet0/0
Protocol Pkts In Chars In Pkts Out Chars Out
Other    1     58    6     360
IP       406   178841 201   16394
ARP      569   34292  0     0
CDP      116   31672  22    7304
```

**Note**

Check IP Pkts In and Pkts Out counters. These counters must be increasing at reasonable rate. For example, if you are using G.711 20ms packetization and no VAD, you must see the packet counters increase by around 50 every second.
**Force a Manual Failover for Testing**

Box-to-box redundancy using HSRP supports the stateful switchover of calls which means both media (RTP) and call signaling are preserved. Therefore, during the switchover, only calls in the active state (media path in "sendrecv" connection mode) are preserved while calls in the transient state (non-active state, media path not in "sendrecv" connection mode) are not.

You can expect that switchovers occurring in real environments, where there is a constant mixture of calls in transient (call setup or being modified) and established state, result in some dropped calls during a failover. You can estimate the number of dropped calls by using the following formula: $(0.3 + \text{HSRP hold-timer}) \times \text{CPS}$.

To check that your configuration is correct, you can force a manual switchover.

You can achieve manual switchovers in various ways:

- Initiate the manual switchover by using the `redundancy switch-activity force` command on the active router.
- Reload of the active router
- Hard restart of the active router
- Pull out the HSRP interface or power cable of the active router.
- Shut down the HSRP interface of the active router.
- Change in any parameter of the HSRP interface of the active/standby router without shutting down the association under IPC mode leads to a router reload. Therefore, you must shut down the interface before you make any changes, unless you are using this as a trigger to force a switchover.

The `show voip rtp connections` command shows the number of active connections on both the active and standby routers after a switchover.

The `show call active voice brief` command does not show any output on the standby router after a switchover because the signaling information is not checkpointed.

**Before you begin**

Before you start a manual switchover, take note of the following:

- Monitor the CPU utilization % on the active and standby pair. The active router has a higher CPU utilization as it is actively handling the calls, while the standby router shows 0 CPU utilization as it is idle until a switchover occurs.
- Ensure that you perform a manual switchover when the CPU utilization of the active router is no more than 70%. All switchovers lead to a spike in CPU utilization.
- Use the `show voip rtp connection` and `show voice high-availability summary` commands to make sure that the existing calls across the active and standby router pair are in sync.

Perform the following steps to configure and verify a single switch over:

**SUMMARY STEPS**

1. Configure HSRP Box-to-box redundancy as explained in the Configuration section.
2. Reload and keep both routers in rommon.
3. Boot up one router. After the router comes up, execute the `show redundancy state` command and make sure it displays `my state` as active and peer state as Disabled. This can take a while after boot up.

4. Boot up the second router. After the router comes up, execute the `show redundancy state` command and make sure it displays `my state` as standby-Hot and `peer state` as active.

5. Start one or more calls across the system. Execute the `show voice high-availability summary` and `show voip rtp connection` commands on both the active and standby routers to make sure that the calls are up and checkpointed.

6. Test switchover by reloading the active router. If you are using a phone to make calls, you can listen to the phone to make sure that the media path is preserved. If you are using test equipment, you can use the packet displays to determine if media for the calls are flowing.

7. Test Media Inactivity: Stop the call. Repeat `show voip rtp connection`. After the media-inactivity timer expiry, there must be no more active RTP connections. You can also check this using the `show voice high-availability summary` command.

**DETAILED STEPS**

**Step 1** Configure HSRP Box-to-box redundancy as explained in the Configuration section.

**Step 2** Reload and keep both routers in rommon.

**Step 3** Boot up one router. After the router comes up, execute the `show redundancy state` command and make sure it displays `my state` as active and peer state as Disabled. This can take a while after boot up.

**Example:**

```
Router#show redundancy states

my state = 13 -ACTIVE
peer state = 1 -DISABLED
```

**Step 4** Boot up the second router. After the router comes up, execute the `show redundancy state` command and make sure it displays `my state` as standby-Hot and `peer state` as active.

**Example:**

```
Router#show redundancy states

my state = 8 -STANDBY HOT
peer state = 13 -ACTIVE
```

**Step 5** Start one or more calls across the system. Execute the `show voice high-availability summary` and `show voip rtp connection` commands on both the active and standby routers to make sure that the calls are up and checkpointed.

**Step 6** Test switchover by reloading the active router. If you are using a phone to make calls, you can listen to the phone to make sure that the media path is preserved. If you are using test equipment, you can use the packet displays to determine if media for the calls are flowing.

**Example:**

```
Router#show interfaces g0/0 accounting

GigabitEthernet0/0
Protocol Pkts In Chars In Pkts Out Chars Out
Other 1 58 6 360
IP 406 178841 201 16394
ARP 569 34292 0 0
CDP 116 31672 22 7304
```
Step 7  Test Media Inactivity: Stop the call. Repeat `show voip rtp connection`. After the media-inactivity timer expiry, there must be no more active RTP connections. You can also check this using the `show voice high-availability summary` command.

Example:

Router#show voice high-availability summary | include media

Media Inactivity event count: 1

Troubleshoot High Availability Issues

Use the following show and debug commands to troubleshoot any issues:

- `show redundancy state`
- `show redundancy inter-device`
- `show standby brief`
- `show standby internal`
- `show sip-ua status`
- `show sip-ua statistics`
- `show voice high-availability summary`
- `show voip rtp connection | include connection`
- `show arp`
- `debug voip ccapi all`
- `debug voip ccapi error`
- `debug voip rtp session`
- `debug voip rtp session`
- `debug voip rtp error`
- `debug voip rtcp error`
- `debug voice high-availability all`
- `debug voice high-availability error`
- `debug ccsip info`
- `debug ccsip messages`
- `debug ccsip media`
- `debug ccsip error`
- `debug standby terse`
Note
Do not turn on a large number of debugs on a system carrying high volume of active call traffic.

Note
On every switchover, after router reload, you must re-enable the debugs on the new standby router.

Each router in an HSRP group participates in the protocol by implementing a simple state machine. All routers begin in the Initial state.

The following table illustrates the different router states.

<table>
<thead>
<tr>
<th>States</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>This is the starting state and indicates that HSRP is not running. This state is entered through configuration change or when an Interface first comes up.</td>
</tr>
<tr>
<td>Learn</td>
<td>The router has not determined the virtual IP address, and not yet seen an authenticated Hello message from the active router. In this state, the router is still waiting to hear from the active router.</td>
</tr>
<tr>
<td>Listen</td>
<td>The router knows the virtual IP address, but is not the active or standby router. It listens for Hello messages from those routers.</td>
</tr>
<tr>
<td>Speak</td>
<td>The router sends periodic Hello messages and is actively participating in the election of the active and standby router. A router cannot enter the Speak state unless it has the virtual IP address.</td>
</tr>
<tr>
<td>Standby</td>
<td>The router is a candidate to become the next active router and sends periodic Hello messages. Excluding transient conditions, there MUST be at most one router in the group in Standby state.</td>
</tr>
<tr>
<td>Active</td>
<td>The router is currently forwarding packets that are sent to the group's virtual MAC/IP address. The router sends periodic Hello messages. Besides transient conditions, there MUST be at most one router in Active state in the group.</td>
</tr>
</tbody>
</table>

Troubleshooting Tip - Why Are There Two Active Routers?
This scenario occurs when both routers fail to see the HSRP Hellos from each other.

- Check if each router can ping the other's IP interface address. If not, then communication between the routers is down.
• Use the **debug standby** command to see if the routers are sending and receiving HSRP Hello packets. If the peer is sending Hellos, but they are not being received then check **show interface** or **show controller** commands to see if the interface is listening to the HSRP multicast address.
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 811
- Prerequisites for DSP High Availability, on page 811
- Features Supported with DSP High Availability, on page 812
- Restrictions for DSP High Availability, on page 812
- Troubleshooting DSP HA Support on CUBE, on page 812
- Configuration Examples for DSP HA, on page 813

### 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>

### 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)
  • Cisco Catalyst 8200 Edge series
  • Cisco Catalyst 8300 Edge series

• 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

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 example shows 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 13 39.0.0 UP 1 USED xcode 1 16558

Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5
813
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 13 39.0.0 UP 1 USED xcode 1 16558
0 0
0 13 39.0.0 UP 1 USED xcode 1 16559
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 815
- Prerequisites for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 816
- Restrictions for Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 817
- Information About Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices, on page 817

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 (Call Escalation and De-escalation with Stateful Switchover)</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>
<tr>
<td>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices (Media Forking with High Availability)</td>
<td>Cisco IOS XE Release 3.8S 15.3(1)T</td>
<td>Provides support for media forking with high availability mechanism.</td>
</tr>
<tr>
<td>High Availability Protected Mode and Box-to-Box HA Support</td>
<td>Cisco IOS XE Release 3.11S</td>
<td>Provides support for enabling the PROTECTED mode on a Voice HA-enabled ASR.</td>
</tr>
<tr>
<td>OPTIONS PING Support under HA Configuration</td>
<td>15.4(3)M</td>
<td>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 CUBE's. 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. For configuration examples, see the Examples section about configuring interfaces (ISR and ASR) and configuring SIP binding.</td>
</tr>
<tr>
<td>Support for REFER and BYE/Also after Software Switch-Over</td>
<td>Cisco IOS 15.5(2)T Cisco IOS XE Release 3.15S</td>
<td>REFER based supplementary services with high availability is supported on Cisco Unified Border Element (Cisco UBE) after stateful switchover. Support is also provided for SIP-to-SIP BYE/Also calls.</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.

**Note**

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 84: 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 theCisco 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:0x88 video tos:0x0
IP 9.45.25.33:58300 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:0x88 video tos:0x0
IP 9.45.25.33:58400 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:23732/1039376 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:23732/1039376 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:**
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:**
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 P708090 9.42.30.189:26300
4426 ORG T27 g711ulaw VOIP P9876 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:**
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
```
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] :
[ (ACTIVE (13) / PROTECTED (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 STANDBY HOT (8)
Peer current RF state ACTIVE (13)

Current VOIP HA state [LOCAL / PEER] :
[ PROTECTED (7) / ACTIVE (13) ]

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 call 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 ccap all
• debug voip ccap service
• debug voice high-availability all
• debug voip rtp error
• debug voip rtp inout
• debug voip rtp high-availability
• debug voip rtp function
• debug ccsip all

Use the following commands to troubleshoot media forking support on high availability:

• debug ccsip all
• debug voip high-availability all
• debug voip ccap 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 ccap inout
• debug cch323 session
• debug cch323 function
• debug cch323 error
• debug ccsip all

Use the following debug commands to troubleshoot issues related to handling of REFER based supplementary services:

• debug ccsip verbose
• debug voip application all
• debug voip ccap 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
redudancy rii 1
redudancy 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
Example: Configuring SIP Binding
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 829
- Prerequisites, on page 830
- Restrictions, on page 830
- Recommendations, on page 830
- CVP Survivability TCL support with High Availability, on page 830
- Configuring CVP Survivability TCL support with High Availability, on page 830

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 82: Feature Information for CVP Survivability TCL support with High Availability

<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</td>
<td>This feature enables CUBE support call survivability features in CUBE high availability mode for all active calls handled by CVP.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE Denali 16.3.1</td>
<td></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 check pointed. (Calls which are connected - 200OK / ACK transaction completed). Calls in transition state will not be check pointed.

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 833
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 833
- Restrictions for ICE-lite Support on CUBE, on page 834
- Information About ICE-Lite Support on CUBE, on page 834
- How to Configure ICE-Lite Support on CUBE, on page 836
- Additional References, on page 845

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 83: Feature Information for ICE-Lite Support on CUBE

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| ICE-Lite Support on CUBE | Cisco IOS 15.5(3)M  
Cisco IOS XE 3.16S  | 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: debug voip icelib, show voip ice global-stats, show voip ice instance call-id call-id, show voip ice summary, and stun usage ice |

### 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.
• Support for ICE-lite in the contact header with a media-tag option of REGISTER message (as per RFC 5768).

• ICE-lite feature is in compliance with section 4.2 of RFC 7584, with CUBE acting as ICE termination Back-to-Back UA.

• CUBE accepts Full ICE Offer and responds in ICE-lite mode.

• CUBE responds to mid call updates or early dialog updates with changes to SDP parameters, and which requires ICE to restart.

• For outbound offer from CUBE, a Session Description Protocol (SDP) with ICE-lite semantics is sent.

• ICE protocol verifies all types of media streams (audio, video, application media lines) and components (RTP, RTCP), wherever applicable.

**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>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: 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: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>voice class stun-usage tag</td>
<td>Sets STUN usage global parameters, and enters voice class configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# voice class stun-usage 5</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>stun usage ice lite</td>
<td>Configures ICE in ICE-Lite mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-class)# stun usage ice lite</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config-class)# end</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
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 4
25 ANS T189 H264 VOIP-VIDEO P8181 72.163.212.137:2328
30 ORG T189 H264 VOIP-VIDEO P9191 9.45.46.16:8028
35 ANS T189 H264 VOIP-VIDEO P8181 9.45.46.16:8008
36 ORG T189 H264 VOIP-VIDEO P9191 72.163.212.163:2328
```

**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:

<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>20</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>25</td>
<td>30</td>
<td>8000</td>
<td>2326</td>
<td>10.104.45.107</td>
<td>72.163.212.137</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>31</td>
<td>8002</td>
<td>2328</td>
<td>10.104.45.107</td>
<td>72.163.212.137</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>32</td>
<td>8036</td>
<td>2454</td>
<td>10.104.45.107</td>
<td>72.163.212.137</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>33</td>
<td>8004</td>
<td>2330</td>
<td>10.104.45.107</td>
<td>72.163.212.137</td>
</tr>
<tr>
<td>5</td>
<td>29</td>
<td>34</td>
<td>8038</td>
<td>2392</td>
<td>10.104.45.107</td>
<td>72.163.212.137</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>25</td>
<td>8006</td>
<td>8016</td>
<td>9.45.46.16</td>
<td>9.45.46.16</td>
</tr>
<tr>
<td>7</td>
<td>31</td>
<td>26</td>
<td>8008</td>
<td>8028</td>
<td>9.45.46.16</td>
<td>9.45.46.16</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>27</td>
<td>8010</td>
<td>8030</td>
<td>9.45.46.16</td>
<td>9.45.46.16</td>
</tr>
<tr>
<td>9</td>
<td>33</td>
<td>28</td>
<td>8012</td>
<td>8032</td>
<td>9.45.46.16</td>
<td>9.45.46.16</td>
</tr>
<tr>
<td>10</td>
<td>34</td>
<td>29</td>
<td>8014</td>
<td>8034</td>
<td>9.45.46.16</td>
<td>9.45.46.16</td>
</tr>
<tr>
<td>11</td>
<td>35</td>
<td>36</td>
<td>8016</td>
<td>8006</td>
<td>9.45.46.16</td>
<td>9.45.46.16</td>
</tr>
</tbody>
</table>
```
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
---------
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
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107 port 8036 type host RemoteIP 72.163.212.137 port 2332 type host
---------
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
---------
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

CALL-ID | ICE-STATE
---------|-----------------------
  25     | COMPLETED
   30    | RUNNING
   35    | RUNNING
   36    | COMPLETED
```

**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

<callID> A/O FAX T<sec> Codec    type    Peer Address      IP R<ip>:<udp>
Total call-legs: 2
57 ANS  T4   g711ulaw    VOIP    Padithyam  173.39.64.79:7078
58 ORG  T4   g711ulaw    VOIP    P9191       72.163.212.163:2336
```

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</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>
</tr>
</tbody>
</table>

VoIP RTP active connections:

<table>
<thead>
<tr>
<th>No. CallId</th>
<th>dstCallLocalIP</th>
<th>LocalRTP</th>
<th>RmtRTP</th>
<th>MPSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>57</td>
<td>58</td>
<td>8040</td>
<td>7078</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 : 01CD61B24C07331EDC27A5B
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###
```

ICE-Lite Support on CUBE
ICE-Lite on CUBE (Error Flow Calls)
STUN Message Length = 44
004030: *Aug 8 14:25:30.876 IST: //-/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsgIntegrity: Successfully
Encoded MI attribute. Exit.
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
004039: *Aug 8 14:25:30.876 IST: //-/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
User Name : Not Set/Present
Mapped Address : Not Set/Present
Error Code : Number = 400 ,Reason = Bad Request
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 : D0E2E828944BF3D07CC5C06D026D8909B85EF3E9
004040: *Aug 8 14:25:30.876 IST: Finger Print : Not Set/Present
###STUN Message structure End###
the transaction
004042: *Aug 8 14:25:30.876 IST: //57/91300134802E/STUN/Detail/cisco_stun_process_send_msg_event:
STUN message Sent

Sys Log for sending ICE-controlled parameter instead of ICE-controlling parameter:

004132: *Aug 8 14:25:30.912 IST: #CISCO_STUN-4-ICE_ROLE_CONFLICT: Ice Role Conflict detected in the
received STUN message on UDP IP address 10.104.45.107 port 8042
reason:Role Conflict, code:487
with success
004137: *Aug 8 14:25:30.912 IST: //-/xxxxxxxxxxxx/STUN/Detail/stun_post_bind_request_ind_to_app:
Post Message to Application
004138: *Aug 8 14:25:30.912 IST: //-/xxxxxxxxxxxx/STUN/Detail/cisco_stun_process_stun_pak_rcvd_event:
Received New STUN message###STUN Message structure start###
Message Type : STUN_MSG_TYPE_BINDING_REQ
Message Length : 80
Magic Cookie : 2112A442
Transaction ID : F1CF84958CE76D15C83059D9

Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5
ICE-Lite Support on CUBE

**ICE-Lite on CUBE (Error Flow Calls)**

- **Mapped Address**: Not Set/Present
- **User Name**: GAah:4wWY
- **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**: 186270975
- **ICE Controlled**: 1192003560547232620
- **ICE Controlling**: Not Set/Present
- **Cisco-flowdata**: cisco-flowdata is not present
- **Message Integrity**: 0AF4B8C2378CB90AB0B0A3806507D766BF5CD1DD
- **Finger Print**: 4235512547
- **STUN Message structure End###

```plaintext
###STUN Message structure End###
```
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`

Additional References

### Standards and RFCs

<table>
<thead>
<tr>
<th>Standard/RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 5389</td>
<td><code>Session Traversal Utilities for NAT (STUN)</code></td>
</tr>
<tr>
<td>RFC 5245</td>
<td><code>Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols</code></td>
</tr>
<tr>
<td>Standard/RFC</td>
<td>Title</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>RFC 5766</td>
<td><em>Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)</em></td>
</tr>
<tr>
<td>RFC 5768</td>
<td><em>Indicating Support for Interactive Connectivity Establishment (ICE) in the Session Initiation Protocol (SIP)</em></td>
</tr>
<tr>
<td>RFC 3840</td>
<td><em>Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)</em></td>
</tr>
<tr>
<td>RFC 7584</td>
<td><em>Session Traversal Utilities for NAT (STUN) Message Handling for SIP Back-to-Back User Agents (B2BUAs)</em></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. 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. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
</tbody>
</table>
PART XVI

SIP Protocol Handling

• Mid-call Signaling Consumption, on page 849
• Early Dialog UPDATE Block, on page 859
• Consumption of Forked 18x Responses with SDP During Early Dialog, on page 865
• Support for Pass-Through of Unsupported Content Types in SIP INFO Messages, on page 871
• Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element, on page 873
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, on page 849
• Prerequisites, on page 850
• Mid-call Signaling Passthrough - Media Change, on page 850
• Mid-call Signaling Block, on page 854
• Mid Call Codec Preservation, on page 857

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.
### Table 84: 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. The following commands were introduced or modified: midcall-signaling.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.6S</td>
<td></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 disable codec negotiation in the middle of a call and preserves the codec negotiated before the call. The following commands were introduced or modified: midcall-signaling preserve-codec, voice-class sip midcall-signaling preserve-codec.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.9S</td>
<td></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>

### Prerequisites

- Enable CUBE application on a device

### 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.
- Re-invites with media direction changes are consumed.
- Video transcoding is not supported.
- 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 and it includes session timer negotiation for initial INVITE/200 OK transaction as well.
- 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.
- In the SRTP-RTP scenario, re-invites are not consumed.

**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.
- The following 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 transfreee 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 behavior is for INVITE with REPLACES Header consume scenario:
  - CUBE consumes INVITE with REPLACES Header only when the handle-replaces CLI is configured (under sip-ua or voice-class tenant). In this case, CUBE consumes the INVITE and handles it locally. It triggers an outbound INVITE without replaces header and call gets connected with agent.
  - If the handle-replaces CLI is enabled, the 'transfer-to' party must have the same codec that is used for the original call setup. If there is a different codec offer, CUBE rejects the INVITE with 488 error.
  - If the handle-replaces CLI is not configured, CUBE does not consume the INVITE with REPLACES Header and the outgoing INVITE holds same replace header which CUBE is received.
  - INVITE with REPLACES Header consumption does not support the following configurations:
    - Delayed Offer INVITE
    - Codec, DTMF attribute changes, and RSVP
    - Mid-call Signaling block
    - IPv6

- 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 supports 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>
</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 midcall-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><code>enable</code></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>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>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Re-Invites are passed through only when bidirectional media is added.</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><code>midcall-signaling passthru media-change</code></td>
<td></td>
</tr>
<tr>
<td>• In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td><code>voice-class sip midcall-signaling passthru media-change</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>In Global VoIP SIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</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-voi-serv)# midcall-signaling passthru media-change</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

**In Dial-peer configuration mode:**

Device(config)# dial-peer voice 2 voip
Device(config-dial-peer)# voice-class sip midcall-signaling passthru media-change

**Step 4**

| end | Exits to privileged EXEC mode. |

---

**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 midcall-signaling passthru media-change
  incoming called-number 1111111111
  voice-class codec 2 offer-all

dial-peer voice 400 voip
  destination-pattern 1111111111
  session protocol sipv2
  session target ipv4:9.45.38.39:9000
  incoming called-number 2222222222
  voice-class codec 1 offer-all
```

**Example Configuring Passthrough SIP Messages at the Global Level**

The following example shows how to passthrough 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 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.
• 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.
• 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.
• In the SRTP-RTP scenario, re-invites are not consumed.

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 midcall-signaling 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><strong>Example:</strong> Device&gt; enable</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>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></td>
<td>Configure blocking of mid-call signaling changes:</td>
</tr>
<tr>
<td></td>
<td>Mid-call signaling is always blocked.</td>
</tr>
<tr>
<td></td>
<td>• In Global VoIP SIP configuration mode</td>
</tr>
<tr>
<td></td>
<td>midcall-signaling block</td>
</tr>
<tr>
<td></td>
<td>• In dial-peer configuration mode</td>
</tr>
<tr>
<td></td>
<td>voice-class sip midcall-signaling block</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>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 block</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 2 voip</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip</td>
<td></td>
</tr>
<tr>
<td>midcall-signaling block</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>end</td>
</tr>
<tr>
<td></td>
<td>Exits to privileged EXEC mode.</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 midcall-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.

Note
In the SRTP-RTP scenario, re-invites are not consumed.

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> <code>enable</code></td>
<td>Enables privileged EXEC mode.</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> <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><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><code>midcall-signaling preserve-codec</code></td>
<td></td>
</tr>
<tr>
<td>- In dial-peer configuration mode</td>
<td></td>
</tr>
<tr>
<td><code>voice-class sip midcall-signaling preserve-codec</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></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
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 midcall-signaling preserve-codec
!

**Example: Configuring Mid Call Codec Preservation 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 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 859
- Prerequisites, on page 860
- Restrictions, on page 860
- Information about Early Dialog UPDATE Block, on page 860
- Configuring Early Dialog UPDATE Block, on page 861
- Configuring Early Dialog UPDATE Block Renegotiate, on page 862
- Troubleshooting Tips, on page 863

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. The following command is introduced: <strong>early-media update block</strong>.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.16S</td>
<td></td>
</tr>
</tbody>
</table>

Table 85: 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 transrating.
• 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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enabling privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
</tr>
<tr>
<td></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>
</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

```bash
! 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

```bash
! 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
     ```bash
     voice-class sip early-media update block re-negotiate
     ```
   - In the global VoIP configuration mode
     ```bash
     early media update block re-negotiate
     ```
4. **end**

Exits VoIP SIP configuration mode and enters privileged EXEC mode.
### Detailed Steps

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
|        | 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

```plaintext
! 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

```plaintext
! 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 ccsip all`
- `debug voip ccapi inout`
- `show voip rtp connections`
[Begin text]

SIP Protocol Handling

Troubleshooting Tips

[End text]
CHAPTER 65

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 865
- Prerequisites, on page 866
- Restrictions, on page 866
- Information About Consumption of Forked 18x Responses with SDP During Early Dialog, on page 866
- Configuring Consumption of Forked 18x Responses with SDP During Early Dialog, on page 867
- Configuring Consumption of Forked 18x Responses with SDP During Early Dialog Renegotiate, on page 868
- Troubleshooting Tips, on page 870

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 86: 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 Cisco IOS XE Denali 16.3.1</td>
<td>This feature allows CUBE to consume multiple forked 18x responses with SDP received during an early dialog.</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`
**SIP Protocol Handling**

**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>
</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  
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 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, on page 871
- Configure SIP INFO Message with Unsupported Content Type, on page 871
- Information About Pass-Through of Unsupported Content Types in SIP INFO Messages, on page 872

Feature Information

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 pass-through of unsupported content types in SIP INFO messages | Cisco IOS 15.5(3)M  
Cisco IOS XE 3.16S | 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. |

Configure SIP INFO Message with Unsupported Content Type

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
- application/x-cisco-record+json
CHAPTER 67

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 85: 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.

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>
<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`  
This is the default system behavior. |

  Note  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. |
### Configuration Notes

<table>
<thead>
<tr>
<th>Incoming Header</th>
<th>Outgoing Header</th>
<th>Notes</th>
</tr>
</thead>
</table>
| PAID            | PPID            | To enable the translation to PPID 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 PPID 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 P PID 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 P PID 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 P PID 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 P PID 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 P PID headers in incoming messages to RPID headers in the outgoing messages. To disable the default behavior and enable the translation from P PID 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` |

---

**Note**  
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 P PID 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 200OK
The priority of P-headers are in the following order: PAID, PPID, and RPID.

Table 88: 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>PPID</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>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>PPID</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>
</tbody>
</table>
### Configuration Notes

<table>
<thead>
<tr>
<th>Incoming Header</th>
<th>Outgoing Header</th>
<th>Configuration Notes</th>
</tr>
</thead>
<tbody>
<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 <strong>asserted-id ppi</strong> 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 <strong>voice-class sip asserted-id ppi</strong> 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 <strong>asserted-id pai</strong> 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 <strong>voice-class sip asserted-id pai</strong> 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 <strong>asserted-id pai</strong> 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 <strong>voice-class sip asserted-id pai</strong> 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>.</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>. Note 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>. Note PAID and PPID headers are not configured in this case.</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 RPID 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: PAID headers will be given priority and RPID headers will be created using the PAID header information.</td>
</tr>
<tr>
<td>PPID</td>
<td>RPID</td>
<td>Enables PPID headers on incoming dial-peer and RPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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>
</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 dial 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.

<table>
<thead>
<tr>
<th>Incoming Header</th>
<th>Outgoing Header</th>
<th>Configuration Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPID</td>
<td>RPID</td>
<td>Enables PAID headers on incoming dial-peer and RPID headers on outgoing dial-peer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> PPID headers will be given priority and RPID headers will be created using the PPID header information.</td>
</tr>
</tbody>
</table>
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:FefhH3zIHe9i8ImcGjDD1PEc5XfFy51G@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 883
- Prerequisites for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element, on page 884
- Restrictions for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element, on page 885
- Configuring P-Header and Random-Contact Support on the Cisco Unified Border Element, on page 885

### 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](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

#### Table 89: 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>
<tbody>
<tr>
<td>PAID and PPID Headers in mid-call re-INVITE and UPDATE request and responses on Cisco Unified Border Element</td>
<td>Cisco IOS 15.5(3)M</td>
<td>This feature enables CUBE platforms to support:</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.16S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• P-Preferred Identity (PPID) and P-Asserted Identity (PAID) in mid-call re-INVITE messages and responses from end-to-end.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• P-Preferred Identity (PPID) and P-Asserted Identity (PAID) in mid-call UPDATE messages and responses from end-to-end.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Configuration and/or pass through of PAID and PPID header values.</td>
</tr>
</tbody>
</table>
Feature History Table entry for the Cisco Unified Border Element and Cisco Unified Border Element (Enterprise).

### Table 90: 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>
<tbody>
<tr>
<td>PAID, PPID, Privacy, PCPID, and PAURI Headers on the Cisco Unified Border Element</td>
<td>12.4(22)YB 15.0(1)M</td>
<td>This feature enables CUBE platforms to support:</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE Release 3.1S</td>
<td>• P-Preferred Identity (PPID), P-Asserted Identity (PAID), Privacy, P-Called Party Identity (PCPID), in INVITE messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Translation of PAID headers to PPID headers and vice versa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Translation of From: or RPID headers to PAID or PPID headers and vice versa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Configuration and/or pass through of privacy header values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PCPID header to route INVITE messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multiple PAURI headers in the response messages (200 OK) it receives to REGISTER messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• P-Preferred Identity and P-Asserted Identity Headers</td>
</tr>
</tbody>
</table>

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</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><code>Router&gt; enable</code></td>
<td>Enables privileged EXEC mode.</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:</td>
<td></td>
</tr>
<tr>
<td><code>Router# 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 VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router(config)# voice service voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>sip</code></td>
<td>Enters voice service VoIP SIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router(conf-voi-serv)# sip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td><code>asserted-id header-type</code></td>
<td>Specifies the type of privacy header in the outgoing SIP requests and</td>
</tr>
<tr>
<td>Example:</td>
<td>response messages.</td>
</tr>
<tr>
<td><code>Router(conf-serv-sip)# asserted-id ppi</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td><code>exit</code></td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router(conf-serv-sip)# exit</code></td>
<td></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>Enters privileged EXEC mode.</td>
</tr>
</tbody>
</table>

**Step 2**

**Example:**

```
Router# configure terminal
```

Enters global configuration mode.

**Step 3**

**Example:**

```
Router(config)# dial-peer voice 2611 voip
```

Defines the dial peer, specifies the method of voice encapsulation, and enters dial peer voice configuration mode.

**Step 4**

**Example:**

```
Router(config-dial-peer)# voice-class sip asserted-id ppi
```

Specifies the type of privacy header in the outgoing SIP requests and response messages, on this dial peer.

**Step 5**

**Example:**

```
Router(config-dial-peer)# exit
```

Exits the current mode.

---

**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></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>

---

Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5

887
<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>configure terminal</td>
<td>Example:</td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>enters voip voice-service configuration mode.</td>
</tr>
<tr>
<td>voice service voip</td>
<td>Example:</td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>enters voice service VoIP SIP configuration mode.</td>
</tr>
<tr>
<td>sip</td>
<td>Example:</td>
</tr>
<tr>
<td>Router(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>enables the routing of calls based on the PCPID header.</td>
</tr>
<tr>
<td>call-route p-called-party-id</td>
<td>Example:</td>
</tr>
<tr>
<td>Router(conf-serv-sip)# call-route p-called-party-id</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>enables the validation of the random string in the Request URI of the incoming INVITE message.</td>
</tr>
<tr>
<td>random-request-uri validate</td>
<td>Example:</td>
</tr>
<tr>
<td>Router(conf-serv-sip)# random-request-uri validate</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>exits the current mode.</td>
</tr>
<tr>
<td>exit</td>
<td>Example:</td>
</tr>
<tr>
<td>Router(conf-serv-sip)# exit</td>
<td></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
## 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>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 1.   | enable            | Enables privileged EXEC mode.  
|      | Example:          | - Enter your password if prompted. |
|      | Router> enable    |         |
| 2.   | configure terminal| Enters global configuration mode. |
|      | Example:          |         |
|      | Router# configure terminal | |
| 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 | |
| 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 | |
| 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 | |
| 6.   | exit              | Exits the current mode. |
|      | Example:          |         |
|      | Router(config-dial-peer)# 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 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>Enters global configuration mode.</td>
</tr>
<tr>
<td>configure terminal Example:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td>voice service voip Example:</td>
<td></td>
</tr>
<tr>
<td>Router(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters voice service VoIP SIP configuration mode.</td>
</tr>
<tr>
<td>sip Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Enables the privacy settings for the header.</td>
</tr>
<tr>
<td>privacy privacy-option Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-serv-sip)# privacy id</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Specifies the privacy policy to use when passing the privacy header from one SIP leg to the next.</td>
</tr>
<tr>
<td>privacy-policy privacy-policy-option Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-serv-sip)# privacy-policy passthru</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td>exit Example:</td>
<td></td>
</tr>
<tr>
<td>Router(conf-serv-sip)# exit</td>
<td></td>
</tr>
</tbody>
</table>

## 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> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router&gt; enable</code></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><code>Router# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> dial-peer voice <code>tag</code> voip</td>
<td>Defines the dial peer, specifies the method of voice encapsulation, and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router(config)# dial-peer voice 2611 voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> voice-class sip <code>privacy-policy</code> <em>privacy-policy-option</em></td>
<td>Enables the privacy settings for the header on this dial peer.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router(config-dial-peer)# voice-class sip privacy-id</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> voice-class sip <code>privacy-policy</code> <em>privacy-policy-option</em></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>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Router(config-dial-peer)# voice-class sip privacy-policy passthru</code></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><code>Router(config-dial-peer)# exit</code></td>
<td></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></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td>Router&gt; enable</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><code>configure terminal</code></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td><code>sip-ua</code></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td>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><code>credentials username username password password realm domain-name</code></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td>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 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.</td>
</tr>
<tr>
<td><code>registrar ipv4: destination-address random-contact expires expiry</code></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td>Router(config-sip-ua)# registrar ipv4:10.1.2.2 random-contact expires 200</td>
<td>• The <code>random-contact</code> 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><code>exit</code></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td>Router(config-sip-ua)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Enters VoIP voice-service configuration mode.</td>
</tr>
<tr>
<td><code>voice service voip</code></td>
<td><strong>Example:</strong></td>
</tr>
</tbody>
</table>
### 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 **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>
<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>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>
</tbody>
</table>
### Configuring Random-Contact Support for an Individual Dial Peer

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong> sip-ua</td>
<td>Enters SIP user-agent configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config)# sip-ua</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> credentials username <strong>username</strong> password <strong>password</strong> realm <strong>domain-name</strong></td>
<td>Sends a SIP registration message from the Cisco Unified Border Element.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sip-ua)# credentials username 123456 password cisco realm cisco</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> registrar ipv4: <strong>destination-address</strong> random-contact expires <strong>expiry</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><strong>Example:</strong> Router(config-sip-ua)# registrar ipv4:10.1.2.2 random-contact expires 200</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sip-ua)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> dial-peer voice <strong>tag</strong> voip</td>
<td>Defines the dial peer, specifies the method of voice encapsulation, and enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config)# dial-peer voice 2611 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> voice-class sip random-contact</td>
<td>Enables random-contact support on this dial peer.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-dial-peer)# voice-class sip random-contact</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> exit</td>
<td>Exits the current mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-dial-peer)# exit</td>
<td></td>
</tr>
</tbody>
</table>
PART XVII

SIP Supplementary Services

• Dynamic Refer Handling, on page 897
• Cause Code Mapping, on page 903
CHAPTER 68

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 897
- Prerequisites, on page 898
- Restrictions, on page 898
- Configuring REFER Passthrough with Unmodified Refer-to, on page 898
- Configuring REFER Consumption, on page 900
- Troubleshooting Tips, on page 902

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 91: Feature Information for Dynamic REFER Handling

<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.</td>
</tr>
<tr>
<td></td>
<td>IOS XE 3.14.0 S</td>
<td>The following commands were introduced: refer consume.</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>Dynamic REFER Handling</td>
<td>IOS 15.2(1)T, IOS XE Release 3.7S</td>
<td>The Dynamic REFER handling feature provides configurations to pass across or consume the REFER message. The following commands were introduced: <code>refer-to-passing</code>, <code>voice-class sip refer-to-passing</code>.</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 configurations 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>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>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>Configure REFER passthrough:</td>
<td>Configures REFER passthrough. A REFER is sent towards the inbound dial peer</td>
</tr>
<tr>
<td>• supplementary-service sip refer in global VoIP configuration mode.</td>
<td></td>
</tr>
<tr>
<td>• supplementary-service sip refer in dial-peer configuration mode.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>In Global VoIP configuration mode:</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(config-voi-serv)# supplementary-service sip refer</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>In dial-peer configuration mode:</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>Command or Action</strong></td>
</tr>
<tr>
<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>• refer-to-passing 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 92: 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. (Optional) supplementary-service media-renegotiate in global VoIP configuration mode.
6. (Optional) Enter one of the following:
   - xfer target in global VoIP configuration mode.
   - xfer target in voice class tenant configuration mode.
### 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>Enter one of the following:</td>
<td>Configures REFER consumption. An INVITE is sent towards the Refer-to dial peer.</td>
</tr>
<tr>
<td></td>
<td>• no supplementary-service sip refer in global VoIP configuration mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• no supplementary-service sip refer in dial-peer configuration mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In global VoIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(conf-voi-serv)# no supplementary-service sip refer</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></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 22 voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# no supplementary-service sip refer</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>refer consume in global VoIP configuration mode.</td>
<td>Configures REFER consumption.</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 22 voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# refer consume</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>(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></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In global VoIP configuration mode:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(conf-voi-serv)# supplementary-service media-renegotiate</td>
<td></td>
</tr>
</tbody>
</table>
Step 6  
(Optional) Enter one of the following:
  •  \texttt{xfer target} in global VoIP configuration mode.
  •  \texttt{xfer target} in voice class tenant configuration mode.

Example:  
In global VoIP configuration mode:

\begin{verbatim}
router(config)#sip-ua
router(config-sip-ua)#xfer target refer-to
\end{verbatim}

Example:  
In voice class tenant configuration mode:

\begin{verbatim}
Router(config)#voice class tenant 1
Router(config-class)#xfer target refer-to
\end{verbatim}

Step 7  end  
Exits to privileged EXEC mode.

Troubleshooting Tips

Use any of the following debug commands:
  •  \texttt{debug ccsip all}
  •  \texttt{debug voip ccapi inout}
  •  \texttt{debug scgp messages}
  •  \texttt{debug voip application supplementary-service}
  •  \texttt{debug voip application state}
  •  \texttt{debug voip application media negotiation}
CHAPTER 69

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 903
- Cause Code Mapping, on page 904
- Configuring Cause Code Mapping, on page 905
- Verifying Cause Code Mapping, on page 906

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>
<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>
Feature Information

Releases

Feature Name

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 86: 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 94: 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>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enters privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>

---

<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>480</td>
<td>31</td>
<td>480 Temporarily Unavailable</td>
</tr>
<tr>
<td>403</td>
<td>21</td>
<td>403 Forbidden</td>
</tr>
<tr>
<td>480</td>
<td>19</td>
<td>503 Service Unavailable</td>
</tr>
<tr>
<td>504</td>
<td>102</td>
<td>504 Server Time-out</td>
</tr>
<tr>
<td>404</td>
<td>1</td>
<td>404 Not Found</td>
</tr>
<tr>
<td>480</td>
<td>20</td>
<td>480 Temporarily Unavailable</td>
</tr>
<tr>
<td>484</td>
<td>28</td>
<td>484 Address Incomplete</td>
</tr>
<tr>
<td>502</td>
<td>27</td>
<td>502 Bad Gateway</td>
</tr>
<tr>
<td>503</td>
<td>38</td>
<td>503 Service Unavailable</td>
</tr>
</tbody>
</table>

---

**Note**

Cause code mappings for cause code 19 and 21 require configurations mentioned in Configuring Cause Code Mapping, on page 905.

**Note**

This mapping is only for the REFER consume scenario and not for REFER passthrough.
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:
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: 7D7073E4-3F3B11E4-917B9A9-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=nosource
Contact: <sip:2222@9.40.3.231:5060>
Content-Type: message/sipfrag
Content-Length: 25

SIP/2.0 486 Busy here
PART XVIII

Hosted and Cloud Services

• Hosted and Cloud Services Delivery with CUBE, on page 911
• CUBE SIP Registration Proxy, on page 913
• Survivability for Hosted and Cloud Services, on page 929
• SUBSCRIBE-NOTIFY Passthrough, on page 949
Cisco Unified Border Element (CUBE) 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 87: Cisco UBE in Hosted and Cloud Services
CHAPTER 71

CUBE SIP Registration Proxy

The CUBE 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 913
- Registration Overload Protection, on page 916
- Registration Rate-limiting, on page 916
- Prerequisites for SIP Registration Proxy on Cisco UBE, on page 917
- Restrictions, on page 917
- Configuring CUBE SIP Registration Proxy, on page 917
- Configuration Example—CUBE SIP Registration Proxy, on page 926
- Feature Information for CUBE SIP Registration Proxy, on page 927

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 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 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.

Note
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 register success scenario for a peer-to-peer registration pass-through mode is as follows:

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. Cisco UBE receives a success message (200 OK message) from the registrar and forwards it to the endpoint (user). The following functions are performed:
   • Cisco UBE picks up the details about the registrar from the configuration.
   • Cisco UBE passes the registrar details and expiry value to the user.

Registrar Challenging the Register Request Scenario

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 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— in-registration timer and 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 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 917.
- You must configure dial peers manually for call routing and pattern matching

Restrictions

- IPv6 support is not provided.

Configuring CUBE 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>Step</th>
<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 configuration mode.  
**Example:**  
Device(config)# voice service voip |
| **Step 4** | sip | Enters service SIP configuration mode.  
**Example:**  
Device(conf-voi-serv)# sip |
| **Step 5** | registrar server [expires [max value] [min value]] | Enables the local SIP registrar.  
**Example:**  
Device(conf-serv-sip)# registrar server  
- 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 | Exits service SIP configuration mode and returns to privileged EXEC mode.  
**Example:**  
Device(conf-serv-sip)# end |
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

<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 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>Step 4 sip</td>
<td>Enters service SIP configuration mode.</td>
</tr>
<tr>
<td>Example: Device(conf-voi-serv)# sip</td>
<td></td>
</tr>
<tr>
<td>Step 5 registration passthrough [system</td>
<td>[static</td>
</tr>
<tr>
<td>Example: Device(conf-serv-sip)# registration passthrough</td>
<td></td>
</tr>
<tr>
<td>• You can specify different SIP registration pass-through options using the following keywords:</td>
<td></td>
</tr>
<tr>
<td>• dynamic—SIP Registration uses the dynamic registrar details (default).</td>
<td></td>
</tr>
<tr>
<td>• local-fallback—Configures Local Fallback - (e2e).</td>
<td></td>
</tr>
<tr>
<td>• rate-limit—Enables rate-limiting.</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg-sync —</td>
<td>Sends REGISTER messages when registrar up (p2p).</td>
</tr>
<tr>
<td>registrar-index —</td>
<td>Configures a list of registrars to be used for registration. For detailed information, see Configuring Multiple Registrars on SIP Trunks.</td>
</tr>
<tr>
<td>static —</td>
<td>SIP Registration Use static Registrar Details.</td>
</tr>
<tr>
<td>system —</td>
<td>Use system registration passthrough configuration.</td>
</tr>
</tbody>
</table>

### Step 6

**Example:**

Device (conf-serv-sip) # end

Exits service SIP configuration mode and returns to privileged EXEC mode.

---

### 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] registrar-server-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>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**
Example: |

Device> enable

Enables privileged EXEC mode.

- Enter your password if prompted.

| **Step 2**
Example: |

Device# configure terminal

Enters global configuration mode.
<table>
<thead>
<tr>
<th>Step 3</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>voice class tenant</strong> <em>tag</em></td>
<td></td>
<td>Enters the tenant configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice class tenant 1</td>
<td></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>**registrar [ dhcp</td>
<td>[registrar index]**</td>
<td>Configures the registrar server.</td>
</tr>
<tr>
<td>**registrar-server-address [:port]</td>
<td>expires value]**</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)#registrar ipv4:10.65.75.45:9052 expires 3600</td>
<td></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>**registration passthrough [system</td>
<td>[static</td>
<td>dynamic [local-fallback value] ] [rate-limit [expires value] [fail-count value]] [reg-sync value] [registrar-index index]]**</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# registration passthrough static</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- You can specify different SIP registration pass-through options using the following keywords:
  - **dynamic**—SIP Registration uses the dynamic registrar details (default).
  - **local-fallback**—Configures Local Fallback - (e2e).
  - **rate-limit**—Enables rate-limiting.
  - **reg-sync**—Sends REGISTER messages when registrar up (p2p).
  - **registrar-index**—Configures a list of registrars to be used for registration. For detailed information, see Configuring Multiple Registrars on SIP Trunks.
  - **static**—SIP Registration Use static Registrar Details.
  - **system**—Use system registration passthrough configuration.

<table>
<thead>
<tr>
<th>Step 6</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td></td>
<td>Exits tenant configuration mode and returns to global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config-class)# exit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 7</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dial-peer voice</strong> <em>tag</em> <em>voip</em></td>
<td></td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device(config)# dial-peer voice 444 voip</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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  
4. voice-class sip registration passthrough [system | [static | dynamic [local-fallback value]]] [rate-limit [expires value] [fail-count value]] [reg-sync value] [registrar-index index]]  
5. 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 444 voip | Enters dial peer voice configuration mode. |
| **Step 4** voice-class sip registration passthrough [system | [static | dynamic [local-fallback value]]] [rate-limit [expires value] [fail-count value]] [reg-sync value] [registrar-index index]] | Configures SIP registration pass-through options on a dial peer on a dial peer.  
• You can specify different SIP registration pass-through options using the following keywords: |
### Purpose

<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>Device(config-dial-peer)# voice-class sip registration passthrough static</code></td>
<td></td>
</tr>
</tbody>
</table>

- **dynamic**—SIP Registration uses the dynamic registrar details (default).
- **local-fallback**—Configures Local Fallback (e2e).
- **rate-limit**—Enables rate-limiting.
- **reg-sync**—Sends REGISTER messages when registrar up (p2p).
- **registrar-index**—Configures a list of registrars to be used for registration. For detailed information, see [Configuring Multiple Registrars on SIP Trunks](#).
- **static**—SIP Registration Use static Registrar Details.
- **system**—Use system registration passthrough configuration.

### Step 5

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td>Exits dial peer voice configuration mode and returns to global configuration mode.</td>
</tr>
</tbody>
</table>

#### Exit Example:

`Device(config-dial-peer)# exit`

## 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>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><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
</tbody>
</table>

- Enter your password if prompted.
### 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></td>
<td></td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
</tbody>
</table>
Purpose

Command or Action | Purpose
--- | ---
Step 2 configure terminal | Enters global configuration mode.
Example:
Device# configure terminal

Step 3 dial-peer voice tag \{pots | voatm | vofr | voip\} | Enters dial peer voice configuration mode.
Example:
Device(config)# dial-peer voice 444 voip

Step 4 session target registrar | Configures Cisco UBE to route the call to the registrar endpoint.
Example:
Device(config-dial-peer)# session target registrar

Step 5 exit | Exits dial peer voice configuration mode and returns to global configuration mode.
Example:
Device(config-dial-peer)# exit

---

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
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—CUBE SIP Registration Proxy

! 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 pass-thru content unsupp
    session protocol sipv2
    session target registrar

!
Feature Information for CUBE SIP Registration Proxy

Table 95: Feature Information for Support for SIP Registration Proxy on CUBE

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for CUBE SIP Registration Proxy</td>
<td>Cisco IOS XE</td>
<td>CUBE SIP Registration Proxy supports sending outbound registrations from CUBE 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 CUBE deployments with no IPPBX support.</td>
</tr>
<tr>
<td></td>
<td>Fuji 16.9.1</td>
<td>The following commands were introduced or modified: <strong>authentication</strong> (dial peer), <strong>registrar server</strong>, <strong>registration passthrough</strong>, <strong>registration spike</strong>, <strong>show sip-ua registration passthrough status</strong>, <strong>voice-class sip registration passthrough static rate-limit</strong>.</td>
</tr>
</tbody>
</table>
Survivability for Hosted and Cloud Services

The Survivability for Hosted and Cloud Services on the CUBE is used to:

• Monitor the WAN status periodically from the CUBE.
• 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

Advantages of Using CUBE Survivability Feature

The survivability feature on CUBE 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

• CUBE does not need to configure credentials, as the phones trigger registration. Although CUBE 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, CUBE replies with a 500 message and when the phone receives this message, it sends the REGISTER message to CUBE, which is the secondary server (P2P registration). CUBE 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 CUBE 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 CUBE keeps the registration database of the SIP phones that were previously registered successfully, and it does not send REGISTER messages out; CUBE 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 CUBE 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 CUBE.

Figure 88: 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. CUBE 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, CUBE 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, CUBE 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.

**CUBE 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 89: WAN Link is UP - CUBE 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 CUBE.
2. CUBE sends this INVITE to the service provider.
3. The service provider in turn sends the INVITE to CUBE. 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. CUBE sends INVITE with all the above mentioned information to Phone B.
5. Phone B sends a 200 OK response to CUBE for the received INVITE.
6. CUBE sends a 200 OK answer to the service provider.
7. The service provider responds to CUBE with a 200 OK answer.
8. A final 200 OK response is sent to Phone A by CUBE 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
```

**CUBE Survivability When WAN Is Down**

In survivability mode, CUBE 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.

*Figure 90: CUBE Survivability When WAN Is Down*

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 CUBE. CUBE 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 CUBE.
It is possible to register multiple contacts for a single AOR; however, if multiple contacts are registered for a single subscriber, the CUBE 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, send register request to primary registrar only and do not send secondary REGISTER request to the secondary registrar (CUBE) in E2E mode when primary registrar could not be reached. In such scenarios, phone service goes down after it receives 500 response from CUBE for REGISTER request toward primary registrar.

To avoid phones getting into such error condition, CUBE checks for the response from the primary registrar side. When CUBE 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 937.

**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 CUBE running on Cisco 4000 Series Integrated Services Router.

---

**Figure 91: 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 Through Cisco IOS XE 17.5**

933
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>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 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>Step 4 sip</td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>

Example: Survivability Mode in E2E (local fallback mode) when WAN is Down

CUBE# show sip-ua registration passthrough status

<table>
<thead>
<tr>
<th>CallId</th>
<th>DirectoryNum</th>
<th>peer</th>
<th>mode</th>
<th>In-Exp</th>
<th>reg-I</th>
<th>Out-Exp</th>
<th>survival</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>NCPhone1006</td>
<td>1</td>
<td>e2e</td>
<td>35 /70</td>
<td>--</td>
<td>0</td>
<td>locfall</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CallId</th>
<th>DirectoryNum</th>
<th>peer</th>
<th>mode</th>
<th>In-Exp</th>
<th>reg-I</th>
<th>Out-Exp</th>
<th>survival</th>
</tr>
</thead>
<tbody>
<tr>
<td>513</td>
<td>NCPhone1008</td>
<td>1</td>
<td>e2e</td>
<td>40 /70</td>
<td>--</td>
<td>0</td>
<td>locfall</td>
</tr>
</tbody>
</table>
Purpose
Command or Action

Device(conf-voi-serv)# sip

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.

Example:
Step 5
registration passthrough local-fallback tag

Example:
Device(conf-serv-sip)# registration passthrough local-fallback 10

• 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-serv-sip)# end

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>
<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></td>
<td>Device&gt; enable</td>
</tr>
<tr>
<td>Step 2 configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
</tr>
<tr>
<td>Step 3 voice class tenant tag</td>
<td>Enters voice class tenant configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<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></td>
</tr>
<tr>
<td>2.</td>
<td>configure terminal</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>dial-peer voice tag voip</td>
<td>Enters dial peer voice configuration mode.</td>
</tr>
<tr>
<td>4.</td>
<td>voice-class sip tenant tag</td>
<td>Associates the dial-peer with the tenant.</td>
</tr>
<tr>
<td>5.</td>
<td>exit</td>
<td>Exits dial-peer configuration mode and returns to global configuration mode.</td>
</tr>
</tbody>
</table>

### 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 registration passthrough local-fallback tag
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>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Device&gt; enable</strong></td>
</tr>
<tr>
<td></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>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Device# configure terminal</strong></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters dial peer VoIP configuration mode.</td>
</tr>
<tr>
<td>dial-peer voice tag voip</td>
<td>Enters dial peer VoIP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Device(config)# dial-peer voice 4 voip</strong></td>
</tr>
<tr>
<td><strong>Step 4</strong></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>voice-class sip registration passthrough local-fallback</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><strong>Example:</strong></td>
<td><strong>Device(config-dial-peer)# voice-class sip registration passthrough local-fallback 10</strong></td>
</tr>
<tr>
<td></td>
<td>• To configure the registration sync mode, you can use the <strong>voice-class sip registration passthrough reg-sync tag</strong> command.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Device(conf-serv-sip)# end</strong></td>
</tr>
</tbody>
</table>

### Configuring Survivability for Phones Sending Single Register Request

The following configuration enables CUBE to always check for the response from remote side. Request timeout on WAN side or response other than 200, 4XX, and 3XX received by CUBE 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.  
  **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 CUBE 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 CUBE 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></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><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>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></td>
<td></td>
</tr>
<tr>
<td>dial-peer voice tag voip</td>
<td>Enters dial peer configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# dial-peer voice 3 voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>voice-class sip options-keepalive up-interval \ value down-interval \ value</td>
<td>Configures OPTIONS keepalive timer interval for DOWN and UP endpoints.</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# voice-class sip options-keepalive up-interval 120 down-interval 120</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>

Configuring Registration Timer

Perform the following task to configure the registration timer in the CUBE 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>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 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> registrar server expires max value min value</td>
<td>Configures the maximum and minimum time (in seconds) for the registration expiry in CUBE.</td>
</tr>
<tr>
<td>Example: Device(conf-serv-sip)# registrar server expires max 300 min 200</td>
<td>• If the phone sends expiry time as 600 seconds, then the CUBE 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>Example: Device(conf-serv-sip)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Configuring the REGISTER Message Throttling in CUBE

Perform the following task to throttle the REGISTER message in CUBE.

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice service voip
4. sip
5. registration passthrough rate-limit expires value local-fallback tag
### 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></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:</td>
<td></td>
</tr>
<tr>
<td><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:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# voice service voip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>sip</code></td>
<td>Enters voice service SIP configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(conf-voi-serv)# sip</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td><code>registration passthrough rate-limit expires value local-fallback tag</code></td>
<td>Configures the SIP registration passthrough rate-limit expiry value for local-fallback (e2e). Although CUBE receives the REGISTER message every 5 minutes (300 seconds), it will send only one register message every one hour.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(conf-serv-sip)# registration passthrough rate-limit expires 3600 local-fallback 3</code></td>
<td>• Under dial peer configuration mode, you can use the <code>voice-class sip registration passthrough rate-limit expires value reg-sync dial-peer-tag</code> command.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td><code>end</code></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(conf-serv-sip)# end</code></td>
<td></td>
</tr>
</tbody>
</table>

### 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.
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>
</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** | dial-peer voice tag voip | Enters dial peer configuration mode.  
  Example:  
  Device(config)# dial-peer voice 3 voip |
| **Step 4** | corlist incoming dial-peer | Specifies the COR to be applied on an incoming dial peer  
  (for incoming calls).  
  Example:  
  Device(config-dial-peer)# corlist incoming FromPhone |
| **Step 5** | corlist outgoing dial-peer | Specifies the COR to be applied for outgoing dial peer  
  (for outgoing calls).  
  Example:  
  Device(config-dial-peer)# corlist outgoing FromSP |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 6</strong></td>
<td>Adds a description to a dial peer.</td>
</tr>
<tr>
<td>description <em>string</em></td>
<td>Specifies either the prefix or the full E.164 phone number to be used for the dial peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# description registration</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>Device(config-dial-peer)# destination-pattern 1111</td>
</tr>
<tr>
<td>destination-pattern <em>number</em></td>
<td>Specifies to route the call to the registrar endpoint for SIP dial peers.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# session target registrar</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>Specifies the session protocol for SIP calls between local and remote devices using the packet network.</td>
</tr>
<tr>
<td>session protocol sipv2</td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# session protocol sipv2</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>Configures SIP registration passthrough for local fallback mode.</td>
</tr>
<tr>
<td>session target registrar</td>
<td>Device(config-dial-peer)# session target registrar</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# session target registrar</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>Shows the configuration of voice-class sip registration passthrough local-fallback tag.</td>
</tr>
<tr>
<td>voice-class sip registration passthrough local-fallback <em>tag</em></td>
<td>Device(config-dial-peer)# voice-class sip registration passthrough local-fallback 5</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config-dial-peer)# voice-class sip registration passthrough local-fallback 5</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>end</td>
<td>Device(config-dial-peer)# end</td>
</tr>
</tbody>
</table>

**Verifying Survivability for Hosted and Cloud Services**

The *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

```
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>FER</th>
<th>THRU</th>
<th>SESS-TARGET</th>
<th>STAT</th>
<th>PORT</th>
<th>KEEPALIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>voip</td>
<td>up</td>
<td>up</td>
<td>1111...</td>
<td>0 syst registrar</td>
<td>0</td>
<td>syst ipv4:10.104.45.253</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>voip</td>
<td>up</td>
<td>down</td>
<td>1......</td>
<td>0 syst ipv4:9.0.0.174:30601</td>
<td></td>
<td></td>
<td>busyout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>voip</td>
<td>down</td>
<td>down</td>
<td>9990...</td>
<td>0 syst ipv4:10.104.45.31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>voip</td>
<td>down</td>
<td>down</td>
<td>11......</td>
<td>0 syst ipv4:10.104.45.253</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>voip</td>
<td>down</td>
<td>down</td>
<td>111......</td>
<td>0 syst ipv4:10.104.45.253</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>voip</td>
<td>down</td>
<td>down</td>
<td>.T</td>
<td>0 syst</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>voip</td>
<td>down</td>
<td>down</td>
<td>11110...</td>
<td>0 syst registrar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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

```
CallId  Line  peer  mode  In-Exp  reg-I  Out-Exp  survival
-------- ----- ----- ------ -------- ------ -------- -------
5300    1111008 1  e2e  1041 /1200 ----- 1200 normal *
5305    1111002 1  e2e  2847 /3000 ----- 3000 normal *
5311    1111020 1  e2e  1070 /1200 ----- 1200 normal *
```

**Step 4**

**show sip-ua register status**

Displays information about the SIP user agent register status.

**Example:**

Device# show sip-ua register status

```
Line  peer  expires(sec)  reg  survival  P-Associ-URI
------ ------ ------------- --- ---------- ------------
11123  23    59            yes  registrc
```

**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>Port</th>
<th>Ports</th>
<th>Media-Address Range Available</th>
<th>Reserved</th>
<th>In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>2</td>
<td>8091</td>
<td></td>
<td></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

Total call-legs: 2

<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 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>
</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>
</tr>
</tbody>
</table>

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. An account on Cisco.com is not required.

Table 96: 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>
CHAPTER 73

SUBSCRIBE-NOTIFY Passthrough

The SUBSCRIBE-NOTIFY mechanism is used for implementation of features such as Message Waiting Indication (MWI), Shared Call Appearance, Multiple Caller Appearance, Busy Lamp Field, and so on.

In CUBE, the SUBSCRIBE-NOTIFY framework on Unified Communications (UC) products supports the following:

- Configurable and Selective Passthrough of SUBSCRIBE and NOTIFY transactions from phones with the normalization that is required for address or topology hiding and dialog content updates for “dialog” event subscription.

- Survivability mode handling of incoming SUBSCRIBE request for critical events.

- Restrictions for SUBSCRIBE-NOTIFY Passthrough, on page 949
- Information About SUBSCRIBE-NOTIFY Passthrough, on page 950
- Configure SUBSCRIBE-NOTIFY Passthrough, on page 951
- Configuration Examples for SUBSCRIBE-NOTIFY Passthrough, on page 956
- Feature Information for SUBSCRIBE-NOTIFY Passthrough, on page 957

Restrictions for SUBSCRIBE-NOTIFY Passthrough

- The SUBSCRIBE-NOTIFY passthrough framework can only pass through events when there is a one-to-one association between the incoming request and the outgoing location to which the request has been sent out. This means that either:
  - There should be an outbound dial-peer identified for the request received.
  - The outbound target for the request could be only a single registrar.

- The following use cases are not supported:
  - SUBSCRIBE-NOTIFY passthrough with hunting of outbound dial-peers to which the subscribe or notify requests need to be sent.
  - SUBSCRIBE-NOTIFY passthrough where an inbound dial-peer has peer-to-peer mode of registration passthrough enabled with more than one registrar (there will be no forking of Subscribe-Notify Requests)
Information About SUBSCRIBE-NOTIFY Passthrough

The key attributes of the SUBSCRIBE-NOTIFY Passthrough feature are as follows:

- **Message Passthrough Application (MPA)**—The SIP MPA handles SUBSCRIBE-NOTIFY passthrough. This application maintains subscribe dialogs, and references the dial-peer database and registration passthrough configurations to route the initial SUBSCRIBE and unsolicited NOTIFY requests.

- **Header Passthrough**—All the non-mandatory headers in SUBSCRIBE-NOTIFY requests and responses are passed through from one endpoint to the other.

- **Content Passthrough**—The content bodies in SUBSCRIBE-NOTIFY requests are passed through transparently from one endpoint to the other.

- **“Dialog” Event Content Manipulation**—The content in the NOTIFY body for a dialog event is updated before passthrough when the dialog is maintained by the CUBE.

- **Passthrough Configuration and Filtering**—SUBSCRIBE-NOTIFY passthrough is configurable globally as well as under dial-peer, and can be configured for selected events using the configuration of an event list.

- **Error Passthrough for SUBSCRIBE-NOTIFY Requests**—When an error is received for a SUBSCRIBE-NOTIFY request, the error is passed through to the peer with the relevant headers.

- **Backward Compatibility**—The SIP MPA has the highest priority when SUBSCRIBE-NOTIFY passthrough is enabled. If passthrough is not enabled (either for all events or for a specific event), the current applications will control the incoming requests and responses.

- **401/407 Error Message Passthrough**—SIP message 401/407 is sent by the user-agent server (UAS) or end device to challenge messages like INVITE/REFER/SUBSCRIBE and request for endpoint credentials information. CUBE does not store endpoint credential information to act on behalf of phone or endpoint. To enable the passthrough of 401/407, you can enable the `error passthru` command at the global level. The messages 401/407 are in passthru mode for INVITE/REFER/SUBSCRIBE.

SUBSCRIBE-NOTIFY Passthrough Request Routing

The first step of request or response routing is for CUBE to determine whether or not the request has to be passed through. When a new SUBSCRIBE or unsolicited NOTIFY request arrives, its headers are used to match an incoming dial-peer. If the incoming dial-peer has SUBSCRIBE-NOTIFY Passthrough (SNPT) enabled or if there is no incoming dial-peer and global SNPT is enabled for that event, then the request is handed off to be passed through. For solicited subscriptions, the passthrough check is applicable only to the initial SUBSCRIBE request; subsequent requests or responses are not checked and will be routed based on updated dialog parameters.

The second step is to determine the outbound destination of the SUBSCRIBE or unsolicited NOTIFY request.
• First Pass: Outbound dial-peer match—An outbound VoIP dial-peer is first matched based on the request headers (From, To, and Via), the Subscriber Number (userid in the To header), and the incoming dial-peer Class of Restrictions (CoR) if any. If there is a match, the request is routed to the session target.

• Second Pass: Configured registrar for registration passthrough in peer-to-peer mode—If no outbound dial-peer is found and the incoming dial-peer has registration passthrough enabled in static (peer-to-peer) mode with a single registrar configured, then the request is routed to the registrar address.

• Third Pass: Configured registrar for registration passthrough in end-to-end mode—If no outbound dial-peer is found and the incoming dial-peer has registration passthrough enabled in dynamic (end-to-end) mode:
  • If the request Uniform Resource Identifier (URI) has the CUBE IP address, the request is routed to the configured registrar if only a single registrar is configured.
  • If the request URI has a non-CUBE IP address, then the request is routed to that IP address.

• Fourth Pass: Request URI-based routing—If no outbound dial-peer is found and no registration passthrough is configured, the request URI is used to route the request if it does not point to the CUBE’s IP address.

**SUBSCRIBE-NOTIFY Passthrough Survivability Mode**

In survivability mode, the CUBE could encounter the following scenarios:

• When the CUBE receives a line-seize (event) subscribe in survivability mode, it checks the line-seize queue to see if another phone has already seized the same line; if not, CUBE accepts the subscription, sends a NOTIFY response with State = Active, and starts the timer for expiration. In survivability mode, SUBSCRIBE received for any event other than line-seize is rejected.

• If another phone has already subscribed for the line, CUBE sends a 200 OK (request successful) response for the new subscribe, but a final NOTIFY to indicate that the subscription has been terminated.

• If the subscription timer expires without re-subscription from the phone, CUBE sends a final NOTIFY to remove the subscription.

• If a subscription is created in active mode, but re-subscriptions or unsubscriptions are received in survivability mode, then CUBE returns an error for this subscription.

**Configure SUBSCRIBE-NOTIFY Passthrough**

**Configuring an Event List**

**SUMMARY STEPS**

1. enable
2. configure terminal
3. voice class sip-event number
4. event name
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# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> voice class sip-event <em>number</em></td>
<td>Enters voice class configuration mode and configures the list of events to be passed through.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# voice class sip-event 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> event <em>name</em></td>
<td>Adds the name of the event to be added to the event list.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# event message-summary</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-class)# end</td>
<td></td>
</tr>
</tbody>
</table>

### Configuring SUBSCRIBE-NOTIFY Event Passthrough Globally

### SUMMARY STEPS

1. enable
2. configure terminal
3. voice service voip
4. sip
5. pass-thru subscribe-notify-events *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> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>

### Step 2

**configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

### Step 3

**voice service voip**

**Example:**

```
Device(config)# voice class voip
```

Enters voice service VoIP configuration mode.

### Step 4

**sip**

**Example:**

```
Device(conf-voi-serv)# sip
```

Enters voice service SIP configuration mode.

### Step 5

**pass-thru subscribe-notify-events** *tag*

**Example:**

```
Device(conf-serv-sip)# pass-thru subscribe-notify-events 1
```

Confines SUBSCRIBE-NOTIFY passthrough event with the SIP event list tag number to be linked globally.

- You can use the **pass-thru subscribe-notify-events all** command to configure passthrough for all SUBSCRIBE-NOTIFY events.

### Step 6

**end**

**Example:**

```
Device(conf-serv-sip)# end
```

Returns to privileged EXEC mode.

---

### Configuring SUBSCRIBE-NOTIFY Event Passthrough at the Dial-Peer Level

**SUMMARY STEPS**

1. enable
2. configure terminal
3. dial-peer voice *tag voip*
4. voice-class sip pass-thru subscribe-notify-events *tag*
5. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
</tbody>
</table>

**Step 2**

**configure terminal**

*Example:*

Device# configure terminal

**Step 3**

**dial-peer voice tag voip**

*Example:*

Device(config)# dial-peer voice 123 voip

**Step 4**

**voice-class sip pass-thru subscribe-notify-events tag**

*Example:*

Device(config-dial-peer)# voice-class sip pass-thru subscribe-notify-events 1

- Configures SUBSCRIBE-NOTIFY passthrough event with the SIP event list tag number to be linked globally.
- You can use the **voice-class sip pass-thru subscribe-notify-events all** command to configure passthrough for all SUBSCRIBE-NOTIFY events.

**Step 5**

**end**

*Example:*

Device(conf-serv-sip)# end

**Verifying SUBSCRIBE-NOTIFY Passthrough**

Perform this task to verify the configuration for SUBSCRIBE-NOTIFY Passthrough and to verify the subscriptions created. The **show** commands can be entered in any order.

**SUMMARY STEPS**

1. enable
2. show dial-peer voice *number* | inc pass
3. show subscription asnl session active
4. show subscription sip

**DETAILED STEPS**

**Step 1**

**enable**

Enables privileged EXEC mode.

*Example:*

Device> enable
Step 2  show dial-peer voice  number  | inc pass

Displays the information for voice dial peers. The following sample output shows the configured SUBSCRIBE-NOTIFY passthrough event for a particular dial peer.

Example:
Device# show dial-peer voice 123 | inc pass

ip media DSCP = ef, ip media rsvp-pass DSCP = ef
ip video rsvp-none DSCP = af41, ip video rsvp-pass DSCP = af41
voice class sip pass-thru headers = system,
voice class sip pass-thru subscribe-notify-events = system,
voice class sip pass-thru content unsupp = system,
voice class sip pass-thru content sdp = system,
voice class sip privacy-policy passthru = system,
voice class sip registration passthrough = System
voice class sip referto-passing = system

Step 3  show subscription asnl session active

Displays information about Application Subscribe/Notify Layer (ASNL)-based and non-ASNL-based SIP subscriptions.

Example:
Device# show subscription asnl session active

ASNL Active Subscription Records Details:
-----------------------------------------------
Number of active subscriptions: 1
URL: sip:user@10.7.104.88
   Event Name : stress
   Session ID : 8
   Expiration Time : 50 seconds
   Subscription Duration : 5 seconds
   Protocol : ASNL_PROTO_SIP
   Remote IP address : 10.7.104.88
   Port : 5060
   Call ID : 5
   Total Subscriptions Sent : 1
   Total Subscriptions Received: 0
   Total Notifications Sent : 0
   Total Notifications Received : 2
   Last response code : ASNL_NOTIFY_RCVD
   Last error code : ASNL_NONE
   First Subscription Time : 10:55:12 UTC Apr 9 2000
   Last Subscription Time : 10:55:12 UTC Apr 9 2000
   First Notify Time : 10:55:12 UTC Apr 9 2000
   Last Notify Time : 10:55:17 UTC Apr 9 2000
   Application that subscribed : stress
   Application receiving notification: stress

Step 4  show subscription sip

Displays information about ASNL-based and non-ASNL-based SIP subscriptions.

Example:
Device# show subscription sip

ASNL Active Subscription Records Summary:
-----------------------------------------------
Number of active subscriptions: 2

<table>
<thead>
<tr>
<th>SubId</th>
<th>CallId</th>
<th>Proto</th>
<th>URL</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A</td>
<td>ASNL_PROTO_SIP</td>
<td>&quot;Plutus&quot; <a href="">sip:1111003@primary</a></td>
<td>all</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>ASNL_PROTO_SIP</td>
<td>sip:1111003@primaryappserver1</td>
<td>as-feature</td>
</tr>
</tbody>
</table>

Client EXPIRES(sec) EVENT

<table>
<thead>
<tr>
<th>Client</th>
<th>EXPIRES(sec)</th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111003</td>
<td>0</td>
<td>as-feature-event</td>
</tr>
<tr>
<td>1234</td>
<td>0</td>
<td>message-summary</td>
</tr>
</tbody>
</table>

Troubleshooting Tips

Use the following commands to troubleshoot SUBSCRIBE-NOTIFY Passthrough:

- `debug mpa events`
- `debug mpa error`
- `debug ccsip messages`
- `debug asnl events`
- `debug asnl error`
- `debug ccsip all`

Configuration Examples for SUBSCRIBE-NOTIFY Passthrough

Example: Configuring an Event List

The following example shows how to configure an event list and add an event to the list of events that have to be passed through.

```
Device> enable
Device# configure terminal
Device(config)# voice class sip-event-list 1
Device(config-class)# event 1 message-summary
Device(config-class)# end
```

Example: Configuring SUBSCRIBE-NOTIFY Event Passthrough Globally

The following example shows how to configure the SUBSCRIBE-NOTIFY passthrough event and link the SIP event list tag number globally.
Example: Configuring SUBSCRIBE-NOTIFY Event Passthrough under a Dial Peer

The following example shows how to configure the SUBSCRIBE-NOTIFY passthrough event and link the SIP event list tag number under a dial peer.

```sh
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# pass-thru subscribe-notify-events 1
Device(conf-serv-sip)# end
```

Feature Information for SUBSCRIBE-NOTIFY Passthrough

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.
The SUBSCRIBE-NOTIFY mechanism is used for implementation of features such as Message Waiting Indication (MWI), Shared Call Appearance, Multiple Caller Appearance, Busy Lamp Field, and so on. In CUBE, the SUBSCRIBE-NOTIFY framework on Unified Communications (UC) products supports the following:

- Configurable and Selective Passthrough of SUBSCRIBE and NOTIFY transactions from phones with the normalization required for address or topology hiding and dialog content updates for “dialog” event subscription.
- Survivability mode handling of incoming SUBSCRIBE request for critical events.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBE (SUBSCRIBE-NOTIFY Passthrough)</td>
<td>Cisco IOS XE Fuji 16.9.1</td>
<td>The SUBSCRIBE-NOTIFY mechanism is used for implementation of features such as Message Waiting Indication (MWI), Shared Call Appearance, Multiple Caller Appearance, Busy Lamp Field, and so on. In CUBE, the SUBSCRIBE-NOTIFY framework on Unified Communications (UC) products supports the following: • Configurable and Selective Passthrough of SUBSCRIBE and NOTIFY transactions from phones with the normalization required for address or topology hiding and dialog content updates for “dialog” event subscription. • Survivability mode handling of incoming SUBSCRIBE request for critical events.</td>
</tr>
</tbody>
</table>
PART XIX

Cisco Unified Communications Manager Line-Side Support

- Cisco Unified Communications Manager Line-Side Support, on page 961
Cisco Unified Communications Manager
Line-Side Support

Note
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 961
- Restrictions for Cisco Unified Communications Manager Line-Side Support, on page 962
- Information About Cisco Unified Communications Manager Line-Side Support, on page 963

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 98: 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: extension cucm and voice-class sip extension cucm.</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: access-secure, capf-address, clear voice phone-proxy all-sessions, complete (ctl file), ctl-file (phone proxy), debug voice phone-proxy, description (ctl file), description (phone proxy), disable service-settings, max-concurrent-sessions, phone-proxy (dial peer), port-range, record-entry, show voice class ctl-file, show voice class phone-proxy, service-map, session-timeout, tftp-server address, voice-ctl-file, voice-phone-proxy.</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.

Figure 92: Cisco UBE Line-Side Deployment

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 99: 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 93: Predefined Supported NonMandatory Headers

| voice class sip hdr-pass through list 20001 | pass thru hdr Call-Info |
| pass thru hdr Content-ID | pass thru hdr Allow-Events |
| pass thru hdr Supported | pass thru hdr Remote-Party-ID |
| pass thru hdr Require | pass thru hdr Referred-By |

When Line Side Support for CUCM on CUBE is configured, predefined SIP profiles automatically remove the Cisco-Guide header from the outgoing INVITE.

Figure 94: Predefined SIP Profile

| voice class sip profile 20001 | request INVITE sip-header Cisco-Guide remove |

Note

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>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-ca-trustpoint)# 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 |
### Purpose

**Command or Action**

**Purpose**

**Step 6**

- `revocation-check method1[method2 [method3]]`
  - **Example:**
    
    ```
    Device(config-ca-trustpoint)# revocation-check crl
    ```
  - Checks the revocation status of a certificate.

**Step 7**

- `rsakeypair key-label`
  - **Example:**
    
    ```
    Device(config-ca-trustpoint)# rsakeypair pp1
    ```
  - Specifies which RSA keypair to associate with the certificate.

### What to do next

Import the CUCM and CAPF key.

### 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>
</table>
| **Step 1**

  - `crypto pki trustpoint name`
  - **Example:**
    
    ```
    Device(config)# crypto pki trustpoint cucm_trustpoint
    ```

  - Creates a trustpoint for the CUCM key and enters ca-trustpoint configuration mode. |
| **Step 2**

  - `revocation-check method1[method2 [method3]]`
  - **Example:**
    
    ```
    Device(config-ca-trustpoint)# revocation-check none
    ```

  - Checks the revocation status of a certificate. |
### 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`

**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>voice-ctl-file ctl-filename</code></td>
<td>Creates a CTL file and enters CTL file configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)#voice-ctl-file ctl1</td>
<td></td>
</tr>
</tbody>
</table>

| **Step 2** | record-entry selfsigned trustpoint trustpoint-name | Configures the trustpoints to be used for creating the CTL file. |
| Example:    | Device(config-ctl-file)#record-entry selfsigned trustpoint self-trustpoint6s | |

**What to do next**

Repeat the above steps for the CAPF key (the CAPF.pem file).
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>Configures a phone proxy and enters phone-proxy configuration mode.</td>
</tr>
<tr>
<td><code>voice-phone-proxy phone-proxy-name</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice-phone-proxy pp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Configures the phone-proxy file buffering parameter, in MB.</td>
</tr>
<tr>
<td><code>voice-phone-proxy file-buffer size</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice-phone-proxy file-buffer 30</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Configures the TFTP server address.</td>
</tr>
<tr>
<td>`tftp-server-address [ipv4 server-ip-address</td>
<td>domain-name]`</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>tftp-server-address ipv4 172.110.36.2</code></td>
<td>Configures the CTL filename.</td>
</tr>
<tr>
<td><strong>Step 4</strong> ctl-file <code>ctl-filename</code></td>
<td>Specifies that the secure (encrypted) mode is to be used for access.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>ctl-file ct1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> access-secure</td>
<td>Completes the phone-proxy configuration.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>access-secure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> complete</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> <code>complete</code></td>
<td></td>
</tr>
</tbody>
</table>

### 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>
<tbody>
<tr>
<td><strong>Step 1</strong> dial-peer voice <code>tag voip</code></td>
<td>Defines a particular dial peer, specifies the method of voice encapsulation, and enters dial peer configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>dial-peer voice 10 voip</code></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></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></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# phone-proxy pp1 signal-addr ipv4 10.0.0.8 cucm ipv4 198.51.100.1</code></td>
<td></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></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# session protocol sipv2</code></td>
<td></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></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# session target registrar</code></td>
<td></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><code>session transport {udp [tcp [tls]]}</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# session transport tcp tls</code></td>
<td></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></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# incoming uri request 11</code></td>
<td></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></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# destination uri 12</code></td>
<td></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></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# voice-class sip call-route url</code></td>
<td></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></td>
</tr>
<tr>
<td><code>Device(config-dial-peer)# voice-class sip profiles 10</code></td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 10 voice-class sip registration passthrough [registrar-index index]</td>
<td>Configures the SIP registration pass-through options on the dial peer.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip registration passthrough registrar-index 1</td>
<td></td>
</tr>
<tr>
<td>Step 11 voice-class sip pass-thru headers</td>
<td>Configures a list of headers for pass through by referring to a globally configured list.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip pass-thru headers 10</td>
<td></td>
</tr>
<tr>
<td>Step 12 voice-class sip copy-list {tag</td>
<td>system}</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# voice-class sip copy-list 10</td>
<td></td>
</tr>
<tr>
<td>Step 13 codec transparent</td>
<td>Enables codec capabilities to be passed transparently between endpoints in a Cisco Unified Border Element.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# codec transparent</td>
<td></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</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.

```
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

```
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

```
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

```
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 55C2FCBFAC552B7C6CED497D4AD33F8
[Certificate data omitted]

Device(config)# crypto pki certificate chain Cisco_Manufacturing_CA
Device(config)# certificate ca 6A6967B3000000000003
[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 ls-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 local-addr ipv4 10.50.209.100 acc-addr ipv4 172.18.110.120
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 8443
Device(config-phone-proxy)# service-map server-addr ipv4 10.50.209.215 port 8080 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
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

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 certificate chain selfsignx
Device(config)# certificate self-signed 01
[Certificate data omitted]

Device(config)# crypto pki certificate chain ccm1
Device(config)# certificate ca 55C2FCBF8AC552B7C6CED497D4AD33F8
[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 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
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 8443
Device(config-phone-proxy)# service-map server-addr ipv4 10.50.209.215 port 8080 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 983
CHAPTER 75

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 983
- Restrictions, on page 984
- Information About SIP TLS Support on CUBE, on page 985
- How to Configure SIP TLS Support on CUBE, on page 986
- SIP TLS Configuration Examples, on page 995

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 Name Indication (SNI)</td>
<td>Cisco IOS XE Amsterdam 17.3.1a</td>
<td>Support for Server Name Indication (SNI), a TLS extension that allows a TLS client to indicate the name of the server that it is trying connect during the initial TLS handshake process.</td>
</tr>
<tr>
<td>Command—voice class tls-profile tag</td>
<td>Cisco IOS XE Amsterdam 17.3.1a</td>
<td>Support for voice class TLS profile configuration. The tag associates voice class TLS profile configuration to the command crypto signaling.</td>
</tr>
<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>Feature Name</td>
<td>Releases</td>
<td>Feature Information</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Elliptical Curve Ciphers</td>
<td>Cisco IOS XE</td>
<td>Support for configuring Elliptic Curve for a TLS session.</td>
</tr>
<tr>
<td></td>
<td>Gibraltar Release</td>
<td>16.10.1a</td>
</tr>
<tr>
<td>Change in the default SIP TLS</td>
<td>Cisco IOS XE</td>
<td>Behavior of the command <code>transport tcp tls</code> is modified.</td>
</tr>
<tr>
<td>Versions support on CUBE</td>
<td>16.9.1</td>
<td>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</td>
<td>Cisco IOS XE 15.6(1)T</td>
<td>Support is provided for SIP-to-SIP calls with Transport Layer Security (TLS) version 1.2.</td>
</tr>
<tr>
<td>CUBE</td>
<td>Cisco IOS XE</td>
<td>The following cipher suites are introduced for release Cisco IOS 15.6(1)T:</td>
</tr>
<tr>
<td></td>
<td>3.17S</td>
<td>- TLS_DHE_RSA_WITH_AES_128_CBC_SHA1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</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.</td>
</tr>
<tr>
<td>CUBE</td>
<td></td>
<td>The following cipher suites are introduced for release Cisco IOS 12.4(6)T:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SSL_RSA_WITH_RC4_128_MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TLS_RSA_WITH_AES_128_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.

![Figure 95: CUBE with SIP TLS connections](image)

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. The CA server 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

Before 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
Following ciphersuites are added in the Cisco IOS XE Amsterdam 17.3.1a release:

- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA

Use the `srtp pass-thru` command to globally enable the transparent passthrough of all (supported and unsupported) crypto suites. If SRTP pass-thru feature is enabled, media interworking features such as transcoding, transrating, DTMF interworking, and so on, 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

**Note**

From IOS XE Release 16.6.1 onwards, the key-pair information is encrypted in all the router platforms.

When you downgrade the router 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 ec keysize {256 | 384}[label label][ec key-label] ! Applicable only for TLS version 1.2.
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 enroll name or crypto pki enroll name`
16. `crypto ca authenticate name or crypto pki authenticate name`
17. `crypto pki import <trustpoint> certificate`
18. `sip-ua`
19. `transport tcp tls [v1.0 | v1.1 | v1.2 ]`
20. `crypto signaling [remote-addr ip address subnet mask | default] [ tls-profile tag | trustpoint trustpoint-name [client-vtp trustpoint-name] [ecdsa-cipher [curve-size 384] | strict-cipher]]`
   `cn-san-validate [server [client-vtp trustpoint-name] [ecdsa-cipher [curve-size 384] | strict-cipher]]`  
   ECDSA ciphers are not supported on TLS version 1.0.
21. `voice service {pots | voatm | vofr | voip}`
22. `transport tcp tls`
23. `url {sip | sips | tel}`
24. `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>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> crypto key generate rsa [general-keys</td>
<td>usage-keys] label key-label [exportable</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config)# crypto key generate rsa general-keys label kp1 exportable</td>
<td></td>
</tr>
</tbody>
</table>

- **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 350–2048. 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 (:).
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>crypto key generate ec keysize {256</td>
<td>384}</td>
</tr>
</tbody>
</table>

**Example:**

Router(config)# crypto key generate ec keysize 384 <cr>

<table>
<thead>
<tr>
<th><strong>Step 5</strong></th>
<th>Declares the trustpoint that your router should use. Argument is as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required: crypto pki trustpoint name</td>
<td>name - Creates a name for the trustpoint that you created.</td>
</tr>
<tr>
<td>Example:</td>
<td>cube1 - Represents the trustpoint name that the user specifies.</td>
</tr>
</tbody>
</table>

Router(config)# crypto pki trustpoint cube1

<table>
<thead>
<tr>
<th><strong>Step 6</strong></th>
<th>Specifies which key pair to associate with the certificate. Arguments are as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsakeypair key-label [key-size [encryption-key-size]]</td>
<td>key-label - Name of the key pair, which is generated during enrollment if it does not exist or if the auto-enroll regenerate command is configured.</td>
</tr>
<tr>
<td>Example:</td>
<td>key-size - (Optional) Size of the desired RSA key. If not specified, the existing key size is used.</td>
</tr>
<tr>
<td></td>
<td>encryption-key-size - (Optional) Size of the second key, which is used to request separate encryption, signature keys, and certificates.</td>
</tr>
</tbody>
</table>

Router(config)# rsakeypair kp1

<table>
<thead>
<tr>
<th><strong>Step 7</strong></th>
<th>Generates EC keys for ECDSA cipher suites.</th>
</tr>
</thead>
<tbody>
<tr>
<td>eckeypair keyname ! Applicable only for TLS version 1.2.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>

Router(config)# eckeypair mykey

<table>
<thead>
<tr>
<th><strong>Step 8</strong></th>
<th>Specifies whether the router serial number should be included in the certificate request. Keyword is as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial-number [none]</td>
<td>none - (Optional) Specifies that a serial number will not be included in the certificate request.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>

Router(ca-trustpoint)# serial-number

<table>
<thead>
<tr>
<th><strong>Step 9</strong></th>
<th>Specifies a dotted IP address or an interface that will be included as &quot;unstructuredAddress&quot; in the certificate request. Arguments and keyword are as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip-address {ip-address</td>
<td>interface</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>

Router(ca-trustpoint)# ip-address 172.18.197.154
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• interface—Specifies an interface, from which the router can get an IP address, that will be included as &quot;unstructureAddress&quot; in the certificate request.</td>
<td></td>
</tr>
<tr>
<td>• none—Specifies that an IP address is not to be included in the certificate request.</td>
<td></td>
</tr>
</tbody>
</table>

**Step 10**

**subject-name** [x.500-name]

Example:

Router (ca-trustpoint)# subject-name
CN=172.18.197.154

Similarly specifies the subject name in the certificate request. Argument is as follows:

- **x.500-name**—(Optional) Specifies the subject name that is used in the certificate request.

**Step 11**

**enrollment** [mode][retry period minutes][retry count number]url url[.pem]

Example:

Router (ca-trustpoint)# enrollment url
http://172.18.193.103

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 waits 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 resends 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 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.

**Step 12**

**crl optional** or revocation-check method1 [method2 [method3]]

Example:

Router (ca-trustpoint)# crl optional
or
Router (ca-trustpoint)# revocation-check none

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.

Available methods are as follows:

- **crl**—Certificate checking is performed by a certificate revocation list (CRL). This is the default behavior.

- **none**—Certificate checking is not required.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <code>ocsp</code> — Certificate checking is performed by an online certificate status protocol (OCSP).</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>(Optional) Specifies the revocation password for the certificate. Argument is as follows:</td>
</tr>
<tr>
<td><code>password string</code></td>
<td><strong>• string</strong> — Name of the password</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router((ca)-trustpoint)# password password</td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td>Exists the current mode.</td>
</tr>
<tr>
<td><code>exit</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td>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:</td>
</tr>
<tr>
<td><code>crypto ca enroll name</code> or <code>crypto pki enroll name</code></td>
<td><strong>• name</strong> — Specifies the name of the CA. Use the same name when you declared the CA using the <code>crypto pki trustpoint</code> command.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config)# crypto ca name cubel</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>Router(config)# crypto pki name cubel</td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td>Authenticates the CA (by getting the certificate of the CA). Argument is as follows:</td>
</tr>
</tbody>
</table>
| `crypto ca authenticate name` or `crypto pki authenticate name` | **• name** — Specifies the name of the CA. This is the same name that is 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). |
| **Example:**                          |                                                                         |
| Router(config)# crypto ca authenticate cubel |                                                                         |
| or                                    |                                                                         |
| Router(config)# crypto pki authenticate cubel |                                                                         |
| **Step 17**                           | Imports the certificate given by the CA.                                |
| `crypto pki import <trustpoint> certificate` |                                                             |
| **Step 18**                           | Enters SIP user-agent configuration mode.                               |
| `sip-ua`                              |                                                                         |
| **Example:**                          |                                                                         |
| Router(config)# sip-ua                |                                                                         |
| **Step 19**                           | Configures the specified TLS version.                                   |
| `transport tcp tls [v1.0 | v1.1 | v1.2 ]` |                                                                         |
| **Example:**                          |                                                                         |
**Command or Action**

Router(config-sip-ua)# transport tcp tls v1.2

**Purpose**

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.

**Note**

Configure the SIP gateway to use its trustpoint when it establishes or accepts TLS connection with a remote device with an IP address.

The trustpoint label refers to the CUBE’s certificate that is generated with the Cisco IOS PKI commands as part of the enrollment process. strict-cipher means that the SIP TLS process uses only those cipher suites that are mandated by the SIP RFC. When you use the strict-cipher 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.

**Step 20**

```
crypto signaling {remote-addr ip address subnet mask | default} [tls-profile tag | trustpoint trustpoint-name [client-vtp trustpoint-name | {ecdsa-cipher [curve-size 384] | strict-cipher}]]
  cn-san-validate {server [client-vtp trustpoint-name | {ecdsa-cipher [curve-size 384] | strict-cipher}]}
```

ECDSA ciphers are not supported on TLS version 1.0.

**Example:**

Router(config-sip-ua)# crypto signaling default trustpoint cube1

**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
  - 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,
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

- **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, CUBE validates the domain name that is configured as destination against the CN/SAN fields in the certificate provided by server. The TLS connection is established only if the domain name that is configured as destination, matches with one of the domain names in the CN/SAN fields of the server certificate that is configured. Once you configure `cn-san-validate {server}`, validation of the server identity happens for every new TLS connection.

- The keyword **tls-profile tag** associates all the voice class configurations that are made through the command `voice class tls-profile tag`. In addition to all the TLS crypto configuration options available under the command `crypto signaling`, the `voice class tls-profile tag` command has a keyword **sni send**.

  **sni send** enables Server Name Indication (SNI), a TLS extension that allows a TLS client to indicate the name of the server that it is trying connect during the initial TLS handshake process. Only the fully qualified DNS hostname of the server is sent in the client hello. SNI does not support IPv4 and IPv6 addresses in the client hello extension. After receiving a “hello” with the server name from the TLS client, the server uses appropriate certificate in the subsequent TLS handshake process. SNI is supported from TLS 1.2.

  For more information on associating `voice class tls-profile tag` command to `crypto signaling` command, see `crypto signaling` and `voice class tls-profile`. 
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 21</strong></td>
<td>voice service {pots</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config)# voice service voip</td>
</tr>
<tr>
<td></td>
<td>Specifies a voice encapsulation type and enters voice service VoIP configuration mode.</td>
</tr>
<tr>
<td><strong>Step 22</strong></td>
<td>transport tcp tls</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-voi-sip)# transport tcp tls</td>
</tr>
<tr>
<td></td>
<td>Enters this command in SIP configuration mode to enable the TLS port on TCP 5061 to listen.</td>
</tr>
<tr>
<td><strong>Step 23</strong></td>
<td>url {sip</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-serv-sip)# url sips</td>
</tr>
</tbody>
</table>
| | Configures URLs to either the SIP, SIPS, or TEL format for your VoIP SIP calls. Keywords are as follows:  
  - **sip**—Generate URLs in SIP format for VoIP calls. This is the default.  
  - **sips**—Generate URLs in SIPS format for VoIP calls.  
  - **tel**—Generate URLs in TEL format for VoIP calls. |
| **Note** | This SIP gateway is now configured to use TLS with endpoints sharing the same CA. |
| **Step 24** | end |
| **Example:** | | Ends the current mode. |
Verifying SIP TLS Configuration

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:

**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

<table>
<thead>
<tr>
<th>Conn-Id</th>
<th>Local-Address</th>
<th>TLS-Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>5061</td>
<td></td>
<td>TLSv1.2</td>
</tr>
<tr>
<td>36289</td>
<td></td>
<td>TLSv1.2</td>
</tr>
</tbody>
</table>

Cipher Curve

| ECDHE-ECDSA-AES256-GCM-SHA384 | F-384 |
| ECDHE-ECDSA-AES256-GCM-SHA384 | F-384 |

------------- SIP Transport Layer Listen Sockets -------------
Conn-Id Local-Address
0 [0.0.0.0]:5061:

Alternatively, the debug ccsp 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:7778@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-BFAAC35@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, 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
c=IN IP4 1.1.1.1
t=0 0
m=audio 18378 RTP/AVP 0 19
c=IN IP4 1.1.1.1
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
a=ptime:20

SIP TLS Configuration Examples

Example: SIP TLS Configuration

show running-config
Building configuration...
Current configuration : 10894 bytes
!
Last configuration change at 23:19:20 IST Wed Aug 19 2015
NVRAM config last updated at 20:25:45 IST Tue Aug 18 2015
!
version 15.6
service timestamps debug datetime msec localtime show-timezone
service timestamps log datetime msec localtime show-timezone
no service password-encryption
!
hostname CUBE
!
boot-start-marker
boot system flash:ctestimg
boot-end-marker
!
aqm-register-fnf
!
logging queue-limit 1000
logging buffered 9999999
no logging rate-limit
no logging console
!
no 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
Security
Example: SIP TLS Configuration

enrollment terminal
revocation-check none
!
!
crypto pki certificate chain
certificate 07
30820248 308201CD A0030201
30100603 5504030C 09706C75
310E300C 06035504 0A0C0543
16060355 04070C0F 74757269
38313235 3431345A 170D3136
0403130A 706C7574 6F646F64
656E7472 616C6973 65645F72
04002203 62000446 4E28C72B
F21825DF 0621A34D 3119E23F
CDA24A8B 5318F11B EBA618A1
5BDCDBD0 BC6B1FA3 818B3081
A0EC7EA1 A47F3AEB 6B16D388
BB00C516 ACEEAF72 FDB6198D
082B0601 05050730 01861A68
38303830 2F300E06 03551D0F
03036900 30660231 00977017
A3316D34 43E9C928 8FBF42A4
48AC2C81 884E8C8D 421A9B11
F5F7FDFA 53052711 E493A754
quit
certificate ca 00
3082023B 308201C1 A0030201
30100603 5504030C 09706C75
310E300C 06035504 0A0C0543
16060355 04070C0F 74757269
36303934 3730345A 170D3136
04030C09 706C7574 6F31646F
0355040A 0C054369 73636F31
070C0F74 7572696E 672D6578
052B8104 00220362 0004D2EE
241719ED 4D733BDC 061F92D0
71E3151B 293971A2 94B14074
B303068C 66899963 763CA35D
551D0F04 04030201 06301D06
EEAF72FD B6198D6C 301F0603
ACEEAF72 FDB6198D 6C300A06
9AF19940 B0898320 AE96D8CB
3B719834 A99AF174 6EF9E35D
6F5F4BCE 3B0313D1 E3B76598
quit
crypto pki certificate chain
certificate self-signed 01
30820235 3082019E A0030201
36311330 11060355 0403130A
F70D0109 02161063 656E7472
38313434 3234355A 170D3230
0403130A 706C7574 6F646F64
656E7472 616C6973 65645F72
0003818D 00308189 02818100
9138C777 7EAEA997 5EA3B937
782B099F 8F701221 A11C8B81
EBEBBD78 6597DDFB 700D3DA6
BEC9B76E 38C442D9 DB9C2293
05300301 01FF301F 0603551D
C30E2EF0 2C044430 1D060355
0E2EF02C 0444300D 06092A86
B03AE032 ABBD80B7 7418402B
C878C693 A491B433 37BA1170

ecdsacert1
02020107
746F3164
6973636F
6E672D65
30383137
736E311F
74723230
9A66C344
DB2A5ACE
4BE2C66A
88301D06
301F0603
6C303606
7474703A
0101FF04
6DCE34A4
84583017
3177582D

300A0608
6F64310C
310B3009
7865632D
31323534
301D0609
76301006
7D6EB2C7
1C744F17
27857932
03551D0E
551D2304
082B0601
2F2F2031
04030207
3B0F78CF
856D513D
DBE9973F

2A8648CE
300A0603
06035504
6C6E7830
31345A30
2A864886
072A8648
51CE17F3
789450F5
48485192
04160414
18301680
05050701
37332E33
80300A06
2C69C7AD
C5B66547
D67EA687

3D040303
55040B0C
06130242
1E170D31
36311330
F70D0109
CE3D0201
D125D12B
1071E504
74923495
BE2A3FDE
1460CAB8
01042A30
392E3537
082A8648
2123B5F9
1E023100
0CF08620

30593112
03544143
45311830
35303831
11060355
02161063
06052B81
7043A98B
DA7DC56C
E762E3B4
F3CDA85E
AF1250CF
28302606
2E38333A
CE3D0403
C10999E7
AEF7EFE8
375628D0

02020100
746F3164
6973636F
6E672D65
30383035
64310C30
0B300906
65632D6C
C8BE0015
36899A05
893BB537
305B300C
03551D0E
551D2304
082A8648
52FB3B75
0231008E
0D9E43EB

300A0608
6F64310C
310B3009
7865632D
30393437
0A060355
03550406
6E783076
AE8DF590
71E515B9
09D4BC9C
0603551D
04160414
18301680
CE3D0403
CE599444
337B0A8F
B11EFCF5

2A8648CE
300A0603
06035504
6C6E7830
30345A30
040B0C03
13024245
30100607
3F0A8955
A86306B4
BF57E3DC
13040530
60CAB8AF
1460CAB8
03036800
EA3DBAC1
864F32D4
8C76318C

3D040303
55040B0C
06130242
1E170D31
59311230
54414331
31183016
2A8648CE
C1B3D80F
6DC49D66
AD5FA66B
030101FF
1250CFBB
AF1250CF
30650230
F4517F13
C85CC7CC
0F835560

30593112
03544143
45311830
35303830
10060355
0E300C06
06035504
3D020106
99B3CE51
87843054
590DA475
300B0603
00C516AC
BB00C516
390E60B9
B96C26CB
585FCD8B
0CD149

300D0609
6F646F64
65645F72
30303030
301D0609
819F300D
9A599812
2A022ADA
DBC1FFCB
568670EA
01A35330
1680141D
04141D59
01010505
341CB523
9BC22CA8

2A864886
736E311F
74723230
30305A30
2A864886
06092A86
F5CC7347
7D29C4C6
809FC745
1DEB6619
51300F06
5971FE06
71FE061D
00038181
7077570C
94E25907

F70D0101
301D0609
1E170D31
36311330
F70D0109
4886F70D
1F9E223C
DC9B01EB
3FC6BD87
2ED5EC71
03551D13
1D126AA3
126AA367
0033BC90
CD495BE3
F91C7B75

05050030
2A864886
35303831
11060355
02161063
01010105
E395073B
0E9E77DF
725F6B66
99B2612A
0101FF04
6767DBA6
67DBA6C3
8AF1DFBD
47A1B35B
450B0DE1

selfsign
02020101
706C7574
616C6973
30313031
736E311F
74723230
A01400F8
4B858866
D82AB7F3
73C52342
02030100
23041830
1D0E0416
4886F70D
0BFB9E0B
45F1DF85

Cisco Unified Border Element Configuration Guide Through Cisco IOS XE 17.5
997


Example: SIP TLS Configuration

76AC2C6B 5517F42A 46260F76 4A1DF81F 733A14FE 918F43F4 9BABAA49 227B5014 986044E7 8E98E373 7A361696 F0AD3ACC C9B101DF 2F80CCF7 E3
quit
crypto pki certificate chain ccm155RSA
certificate ca 4E23E56C7339CC679FD444D77F 986044E7 8E98E373 7A361696 F0AD3ACC C9B101DF 2F80CCF7 E3
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 sn FGL1538116L
hw-module pvdm 0/0
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 cins 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

scpp local GigabitEthernet0/1
scpp ccm 10.64.86.154 identifier 1 version 7.0
Example: SIP TLS Configuration

```text
! 
! 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
```
PART XXI

Voice Quality in CUBE

• CUBE Call Quality Statistics Enhancement, on page 1003
• Voice Quality Monitoring, on page 1009
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 1003
- Restrictions for Call Quality Statistics Enhancement, on page 1004
- Information About Call Quality Statistics Enhancement, on page 1004
- How to Configure Call Quality Parameters, on page 1005
- Configuration Example for Call Quality Statistics, on page 1007

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 100: Feature Information for Call Quality Statistics Enhancement

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
</table>
| Call Quality Statistics Enhancement | Cisco IOS XE 3.14S | 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. |

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.

# 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. `rtp 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> 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 global VoIP configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> call-quality</td>
<td>Enters call quality configuration mode; this is the global call quality of service setup.</td>
</tr>
<tr>
<td>Example: Device(conf-voi-serv)# call-quality</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> max-dropout number-of-packets</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>Example: Device(conf-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

```plaintext
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
```
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 1009
- Prerequisites for Voice Quality Monitoring, on page 1010
- Restrictions for Voice Quality Monitoring and Voice Quality Statistics, on page 1011
- Information About Voice Quality Monitoring, on page 1011
- How to Configure Voice Quality Monitoring, on page 1012
- Configuration Examples for Voice Quality Monitoring, on page 1016

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 101: Feature Information for Voice Quality Monitoring and Voice Quality Statistics |
|----------------------------------------|-----------------|-----------------|
| Feature Name                           | Releases        | Feature Information                                      |
| Voice Quality Statistics               | Cisco IOS XE Everest 16.5.1b | 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. |
Feature Information

<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>

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 number-of-packets`—Configures the acceptable out of sequence future packets to drop. The range is 2–2000 packets. The default value is 100.
  - `max-reorder number-of-packets`—Configures the acceptable out of sequence late packets. The range is 2–2000 packets. The default value is 100.

**Note**

The values of `max-dropout` and `max-reorder` must be configured based on the network loss and network latency. In a lossy or high latency network, it’s recommended to configure higher values. In a loss less or low latency network, lower values are fine. Packet loss and packet reorder are calculated based on RFC3550.

**Sample Configuration**

The following is a sample for the recommended voice quality monitoring configuration:

callmonitor
rtcp all-pass-through
media statistics
media bulk-stats
call-quality
  max-dropout 100
  max-reorder 100
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.
- MOS value shows 0 if endpoint does not send RTCP packets.
- The voice quality statistics covers only the TDM-IP call. The implementation focuses on filling the following statistics based on query response from DSP for TDM-SIP and TDM-H323 call:
  - RoundTripDelay
  - GapFillWithSilence
  - GapFillWithPrediction
  - GapFillWithInterpolation
  - GapFillWithRedundancy
  - HiWaterPlayoutDelay
  - LoWaterPlayoutDelay
  - PlayDelayJitter

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 exists 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:

```
<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:

<table>
<thead>
<tr>
<th>IOS VQM, Voice/Audio Description Quality Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOS-Con</td>
<td>The conversational quality MOS. Conversational quality indicates the impact of the quality of the transmission on the dynamics of conversational exchanges between two parties; such metrics take into account delay, echo, and recency.</td>
</tr>
<tr>
<td>round-trip-delay</td>
<td>The instantaneous round-trip delay. This may be obtained from the RTCP SR reports.</td>
</tr>
<tr>
<td>receive-delay</td>
<td>The minimum delay that will be applied to the packets received when using an adaptive jitter buffer.</td>
</tr>
<tr>
<td>voice-quality-total-packet-loss</td>
<td>The total number of packets lost by the jitter buffer in the RTP stream.</td>
</tr>
</tbody>
</table>

### 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>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><em>Example:</em></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><em>Example:</em></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><em>Example:</em></td>
<td></td>
</tr>
<tr>
<td>Device(config)# voice service voip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> media statistics</td>
<td>Enables media statistics to estimate the values of packet loss, jitter, and Round Trip Time (RTT) statistics.</td>
</tr>
<tr>
<td><em>Example:</em></td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# media statistics</td>
<td>• The statistics are displayed using the show voice history and show call active voice commands.</td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td><em>Example:</em></td>
<td></td>
</tr>
<tr>
<td>Device(conf-voi-serv)# end</td>
<td></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**  
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
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:  
cmqVoIPCallActiveRxPred107RMsConv.8520964.1 = 423 (MC)
```
For more information on the SNMP MIB "cmqVoIPCallActiveRxBalPred107RMosConv", 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 102: Router Output Definitions for the show call active voice stats command**

<table>
<thead>
<tr>
<th>Type</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP/RF: R-Factor Statistics</td>
<td>ML</td>
<td>R-factor MOS listening quality objective</td>
</tr>
<tr>
<td>(G.107 MOS)</td>
<td>MC</td>
<td>R-factor MOS-CQE</td>
</tr>
<tr>
<td></td>
<td>R1</td>
<td>R-factor for LQ profile 1</td>
</tr>
<tr>
<td></td>
<td>R2</td>
<td>R-factor for LQ</td>
</tr>
<tr>
<td></td>
<td>IF</td>
<td>Effective codec impairment (IeEff)</td>
</tr>
<tr>
<td></td>
<td>ID</td>
<td>Delay factors</td>
</tr>
<tr>
<td></td>
<td>IE</td>
<td>Codec baseline score (Ie)</td>
</tr>
<tr>
<td></td>
<td>BL</td>
<td>Codec baseline (Bpl)</td>
</tr>
<tr>
<td></td>
<td>R0</td>
<td>Nominal value for R0 (default)</td>
</tr>
<tr>
<td></td>
<td>VR</td>
<td>R-Factor version</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(conf-voi-serv)# media statistics
Device(conf-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

PART XXII

Smart Licensing

• CUBE Smart Licensing, on page 1019
CUBE Smart Licensing

Cisco Smart Licensing using Policy is a software licensing model that provides visibility of ownership and usage through the Cisco Smart Software Manager (CSSM) portal. CSSM is a central license repository that manages licenses across all Cisco products that you own, including CUBE. Devices send license usage to CSSM either directly or use an on-premises application. Your Smart Account Administrator controls your access to CSSM. Use your Cisco credentials to access the CSSM portal through http://software.cisco.com.

Smart Licensing applies to all platform technology (UCK9, Security, DNA) and CUBE feature licenses that the platform uses.

CSSM shows license usage across all devices that are registered to a virtual account. A Virtual Account License Inventory displays the quantity of licenses that are purchased, those licenses in use, and a balance. An Insufficient Licenses alert is displayed if the license balance is below 0.

For example, consider a smart account in CSSM with 50 CUBE trunk session licenses. If you have a single registered CUBE router using 20 trunk sessions, the CSSM licenses page shows Purchased as 50, In Use as 20, and Balance as 30.

For more information on Smart Software Manager, see the Cisco Smart Software Manager User Guide.

- Smart License Operation, on page 1019
- Smart Software Licensing Task Flow for CUBE, on page 1021
- Verify Smart Licensing Operation for CUBE, on page 1023
- CUBE High Availability Configurations, on page 1027
- Syslog Messages, on page 1032

Smart License Operation

Smart Licensing using Policy introduces a new paradigm for tracking license usage across your business. In earlier releases, license authorization was forward looking, binding licenses to a device until the next authorization request. Actual license use during the proceeding reporting period is now sent to CSSM, allowing you to plan ongoing license requirements based on historical usage data.

Each time a change in license usage is detected, the account policy defines how soon the change should be reported to CSSM. Typically, this means that CUBE must send a report at least every 90 days, although it is recommended that reports are sent more frequently. CSSM acknowledges each submitted Resource Utilization Monitoring (RUM) report to ensure that the use is recorded reliably. If the router does not receive an acknowledgment within the minimum reporting period, call processing may be disabled. Call processing is resumed when a valid acknowledgment is received.
Submit the reports to CSSM directly or through a Smart Software Manager On-Prem server. The Cisco Smart Licensing Utility (CSLU) application can also collect usage reports, providing more flexibility in managing your license usage. When a device is not able to communicate directly with a licensing server, a signed usage report can be generated and manually uploaded to CSSM. The acknowledgment that is generated by CSSM must be uploaded to the device within the license reporting policy period to ensure continued use.

**Note**

CUBE_T_VGW records TDM license count when there is a TDM-SIP call. The `show license all` command displays the number of licenses that are used and is an expected behaviour. The CUBE_T_VGW is not displayed when there is no TDM-SIP call, and the CUBE does not act as a gateway. The TDM-SIP call service continues uninterrupted even if the device is not registered to CSSM or RUM acknowledgement is not received from the CSSM server.

**Note**

Cisco Smart Software Manager On-Prem Version 8 Release 202102 or later is required for any device using SLP. Refer to the SLP feature documentation for further information (Smart Licensing Using Policy for Cisco Enterprise Routing Platforms).

**Warning**

When using any of the following Smart Licensing using Policy releases, CUBE shuts down if the router does not receive a report acknowledgment from CSSM before the acknowledgment deadline set by the account policy: 17.3.2, 17.3.3, 17.3.4a, 17.6.1a, or any 17.4 or 17.5 release. CUBE does not shut down in this way with later releases.

License usage is calculated dynamically in the same way as earlier releases, with measurements recorded periodically based on the periodicity timer. Measurements are stored locally until they are submitted to CSSM. This historical usage reporting allows for more accurate aggregation of use across multiple devices over time. The minimum value for the periodicity timer interval is 8 hours.

For example, consider a measurement periodicity of 8 hours and a reporting policy interval of 30 days. Call load is measured every second. The average of the top three readings during each minute is recorded to mitigate anomalies. Similarly, the average of the top three minute measurements is used to log usage over an hour. After 8 hours, the maximum hourly measurement is used to record usage locally for that period. The 90 measurements that are recorded over a 30 day reporting period are sent to CSSM or a CSLU in a single report.

If the peak license usage for the current period is different by more or less than 25% of the previously reported value, it is reported and the periodicity interval is reset. Use of CUBE Standard Trunk and CUBE Enhanced Trunk licenses are monitored separately.

**Note**

Smart License Reservation (SLR) for CUBE licenses is not compatible with SLP. Even if a reservation is in place when upgrading, license use reporting will still be required in accordance with the device policy.

**Note**

From Cisco IOS XE Bengaluru 17.6.1a onwards, calls that are forked using WebSockets are marked as consuming an Enhanced session license.
Smart Software Licensing Task Flow for CUBE

IOS XE license use reports may be pushed to, or pulled by a licensing server. Refer to the workflows in Smart Licensing Using Policy for Cisco Enterprise Routing Platforms for more details.

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 is 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  license smart transport smart
Example:
cube(config)# license smart transport smart
Smart transport is the preferred method for sending usage reports.

Step 5  license smart proxy address hostname
Example:
cube(config)# license smart proxy address proxy.cisco.com
If necessary, configure a proxy-server for the platform when a direct HTTP connection to CSSM is not permitted.
Step 6  license smart proxy port  

Example:

```
cube(config)# license smart proxy port 80
```

If using an internet proxy, configure a proxy-server port number.

Step 7  license smart url smart

Example:

```
cube(config)# license smart url smart
```

Use the default URL for sending reports to CSSM.

---

**Associate the Host Platform with CSSM**

From Cisco IOS XE Everest 16.11.1a to Cisco IOS XE Amsterdam 17.3.1a, you must register the host platform to either CSSM or SSM On-Prem to report license usage. From Cisco IOS XE Amsterdam 17.3.2 onwards, license use must be reported to CSSM or SSM On-Prem in accordance with the Smart Account reporting policy.

**Before you begin**

1. Obtain the registration ID token from your Smart Account.
2. Configure Smart Licensing transport settings.

From Cisco IOS XE 16.10.1 through Cisco IOS XE Amsterdam 17.3.1a, use the following command for registering the CUBE with the CSSM or satellite.

```
license smart register id_token id_token
```

Example:

```
Router# license smart register id_token XXXXXXXXXtNvhaUZlRHorQjJERT0%3D
```

Use the following command to register the CUBE platform with CSSM.

```
license smart trust id_token id_token...local [force]
```

Example:

```
Router# license smart trust id_token ZDEwZDFiODktNWF............
```

---

**Configure CUBE Licensed Features**

Step 1  voice service voip

Example:

```
Device(config)# voice service voip
```
Enters global VoIP configuration mode.

**Step 2**  
\`\`\texttt{mode border-element [ license periodicity \{hours \(<8-23>\) | days \(<1-30>\) \}]}\`\`  
The periodicity keyword configures the CUBE license usage measurement interval. If you do not configure the periodicity keyword, license usage is measured once every 7 days.

## Verify Smart Licensing Operation for CUBE

This section shows CUBE license usage status and license usage history.

- **show cube status**—Displays CUBE license status.

  ```
cube#show cube status  
CUBE-Version : 14.1  
SW-Version : 17.3.2, Platform CSR1000V  
HA-Type : none  
```

- **show license status**—Displays the license policy and reporting status.

  ```
cube#show license status  
Utility:  
  Status: DISABLED  
Data Privacy:  
  Sending Hostname: yes  
  Callhome hostname privacy: DISABLED  
  Smart Licensing hostname privacy: DISABLED  
  Version privacy: DISABLED  
Transport:  
  Type: Callhome  
Policy:  
  Policy in use: Merged from multiple sources.  
  Installed Time: Jan 01 05:30:00 1970 IST  
  Reporting ACK required: yes  
  Perpetual Attributes:  
    First report requirement (days): 365 (CISCO default)  
    Reporting frequency (days): 90 (CISCO default)  
    Report on change (days): 90 (Product default)  
  Subscription Attributes:  
    First report requirement (days): 90 (CISCO default)  
    Reporting frequency (days): 90 (CISCO default)  
    Report on change (days): 80 (Product default)  
  Enforced License Attributes:  
    First report requirement (days): 90 (Customer Policy)  
    Reporting frequency (days): 90 (Customer Policy)  
    Report on change (days): 80 (Customer Policy)  
  Export License Attributes:  
    First report requirement (days): 90 (Customer Policy)  
  ```

**Note**  
The acknowledgment deadline is presented in this output. Ensure that an acknowledgment is received before this time to ensure continued operation of the SIP service.
Reporting frequency (days): 90 (Customer Policy)
Report on change (days): 90 (Customer Policy)

Miscellaneous:
  Custom Id: <empty>

Usage Reporting:
  Last ACK received: <none>
  Next ACK deadline: May 26 08:24:45 2020 IST
  Reporting Interval: 30
  Next ACK push check: <none>
  Next report push: Jun 15 08:24:45 2020 IST
  Last report push: <none>
  Last report file write: <none>
  Last report pull: <none>

Trust Code Installed: <none>

- **show voice sip license stats**—Displays CUBE trunk license usage history.

License usage is recorded in tabular and graphical format for all the three types of trunk call count (Enhanced, Standard, and Aggregate). Usage is recorded based on the peak value of concurrent calls for a defined interval of time:

- **Seconds Table**—This table stores concurrent calls at every second for the last 60 seconds.

- **Minutes Table**—This table stores regularized peak value of concurrent calls at every minute for the last 60 minutes. Regularized peak for a minute is the average of top 3 peak values that occurs in a minute.

- **Hours Table**—This table stores peak value for each hour for the last 72 hours.

- **Days Table**—This table stores peak value for each day for the last 72 days.

The following example outputs are truncated to display 60-second and 60-minute tables only.

```
cube#show voice sip license stats table

02:50:16 PM Wednesday Nov 13 2019 UTC

CUBE Trunk License Usage (last 60 seconds)
Period | Average | Max
--- | --- | ---
1-5 | 0 | 0
6-10 | 0 | 0
11-15 | 0 | 0
16-20 | 0 | 0
21-25 | 0 | 0
26-30 | 0 | 0
31-35 | 0 | 0
36-40 | 0 | 0
41-45 | 0 | 0
46-50 | 0 | 0
51-55 | 0 | 0
56-60 | 0 | 0

CUBE Trunk License Usage (last 60 minutes)
Period | Average | Max
--- | --- | ---
1-5 | 0 | 0
```
cube#show voice sip license stats

11:01:01 AM Thursday Aug 29 2019 IST

+----------+----------+----------+----------+----------+----------+----------+----------+----------+----------+
|          | 0        | 5        | 0        | 5        | 0        | 5        | 0        | 5        | 0        |
+----------+----------+----------+----------+----------+----------+----------+----------+----------+----------+
| 6-10     | 0        | 0        | 0        |          |          |          |          |          |          |
| 11-15    | 0        | 0        | 0        |          |          |          |          |          |          |
| 16-20    | 0        | 0        | 0        |          |          |          |          |          |          |
| 21-25    | 0        | 0        | 0        |          |          |          |          |          |          |
| 26-30    | 0        | 0        | 0        |          |          |          |          |          |          |
| 31-35    | 0        | 0        | 0        |          |          |          |          |          |          |
| 36-40    | 0        | 0        | 0        |          |          |          |          |          |          |
| 41-45    | 0        | 0        | 0        |          |          |          |          |          |          |
| 46-50    | 324      | 900      |          |          |          |          |          |          |          |
| 51-55    | 343      | 899      |          |          |          |          |          |          |          |
| 56-60    | 292      | 600      |          |          |          |          |          |          |          |
| 6-10     | 0        | 0        | 0        |          |          |          |          |          |          |
+----------+----------+----------+----------+----------+----------+----------+----------+----------+----------+

CUBE Trunk License Usage (last 60 seconds)

• show voice sip license status—Displays the license status.
**Verify Smart Licensing Operation for CUBE**

cube#show voice sip license status

Host Name: cube
SIP service: Up
License request interval: 5 Minute(s)
Next request at: Nov 25 2019 14:50:44 IST
Recent request(s) for CUBE Standard Trunk

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Count</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov 25 2019 14:45:44 IST</td>
<td>10</td>
<td>Out of compliance</td>
</tr>
<tr>
<td>Nov 25 2019 14:40:44 IST</td>
<td>4</td>
<td>Authorized</td>
</tr>
<tr>
<td>Nov 25 2019 14:35:44 IST</td>
<td>2</td>
<td>Authorized</td>
</tr>
</tbody>
</table>

**show license usage**—Displays the license usage.

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 the license summary information.

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:

<table>
<thead>
<tr>
<th>License</th>
<th>Entitlement tag</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBE_T_STD</td>
<td>(CUBE_T_STD)</td>
<td>5</td>
<td>IN USE</td>
</tr>
<tr>
<td>uxk9</td>
<td>(ISR_4351_UnifiedCommun...)</td>
<td>1</td>
<td>IN USE</td>
</tr>
<tr>
<td>CUBE_T_VGW</td>
<td>(CUBE_T_VGW)</td>
<td>4</td>
<td>IN USE</td>
</tr>
</tbody>
</table>

The following commands are also available related to your Smart License:

  - **show license all**—Displays all the information that is related to licensing.
  - **show license tech support**—Displays the license technical support information.
- **show call-home smart-licensing** — Displays the destination URL that is configured.

- **debug license feature cube all**

  ```
  Request successful for license_type:<license_type_int> and count <usage_count>.
  ```

  Example:

  ```
  ```

  Request successful for license <license_type_string>

  Example:

  ```
  *May 18 10:15:45.181: //CUBE-SL/Info/cube_license_request: Request successful for license CUBE_T_STD
  ```

## CUBE High Availability Configurations

### Smart Licensing with CUBE Box-to-Box High Availability

Box-to-Box redundancy uses the Redundancy Group Infrastructure to form a high availability pair of platforms. For Smart License configurations on the high availability pair of platforms, see Smart Software Licensing Task Flow for CUBE, on page 1021. When reporting license usage, the Smart Agent includes details of its high availability group and, if it is in the active or standby state. These details allow the CSSM or satellite to group license requirements for the high availability pair.

Box-to-Box High Availability requires CUBE Trunk Redundant or Enhanced Session licenses. From Cisco IOS XE Amsterdam 17.2.1r onwards, license usage is based on dynamic call counting.

#### Before Failover

- Establish a trust relationship for both platforms in the high availability configuration with the same CSSM or satellite Smart Virtual Account.

- CSSM or satellite sets the reporting policy for each platform.

- Only the active platform submits license usage reports to CSSM.

#### After Failover

- The platform that switches to the active mode reports license usage to the CSSM.

- The new active platform starts a new license request interval timer. For example, if a periodicity of five days is configured and failover occurs after three days, the next request will occur five days later.
Verify Smart Licensing Operation for Box-to-Box High Availability

You can use all the commands that are given in the section Verify Smart Licensing Operation for CUBE, on page 1023 to verify the licensing status in High Availability mode.

The following commands reflect Box-to-Box High Availability licensing information.

- **show cube status** — Displays CUBE license capacity and the 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
  ```

  **Note** Effective from Cisco IOS XE Amsterdam 17.2.1r, Licensed-Capacity and blocked call information is no longer included in the output.

- **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
  Version: 1.0
  Status: AUTHORIZED
  Export status: NOT RESTRICTED
  ```

  ```
cube-2#show license usage
  CUBE_Trunk_Standard_Session (CUBE_T_RED):
  Description: Cisco Unified Border Element (CUBE) Trunk Redundant Session License
  Count: 10
  Version: 1.0
  Status: AUTHORIZED
  Export status: NOT RESTRICTED
  ```

- **show license summary** — Displays the license summary information.

  Following is the sample output from the active instance of CUBE.

  ```
Device_CUBE1# show license summary
License Usage:
License Entitlement tag Count Status
---------------------------------------
CUBE_Redundant_Session (CUBE_T_RED) 20 AUTHORIZED
```  

- **show license all** — Displays active and standby states.

  ```
c8kv#sh license all
  Smart Licensing Status
  ------------------------
  Smart Licensing is ENABLED
  ```

  ```
  Export Authorization Key:
  Features Authorized:
  <none>
  ```

  ```
  Utility:
  Status: DISABLED
  ```
Smart Licensing Using Policy:
Status: ENABLED

Data Privacy:
Sending Hostname: yes
  Callhome hostname privacy: DISABLED
  Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED

Transport:
  Type: Smart
  URL: https://smartreceiver-stage.cisco.com/licservice/license
  Proxy:
    Not Configured

Miscellaneous:
  Custom Id: <empty>

Policy:
  Policy in use: Installed On Apr 20 13:26:18 2021 UTC
  Policy name: SLE Policy
  Reporting ACK required: yes (Customer Policy)
  Unenforced/Non-Export Perpetual Attributes:
    First report requirement (days): 30 (Customer Policy)
    Reporting frequency (days): 60 (Customer Policy)
    Report on change (days): 60 (Customer Policy)
  Unenforced/Non-Export Subscription Attributes:
    First report requirement (days): 120 (Customer Policy)
    Reporting frequency (days): 150 (Customer Policy)
    Report on change (days): 120 (Customer Policy)
  Enforced (Perpetual/Subscription) License Attributes:
    First report requirement (days): 0 (CISCO default)
    Reporting frequency (days): 90 (Customer Policy)
    Report on change (days): 60 (Customer Policy)
  Export (Perpetual/Subscription) License Attributes:
    First report requirement (days): 0 (CISCO default)
    Reporting frequency (days): 30 (Customer Policy)
    Report on change (days): 30 (Customer Policy)

Usage Reporting:
  Last ACK received: Oct 08 13:56:07 2021 UTC
  Next ACK deadline: Dec 07 13:56:07 2021 UTC
  Reporting push interval: 1 days
  Next ACK push check: Oct 22 20:44:57 2021 UTC
  Next report push: Oct 23 16:24:52 2021 UTC
  Last report push: Oct 22 16:24:52 2021 UTC
  Last report file write: <none>

Trust Code Installed: Apr 20 13:26:18 2021 UTC

License Usage
---------------

network-advantage_1G (ESR_P_1G_A):
  Description: network-advantage_1G
  Count: 1
  Version: 1.0
  Status: IN USE
  Export status: NOT RESTRICTED
  Feature Name: network-advantage_1G
  Feature Description: network-advantage_1G
  Enforcement type: NOT ENFORCED
  License type: Perpetual
Smart Licensing with CUBE Inbox High Availability

You can configure a Cisco ASR 1000 Series 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 Smart License configuration, see Smart Software Licensing Task Flow for CUBE, on page 1021. Only the active Route Processor in the SSO configuration reports license usage, so CSSM reserves one set of licenses for the platform.

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.
For Smart License using Policy, the CSSM or satellite 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 Licensing Operation for Inbox High Availability

You can use all the commands that are given in the section Verify Smart Licensing Operation for CUBE, on page 1023 to verify the licensing status in the High Availability mode.

The following commands reflect Inbox High Availability (HA) licensing information:

- **show cube status**—Displays CUBE license capacity and High Availability mode.

**Note** Effective from Cisco IOS XE Amsterdam 17.2.1r, Licensed-Capacity and blocked call information is no longer included in the output.

```
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.

```
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
```

```
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:

<table>
<thead>
<tr>
<th>License Entitlement tag</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR_1000_AdvEnterprise</td>
<td>1</td>
<td>AUTHORIZED</td>
</tr>
<tr>
<td>CUBE_Standard_Session</td>
<td>10</td>
<td>AUTHORIZED</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>License Entitlement tag</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR_1000_AdvEnterprise</td>
<td>1</td>
<td>PENDING</td>
</tr>
</tbody>
</table>

**Syslog Messages**

- In B2BHA mode, syslog messages are generated by the active CUBE router and not the standby router.
The following is a syslog output for an active CUBE router in B2BHA mode:

%CUBE-5-LICENSE_INFO: Requesting for 3 CUBE Enhanced trunk licenses
PART XXIII

Serviceability

- VoIP Trace for CUBE, on page 1035
- Support for Session Identifier, on page 1041
VoIP Trace for CUBE

VoIP Trace is a Cisco Unified Border Element (CUBE) serviceability framework, which provides a binary trace facility for troubleshooting SIP call issues. The VoIP Trace framework records both successful and failed calls. All call trace data is stored in system memory. In addition, data for calls with IEC errors is also written to the logging location configured at the system level which includes logging to a buffer or a syslog server.

• VoIP Trace for CUBE, on page 1035
• Prerequisites for Voip Trace, on page 1036
• Benefits of VoIP Trace, on page 1036
• Guide to using VoIP Trace Framework, on page 1037
• RTP Port Clear, on page 1038
• Feature Information for VoIP Trace, on page 1039

The VoIP Trace feature is enabled by default and can be used to help troubleshoot issues, even in deployments with high call volumes.

You can configure VoIP Trace for CUBE using the `trace` configuration sub-mode under `voice service voip` configuration mode:

```
router (config)#voice service voip
router(conf-voi-serv)#trace
router(conf-serv-trace)#?
```

VoIP Trace submode commands:

- `default` Set a command to its defaults
- `exit` Exit from voice service voip trace mode
- `no` Negate a command or set its defaults
- `shutdown` Shut Voip Trace debugging
- `memory-limit` Set limit based on memory used

Within the VoIP Trace sub-mode (`conf-serv-trace`), you can configure the following CLI commands:

- `memory-limit {platform | memory}`
- `[no] shutdown`

VoIP Trace is used for event logging and debugging of VoIP calls. Using the VoIP Trace framework, the following information is recorded:

- SIP messages for SIP trunk to SIP trunk calls
• Events and API calls from the SIP layer to other layers in CUBE.

• SIP Errors

• Call Control (Unified Communication flows processed by CUBE)

• FSM (Finite State Machine) states and events

VoIP Trace monitors and logs SIP signalling and call events in memory as they occur. In the event that a call error is detected, or calls fail with 3xx, 4xx or 5xx cause codes, these event details are written to the logging buffer after the call clears.

Note: Traces for error calls are logged at the rate of up to five traces per second.

There’s a configurable memory limit allocated for storage of traces in a VoIP Trace framework for CUBE. The configurable maximum memory limit is either available platform memory or 1000 MB, whichever is lower. By default, VoIP Trace will use up to 10% of the total memory available to the IOS processor at the time of configuring the command. For example, if CUBE is used on a platform with 8GB of memory, VoIP Trace will use up to 800MB for trace data. Once the trace memory limit is reached, older traces are overwritten and will no longer be available.

You can configure the trace memory limit using the CLI command `memory-limit {platform | memory)` under `trace` configuration sub-mode:

Router(conf-serv-trace)#memory-limit ?

<10-1000> Specify maximum memory limit in MB
platform Use 10 percent of available memory

To display the traces for a call, use the following show command:

• `show voip trace {call-id identifier | session-id identifier | sip-call-id identifier | correlator identifier | all | cover-buffers | statistics [detail]}`

Prerequisites for VoIP Trace

Cisco IOS XE Amsterdam 17.3.2 or a later release supported by CUBE.

Benefits of VoIP Trace

The following are some of the benefits of VoIP Trace Serviceability framework:

• Enabled by default

• Minimal CLI configuration requirement

• Minimal processing impact

• Automatic call error identification and trace logging based on IEC Errors

• Request-based manual call identification and trace logging based on filters like call-ID, session-ID, and so on.
Guide to using VoIP Trace Framework

The following are some of the usage guidelines for the VoIP Trace Serviceability framework.

• Enable or disable your VoIP Trace serviceability framework using the following CLI commands:
  • Enable—Configure `trace` under `voice service voip` configuration mode to enable your VoIP Trace framework (`trace` is enabled by default).
  • Disable—Configure `shutdown` under `voip trace` configuration mode to disable your VoIP Trace framework. To enable VoIP Trace after it’s disabled, configure the CLI command `no shutdown`.

• If you configure `shutdown` the VoIP Trace Serviceability framework:
  • Stops tracing for active calls.
  • Deletes all existing traces in the system memory.

• Monitors calls received after enabling VoIP Trace.

• Traces stored in memory can be displayed using the show command `show voip trace {call-id identifier | session-id identifier | sip-call-id identifier | correlator identifier | all | cover-buffers | statistics [detail]}
  • The show command displays traces for both active and disconnected calls.
  • The show command displays information only for the SIP leg.
  • For media forking, VoIP Trace also displays information for forked legs.

• For the CLI command `memory-limit [platform | memory]`
  • Configure `memory-limit platform` to set 10% of the total memory available to the IOS processor at the time of configuring the command as VoIP Trace memory limit.
  
  • Configure `memory-limit memory` to set a custom VoIP Trace memory limit. Range is 10–1000 MB.
    • Configuration of custom memory-limit more than the available platform memory is not allowed. Configuration fails with an error message:

      ```
      router(config)#voice service voip
      router(conf-voi-serv)#trace
      router(conf-serv-trace)#memory-limit 800
      Error: Setting memory-limit more than available platform memory (732 MB) is not allowed.
      ```
    
    • Configuration of memory-limit more than the 10% of the available platform memory affects the system performance. Configuration is successful with a warning message:

      ```
      router(config)#voice service voip
      router(conf-voi-serv)#trace
      router(conf-serv-trace)#memory-limit 100
      Warning: Setting memory limit more than 10% of available platform memory (73 MB) will affect system performance.
      ```
• Reducing the memory-limit from an existing limit resets the VoIP Trace data. Take copy of the show voip trace statistics detail and show voip trace all output data before reducing the memory-limit.

  • A confirmation message is displayed when you reduce the memory-limit from an existing limit:

  Reducing the memory-limit clears all VoIP Trace statistics and data.
  If you wish to copy this data first, enter ‘no’ to cancel, otherwise enter ‘yes’ to proceed.

• Increasing the memory-limit does not impact the VoIP Trace data.

Note

• Unable to trace incoming calls if active calls exhaust the memory-limit.

• To change the Timestamps displayed in the VoIP Trace, configure the following:

  router(config)#monitor event-trace timestamps datetime ?
  localtime      Use local time zone for timestamps
  msec           Include milliseconds in timestamp
  show-timezone  Add time zone information to timestamp

RTP Port Clear

When establishing a call, CUBE allocates several VoIP RTP ports. These ports are based on the media that are negotiated for the session. RTP ports can be allocated from the following three different tables:

• Global port table

• Media IP address-based table

• Media VRF-based table

The table that is used for allocating RTP ports is based on CUBE feature configuration. Ports are allocated from the VRF table first (if available), and then from the media table. If neither of these tables are available, the global table allocates ports.

Use the show voip rtp stats command to display the ports allocated from the different tables. In the current behavior, this command displays ports that are allocated only from the global port table. From Cisco IOS XE Bengaluru 17.4.1a onwards, this command displays ports that are allocated from all the three tables.

Sometimes, RTP ports can remain assigned after a call ends. Use the clear voip rtp port command to release such hung ports.

The show voip rtp stats command displayed only the port values from the global table, even if the ports are allocated from all the tables. From Cisco IOS XE Bengaluru 17.4.1a onwards, this command displays details of allocated ports from all the three tables.

The clear voip rtp port table ID port number command releases the hung ports. Here, table ID is the identifier of the table from which the port number is released.

A unique identifier is generated and printed for each table, which serves as a reference to clear voip rtp port command.
Feature Information for VoIP Trace

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>
| VoIP Trace for CUBE Serviceability | Cisco IOS XE Amsterdam 17.3.2, Cisco IOS XE Bengaluru 17.4.1a | VoIP Trace is a Cisco Unified Border Element (CUBE) Serviceability framework for Event Logging and Debug Classification. The following are the commands that are introduced as part of this feature:  
  • trace  
  • memory-limit [platform | memory ]  
  • shutdown  
  • show voip trace {call-id identifier | session-id identifier | sip-call-id identifier | correlator identifier | all | cover-buffers | statistics [detail]} |
| RTP Port Clear                   | Cisco IOS XE Bengaluru 17.4.1a        | Sometimes, RTP ports can remain assigned after a call end. This feature enhancement releases such hung ports and makes available for other calls. This release of ports increases the efficiency of the device. The feature introduces the following commands:  
  show voip rtp stats - The enhanced command enables you to print details for in-use ports of other port ranges (along with global port range).  
  Cisco IOS Voice Command Reference - S commands  
  clear voip rtp port <table-id> <port-num> - Use this command to clear VoIP Real Time Protocol (RTP) which are leaked ports.  
  Cisco IOS Voice Command Reference - A through C |
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 1041
- Restrictions, on page 1042
- Information About Session Identifier, on page 1042
- Configuring Support for Session Identifier, on page 1043
- Troubleshooting Tips, on page 1043

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 103: 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                | 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 |
|                               | Cisco IOS XE Denali 16.3.1        |                                                                                                                                                     |

### 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 checkpointed 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 brief session-id  
  
• show call active video session-id  
  
• show call active video brief session-id  

*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=db248b6cbdc547bbcc6c6f6f6916eeb
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 db248b6cbdc547bbcc6c6f6f6916eeb; remote=4fd24d9121935531a7f8d750ad16e19
```

Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
...
... 
```
SessionIDLocaluuid=db248b6cbdc547bbcc6c6f6f6916eeb
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 db248b6cbdc547bbcc6c6f6f6916eeb
```

Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
...
... 
```
SessionIDLocaluuid=db248b6cbdc547bbcc6c6f6f6916eeb
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
...
...
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
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= 4fd24d9* 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 ^[0-9a-fA-F*];remote=[0-9a-fA-F*]+$.
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=0000000000000000000000000000000
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
```

```
Device# show call active voice session-id 0
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
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=db248b6cbdc547bbc6c6f0b6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
```

```
You can search for session identifier using the local UUID as shown below:

Device# show call active voice session-id d82c680a3eaecd5c29ac6c3ea225061
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0

VOIP:
ConnectionId{0x8CDAC180 0x10000 0x1B7 0x5B56400A}
IncomingConnectionId{0x8CDAC180 0x10000 0x1B7 0x5B56400A}
CallId=1022
GlobalCallId{0xC3DAB665 0x770C11E5 0x80318550 0x50000ED7}
SessionIDLocaluuid=d82c680a3eaecd5c29ac6c3ea225061
SessionIDRemoteuuid=6497636d0b747785241cf5a225064
CallReferenceId=0
CallServiceType=Unknown
RTP Loopback Call=FALSE
RemoteIPAddress=10.64.86.91
RemoteUDPPort=16614
RemoteSignallingIPAddress=10.64.86.91
```
From the above output, you get to know that 1022 (highlighted) is the call identifier associated with the local session identifier \texttt{d82c680a3eaeced5c29ac6ceeea225061}. 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
Call 2
SIP Call ID: C3DEFC15-770C11E5-80348550-5A000ED7@10.64.86.132
State of the call: STATE_ACTIVE (7)
Substate of the call: SUBSTATE_NONE (0)
Calling Number: 4443332212
Called Number: 4443332211
Called URI: sip:4443332211@10.64.86.91:5060
Bit Flags: 0xC04018 0x90000100 0x80080
CC Call ID: 1023
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: 1023
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]:16426
Media Dest IP Addr:Port: [10.127.17.188]:21978

Example for video Calls

The following session identifier is considered in the below example:
SessionIDLocaluuid=6f0a93a3a79451aeebe6d83f79a3359f
SessionIDRemoteuuid=a55b0f45861551b88f57d1fb5bb23f89

Note: 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*
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=6f0a93a3a79451aebbc683f79a3359f
SessionIDRemoteuuid=a55b0f4586155b88f57dfb5bb23f89
RemoteIPAddress=10.64.86.70
RemoteSignallingIPAddress=10.64.86.70
RemoteSignallingPort=5061
RemoteMediaIPAddress=10.64.86.70
RemoteMediaPort=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 1053
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 1054
- Supported Hardware and Software for Virtual CUBE, on page 1054
- Common Criteria Configuration on Cisco CSR 1000v, on page 1054
- FIPS Configuration on Cisco CSR 1000v, on page 1066
Feature Information for Common Criteria (CC) and the Federal Information Standards (FIPS) Compliance

The following table provides release information about the feature or features that are 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. You need not require an account on Cisco.com.

<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 1055
2. Configure Certificate Authority Server, on page 1056
3. Configure CSR Trustpoint, on page 1057
4. Configure Peer Trustpoint, on page 1058
5. Add Client Verification Trustpoint, on page 1059
6. Enforce Strict SRTP, on page 1060

**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.
Configure Certificate Authority Server

**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.
**Configure CSR 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 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.

---

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.

Step 3  crypto pki trustpoint name
Example:
Router(config)#crypto pki trustpoint xyzname
Declares the peer trustpoint with the name specified and enters trustpoint configuration mode.

Step 4  enrollment terminal
Example:
Router(ca-trustpoint)#enrollment terminal
Specifies manual certificate enrollment via the cut-and-paste method for trustpoint peers. The certificate request displayed on the console terminal can be manually copied.

Step 5  revocation-check none
Example:
Router(ca-trustpoint)#revocation-check none
Specifies that the certificate check is ignored.

Step 6  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**

```plaintext
crypto signaling remote-addr remote ip address remote ip mask trustpoint CUBE's 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**

```plaintext
exit
```

**Example:**

```
Router(config-sip-ua)#exit
```

Exits sip-ua configuration mode.

---

**Enforce Strict SRTP**

**Step 1**

```plaintext
enable
```

**Example:**

```
Router#enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2**

```plaintext
configure terminal
```

**Example:**

```
Router#configure terminal
```

Enters global configuration mode.

**Step 3**

```plaintext
voice service voip
```

**Example:**

```
Router(config)#voice service voip
```

Enters voice service configuration mode and specifies the encapsulation method as VoIP.

**Step 4**

```plaintext
srtp
```

**Example:**

```
Router(conf-voi-ser)#srtp
```

Enforces SRTP to secure the call flow through CUBE.

**Step 5**

```plaintext
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 1061
2. Create Certificate Map for HTTPS Peer Trustpoint, on page 1062
3. Configure HTTPS TLS Version, on page 1063
4. Configure Supported Cipher Suites, on page 1064
5. Apply Certificate Map to HTTPS Peer Trustpoint, on page 1064

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.
   - Enter your password if prompted.

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-gcm-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
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 (EKR) 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
• `ntp broadcast destination ip-address key key version version`

• `ntp broadcast version version`

• Do not configure NTP multicast command `ntp multicast version version`.

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 1069
• Glossary, on page 1073
CHAPTER 82

Additional References

The following sections provide references related to the CUBE Configuration Guide.

- Related References, on page 1069
- Standards, on page 1070
- MIBs, on page 1070
- RFCs, on page 1070
- Technical Assistance, on page 1072

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 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>RTP: A Transport Protocol for Real-Time Applications</td>
</tr>
<tr>
<td>RFC 2131</td>
<td>Dynamic Host Configuration Protocol</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 with Cisco products and technologies.</td>
<td></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>
Glossary

- 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.