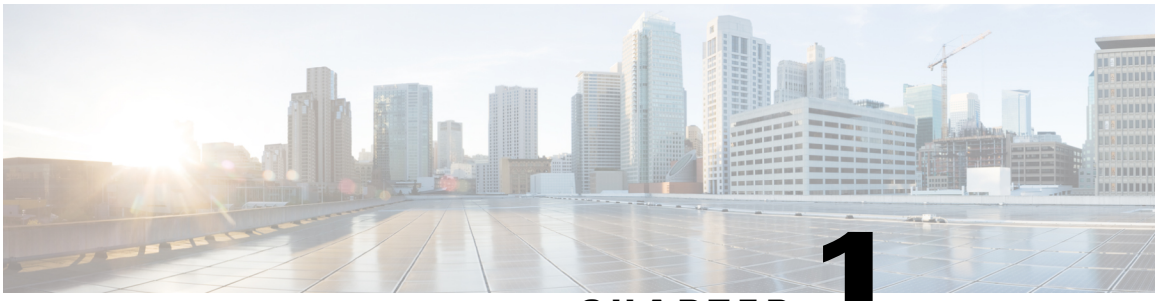
Cisco Unified Border Element Configuration Guide

Last Modified: March 31, 2016

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



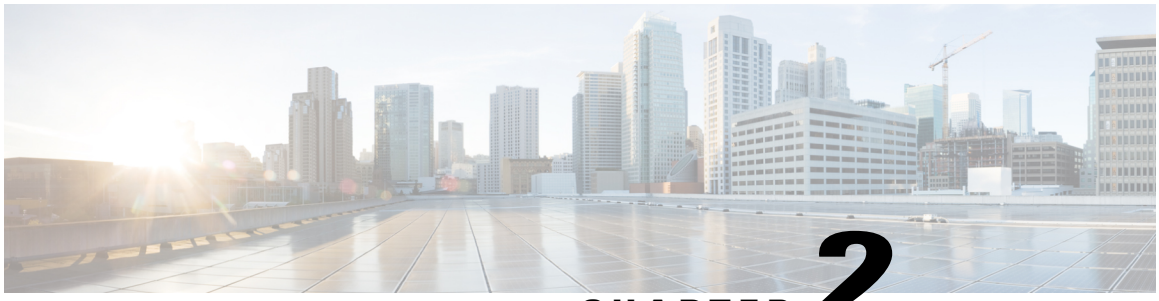
CHAPTER 1

Read Me First

Important Information about Cisco IOS XE 16

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

Note The Feature Information table in the technology configuration guide mentions when a feature was introduced. It might or might not mention when other platforms were supported for that feature. To determine if a particular feature is supported on your platform, look at the technology configuration guides posted on your product landing page. When a technology configuration guide is displayed on your product landing page, it indicates that the feature is supported on that platform.



CHAPTER 2

New and Changed Information

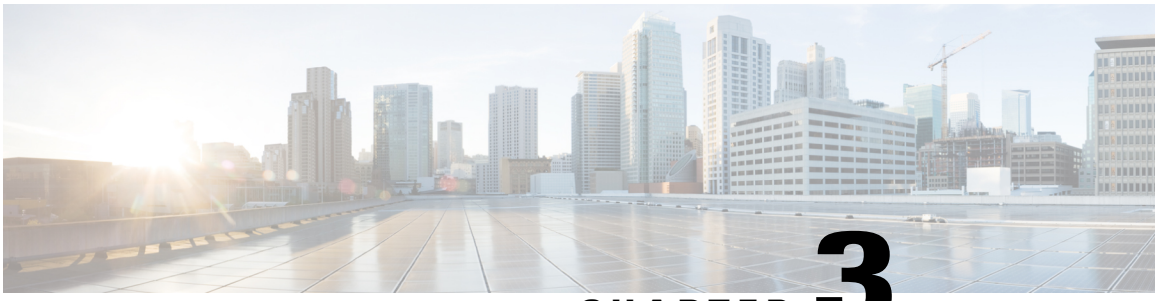
New and Changed Information

This section provides the details of new and changed information in Cisco Unified Border Element (CUBE) Configuration Guide.

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

Description	Documented at
Support for Cisco IOS XE Denali 16	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/read-me-first.html
Support for Multi-VRF	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-cube-multi-vrf.html
Configuring Multi-Tenants on SIP Trunks	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-cube-multi-tenants.html
Support for Video Suppression	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-audio-forced.html
CVP Survivability TCL support with High Availability	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-cube-cvptcl-ha.html
Support for Session-Identifier	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-cube-session-id.html

New and Changed Information



CHAPTER 3

Supported Platforms

CUBE is supported on various platforms running on Cisco IOS Software Releases, Cisco IOS-XE Software Releases, or Cisco IOS XE Denali Software Releases.

Note For information on migrating from existing Cisco IOS XE 3S releases to the Cisco IOS XE Denali 16.2 release, see Cisco IOS XE Denali 16.2 Migration Guide for Access and Edge Routers

The following table provides details of supported platforms:

Cisco Router Platforms	Cisco Router Models	Cisco IOS Software Releases
Cisco Integrated Services Generation 2 Routers (ISR G2)	Cisco 2900 Series Integrated Services Routers Cisco 3900 Series Integrated Services Routers	Cisco IOS 12 M and T Cisco IOS 15 M and T
Cisco 4000 Series Integrated Services Routers (ISR G3)	Cisco 4321 Integrated Services Routers Cisco 4331 Integrated Services Routers Cisco 4351 Integrated Services Routers Cisco 4431 Integrated Services Routers Cisco 4451 Integrated Services Routers	Cisco IOS XE 3S Cisco IOS XE Denali 16.2.1 onwards
Cisco Aggregated Services Routers (ASR)	Cisco 1001-X Aggregated Services Routers Cisco 1002-X Aggregated Services Routers Cisco 1004 Aggregated Services Routers with RP2 Cisco 1006 Aggregated Services Routers with RP2	Cisco IOS XE 3S Cisco IOS XE Denali 16.2.1 onwards[1]

Cisco Router Platforms	Cisco Router Models	Cisco IOS Software Releases
Cisco Cloud Services Routers (CSR)	Cisco Cloud Services Router 1000V series	Cisco IOS XE 3.15 onwards Cisco IOS XE Denali 16.2.1 onwards

1 Cisco IOS-XE Denali 16 .2.1 requires a minimum of ASR1001-X, 1002-X, 1004/1006 RP2, ESP20 (Embedded Service Processor), and SIP40 (SPA Interface processor).

- Feature Comparison on Supported Platforms , page 6

Feature Comparison on Supported Platforms

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

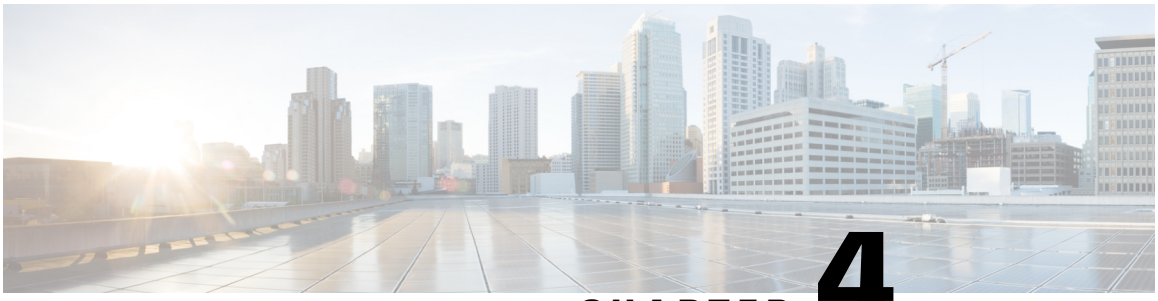
Note For collaboration features' support on Cisco ISR 4000 Series Routers, Cisco IOS XE Release 3.13.1 or later is recommended. Cisco Cloud Services Routers 1000V Series support is available from Cisco IOS XE Release 3.15 onwards.

Features	Cisco ASR 1000 Series Routers	Cisco ISR G2 Series Routers	Cisco ISR 4000 Series Routers	Cisco CSR 1000V Series Routers
High Availability Implementation	Redundancy Group Infrastructure	Hot Standby Router Protocol (HSRP) Based	Redundancy Group Infrastructure	Redundancy Group Infrastructure
Media Forking	Yes (Cisco IOS XE Release 3.8 onwards)	Yes (Cisco IOS Release 15.2(1)T onwards)	Yes (Cisco IOS XE Release 3.10 onwards)	Yes
Software MTP registered to CUCM (Including HA Support)	Yes (Cisco IOS XE Release 3.6 onwards)	Yes	Yes	Yes
DSP Card Type	SPA-DSP	PVDM2/PVDM3	PVDM4	No
Transcoder registered to CUCM	No	Yes (Exists via SCCP)	Yes (Exists via SCCP - Cisco IOS XE Release 3.11 onwards)	No
Transcoder—LTI	Yes	Yes	Yes	No

Features	Cisco ASR 1000 Series Routers	Cisco ISR G2 Series Routers	Cisco ISR 4000 Series Routers	Cisco CSR 1000V Series Routers
Cisco UC Gateway Services API	Yes (Cisco IOS XE Release 3.8 onwards)	Yes (Cisco IOS Release 15.2(2)T onwards)	Yes	Yes
Noise Reduction & ASP	Yes	Yes (Cisco IOS Release 15.2(3)T onwards)	Yes	No
Call Progress Analysis	Yes (Cisco IOS XE Release 3.9 onwards ; Recommended - Cisco IOS XE Release 3.15S)	Yes (Cisco IOS Release 15.3(2)T onwards ; Recommended -Cisco IOS Release 15.5(2)T onwards)	Yes Recommended - Cisco IOS XE Release 3.15S	No
SRTP-RTP	Yes (No DSPs required)	Yes (DSPs required)	Yes (No DSPs required)	Yes (No DSPs required)
CUBE for SP Managed and Hosted Services	Yes	Yes	Yes	Yes

PART I

CUBE Fundamentals and Basic Setup



CHAPTER 4

Overview of Cisco Unified Border Element

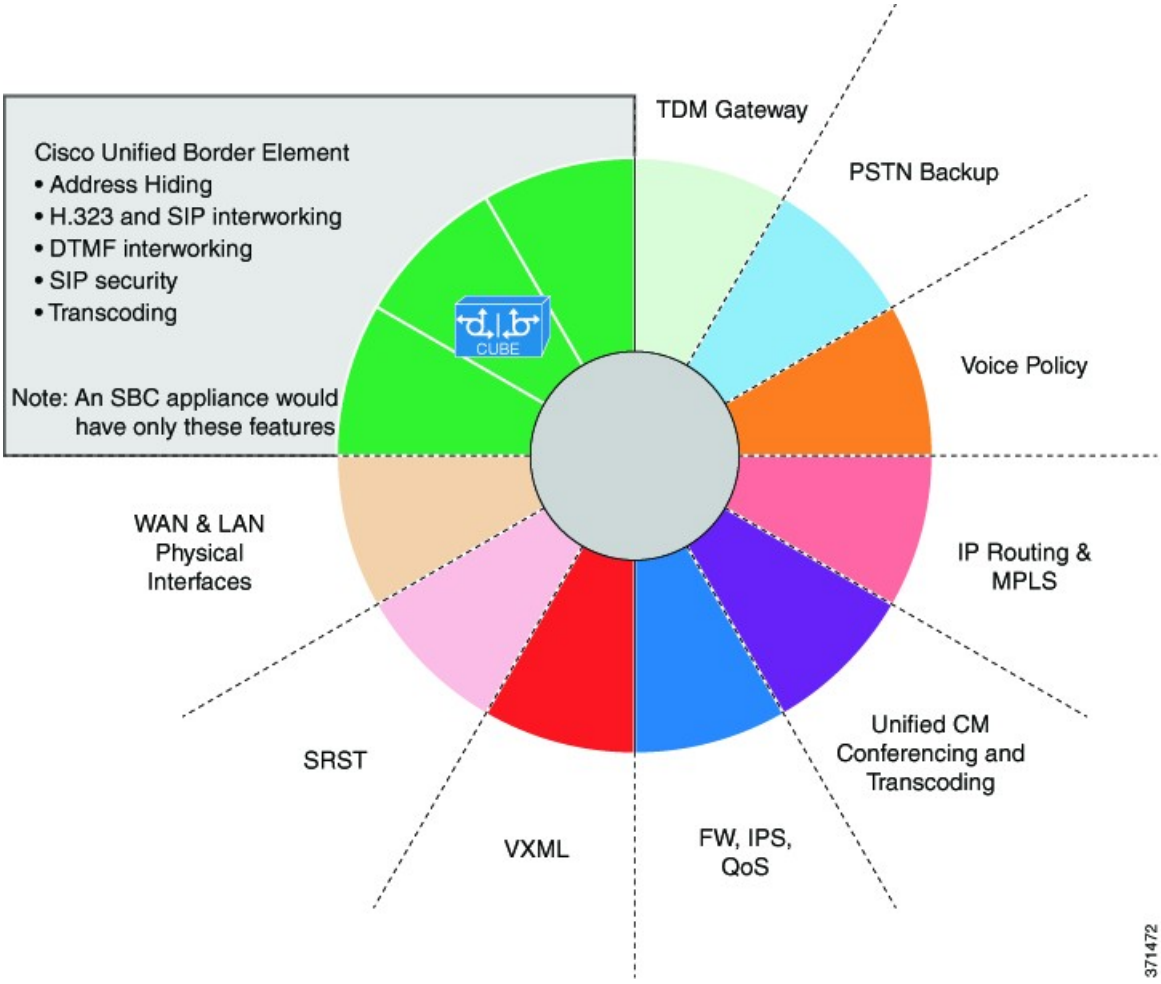
Cisco Unified Border Element (CUBE) is a unified communications border element that bridges voice and video connectivity between two separate VoIP networks. It is similar to a traditional voice gateway, except for the replacement of physical voice trunks with an IP connection. Traditional gateways connect 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

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:

Figure 1: Cisco Unified Border Element—More Than an SBC



The CUBE provides a network-to-network interface point for:

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

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

Protocol interworking is possible for the following combinations:

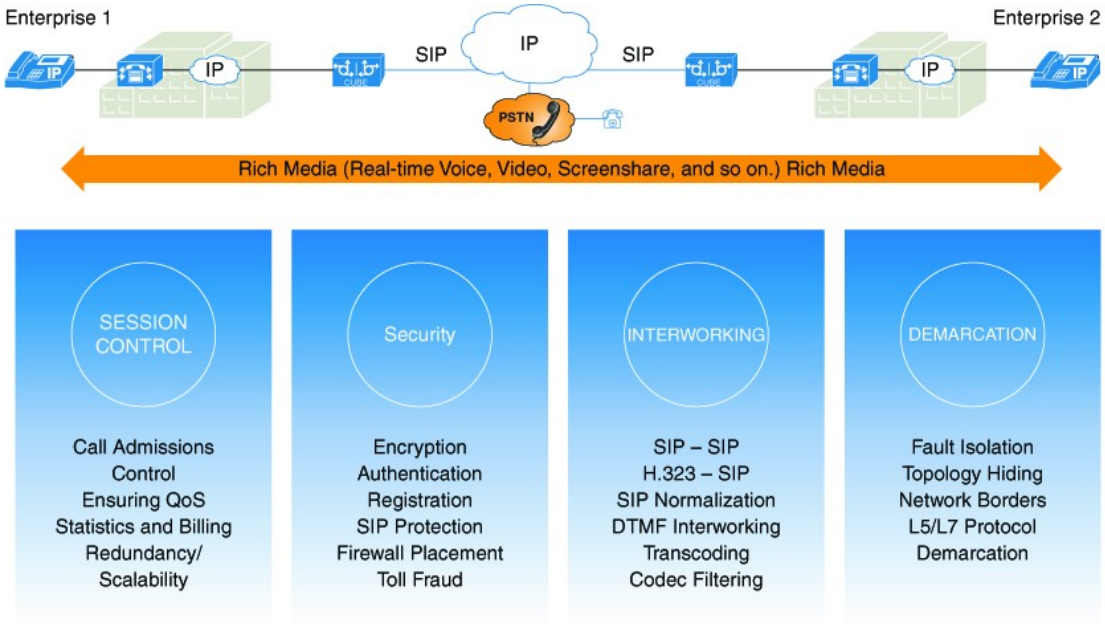
- H.323-to-SIP interworking
- H.323-to-H.323 interworking
- SIP-to-SIP interworking

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

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

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

Figure 2: Why Does an Enterprise Need the CUBE?

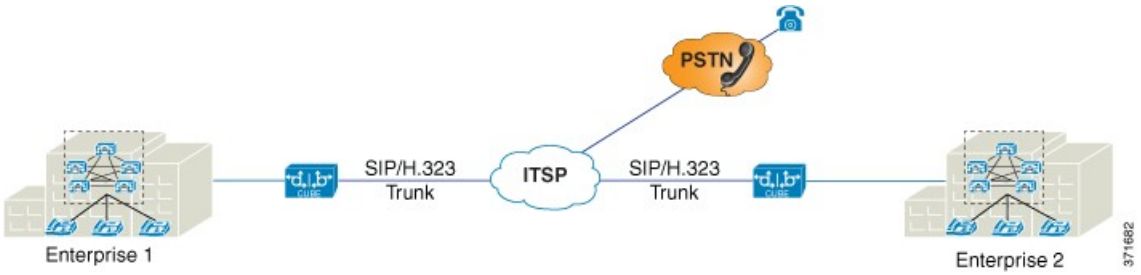


If an enterprise subscribes to VoIP services offered by an ITSP, connecting the enterprise CUCM through a CUBE provides network demarcation capabilities, such as security, topology hiding, transcoding, call admission control, protocol normalization and SIP registration, none of which is possible if CUCM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

SIP/H.323 Trunking

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

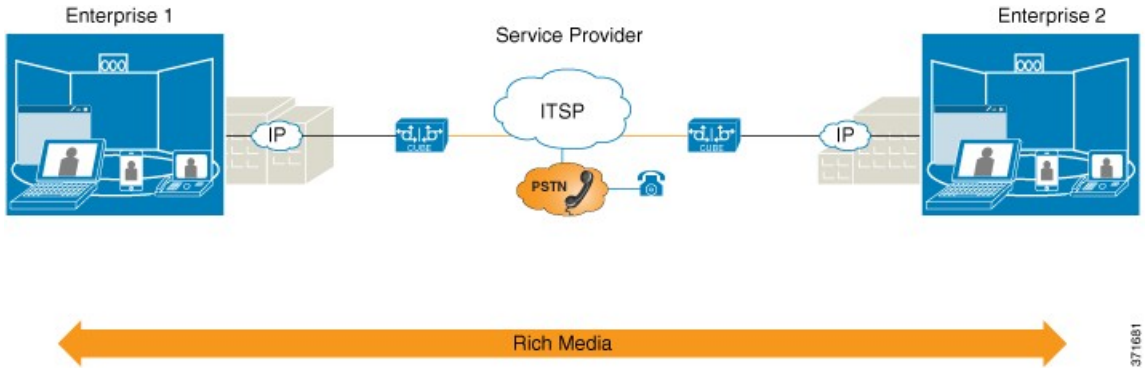
Figure 3: SIP/H.323 Trunking



SIP trunking overcomes TDM barriers, in that it:

- Improves efficiency of interconnection between networks
- Simplifies PSTN interconnection with IP end-to-end
- Enables rich media services to employees, customers, and partners
- Carries converged voice, video, and data traffic

Figure 4: SIP Trunking Overcomes TDM Barriers

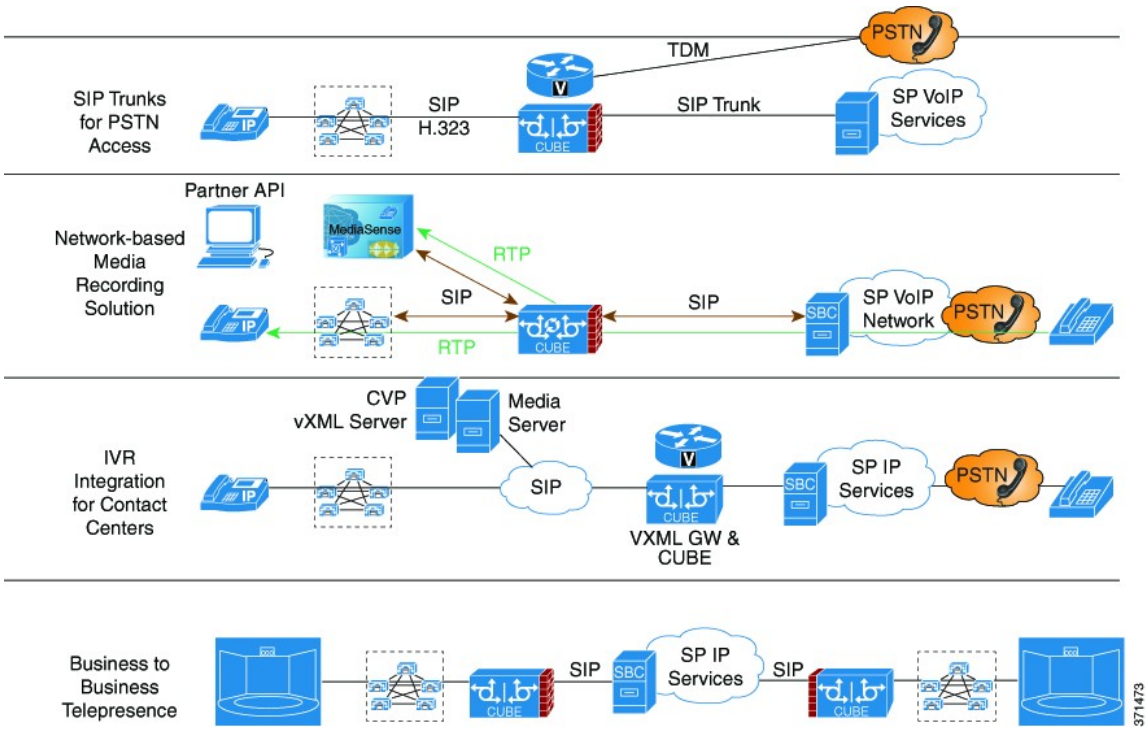


Typical Deployment Scenarios for CUBE

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.

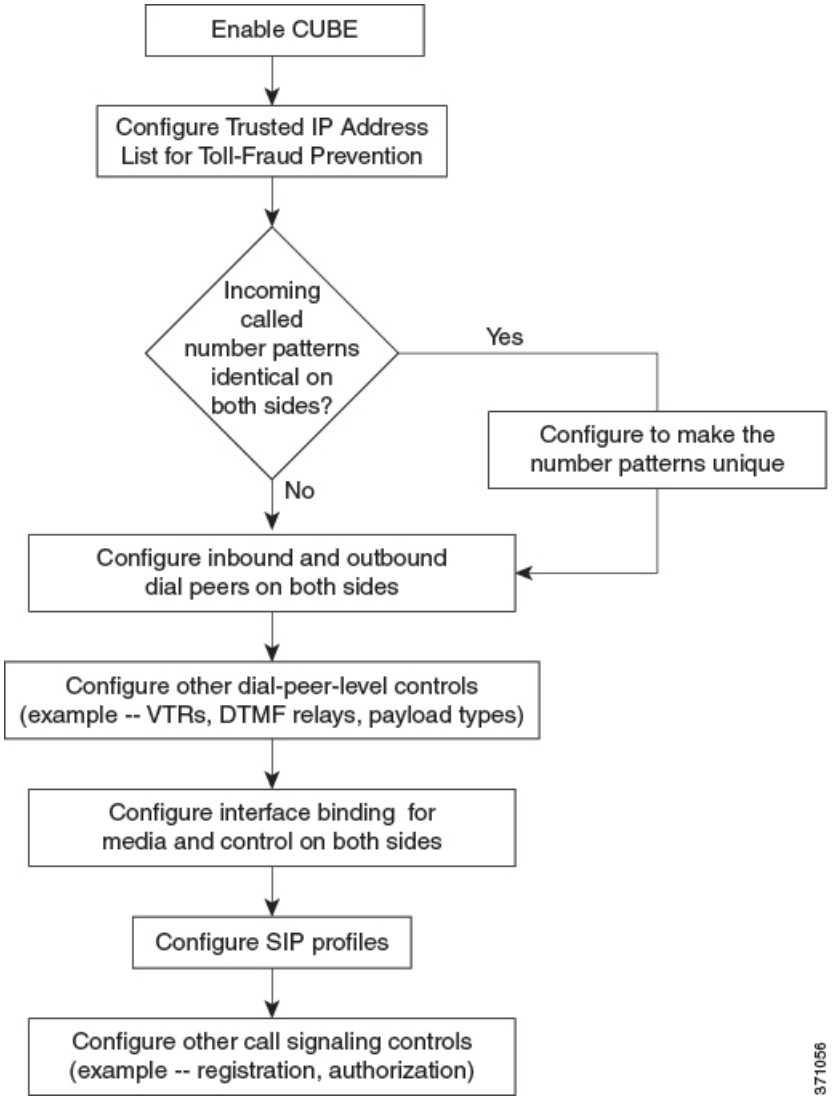
Figure 5: Typical Deployment Scenarios



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.

Figure 6: CUBE Configuration Workflow



The following sections describe the basic setup of CUBE through the steps involved in migrating the XYZ corporation to CUBE using a SIP trunk.

Enabling the CUBE Application on a Device

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters global VoIP configuration mode.
Step 4	**mode border-element license capacity** *sessions* **Example:** `Device(conf-voi-serv)# mode border-element license capacity 200`	Enables the set of commands used in the CUBE. • You can configure the number of licenses (capacity) to be enabled for the CUBE.
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. The CUBE status display is enabled only if the **mode border-element** command is configured with call license capacity.

Example:
```
Device# show cube status

CUBE-Version : 10.0.1
SW-Version : 15.4.2.T, Platform 3845
HA-Type : none
Licensed-Capacity : 200
```

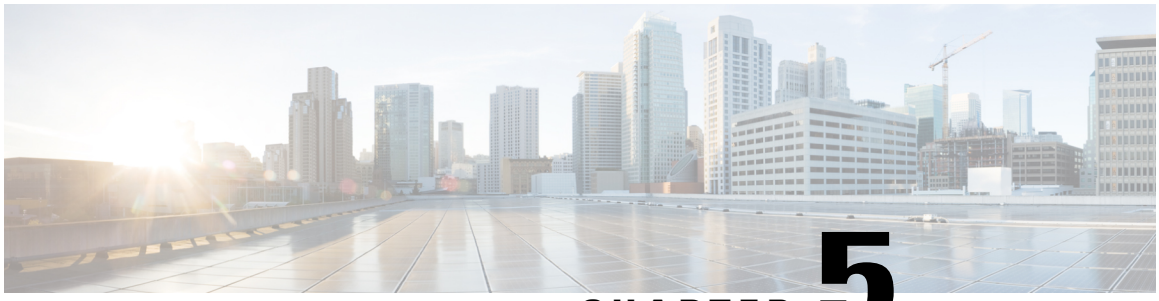
Configuring a Trusted IP Address List for Toll-Fraud Prevention

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters global VoIP configuration mode.
Step 4	**ip address trusted list** **Example:** `Device(conf-voi-serv)# ip address trusted list`	Enters IP address trusted list mode and enables the addition of valid IP addresses.
Step 5	**ipv4** *ipv4-address* [*network-mask*] **Example:** `Device(cfg-iptrust-list)# ipv4 192.0.2.1 255.255.255.0`	Allows you to add up to 100 IPv4 addresses in the IP address trusted list. Duplicate IP addresses are not allowed. • The *network-mask* argument allows you to define a subnet IP address.
Step 6	**ipv6** *ipv6-address* **Example:** `Device(cfg-iptrust-list)# ipv6 2001:DB8:0:ABCD::1/48`	Allows you to add IPv6 addresses to the trusted IP address list.
Step 7	**end** **Example:** `Device(cfg-iptrust-list)# end`	Returns to privileged EXEC mode.



CHAPTER 5

Virtual CUBE

Cisco Unified Border Element (CUBE) has been traditionally supported as an installation on physical routers such as Cisco Aggregation Services Router Series (ASR) and Cisco Integration Services Router Series (ISR). With the introduction of virtualization, CUBE will also be supported as a virtualized form factor from Cisco IOS XE 3.15S release. As part of the enhancement, CUBE features are integrated into Cisco CSR 1000V Series Cloud Services Routers (Cisco CSR 1000V router).

CUBE virtualization enables cloud-based data center deployment strategies.

Feature Information for Virtual CUBE

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

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

Table 1: Feature Information for Virtual CUBE Support

Feature Name	Releases	Feature Information
Virtual CUBE Feature Support	Cisco IOS XE Release 3.15S	Provides support for virtual CUBE on Cisco Cloud Services Router, Cisco CSR 1000V Series.

Prerequisites for Virtual CUBE

Virtual CUBE has the following prerequisites:

Hardware

- Virtual CUBE integrated into the Cisco CSR 1000V routers runs on a hypervisor on the Cisco UCS. Virtual CUBE works with the ESXi hypervisors supported by Cisco CSR 1000V Series router. Virtual CUBE is supported on the following configurations.

Table 2: Virtual CUBE Hardware and Hypervisor Support

Cisco UCS	Hypervisor	Virtual CUBE Form Factor	
		CPU	Memory (GB)
UCSC-Base-M2-C460	ESXi 5.5.0	1	2.5
UCSC-C220-M3S	ESXi 5.5.0	1	4
UCSC-C220-M3S	ESXi 5.1.0	4	4
		4	8

- We recommend that you to disable hyper threading while configuring Virtual CUBE on Cisco UCS.
- A minimum of two network interfaces are required for configuring Virtual CUBE.

Software

- Obtain the relevant license for the Cisco CSR 1000V router. For details on the licensing package support, see Licensing Package Support , on page 24.
- Install the appropriate Cisco IOS image on the Cisco CSR 1000V router and configure a working VoIP network. For details on installation, see Installation, on page 24.

For details on the ESXi hypervisor support, see the section on Hardware, on page 22.

Features Supported with Virtual CUBE

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

- All DSP based features
  - Codec Transcoding, Transrating
  - DTMF interworking
  - CPA
  - Noise Reduction (NR), Acoustic Shock Protection (ASP), and Audio Gain
- Limited Voice Class Codec (VCC) support
  - Codec supported on peer leg will be included in offer. Other codecs will be filtered out.
- IOS Gatekeeper
- H.323 Interworking
- IOS based Hardware MTP

Restrictions

All caveats, restrictions, and limitations of Cisco ASR IOS-XE 3.15 Release are applicable to virtual CUBE.

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

High Availability

Virtual CUBE uses Redundancy Group infrastructure for HA. HA will be between two virtual CUBE CSR instances running on either the same host or across different hosts connected through a switch. Geographic redundancy is not supported.

For more information on High Availability, see CUBE High Availability Overview, on page 417.

Media

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

Licensing Package Support

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

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

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

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

Installation

Virtual CUBE can be installed in two ways:

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

Installation using OVA File

The following file type is used to install virtual CUBE on Cisco UCS:

- .ova

You can use the OVA file included in the Cisco CSR 1000V router software image package to install virtual CUBE. The file is used for deploying the OVA template on the VM (in TAR format).

Note Explicit subscription of CPUs and Memory is required while deploying OVA provided by Cisco CSR 1000V.

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

Installation using ISO Image

The following file type is used to install virtual CUBE on Cisco UCS:

- .iso

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

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

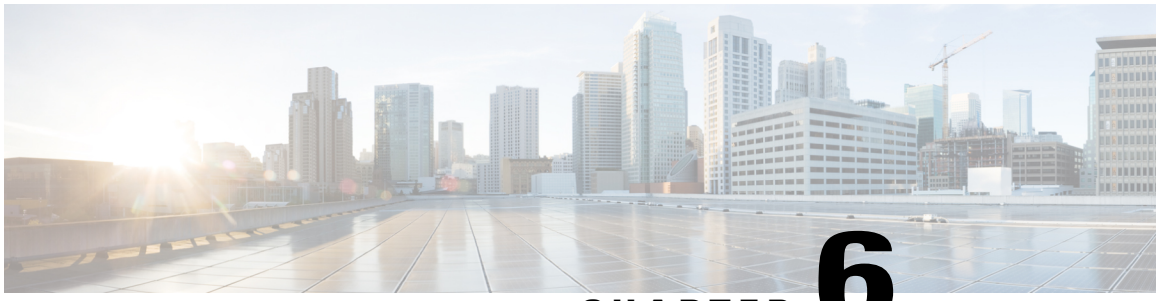
How to Enable Virtual CUBE on Cisco CSR 1000V Series Router

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

Troubleshooting Virtual CUBE Support

To troubleshoot virtual CUBE, follow the same procedure as that of Cisco ASR routers. This includes crash file decoding, decoding traceback, and so on. For more details, see http://www.cisco.com/c/en/us/support/docs/routers/asr-1000-series-aggregation-services-routers/109723-asr-crash.html .

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



CHAPTER 6

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.

Dial Peers in CUBE

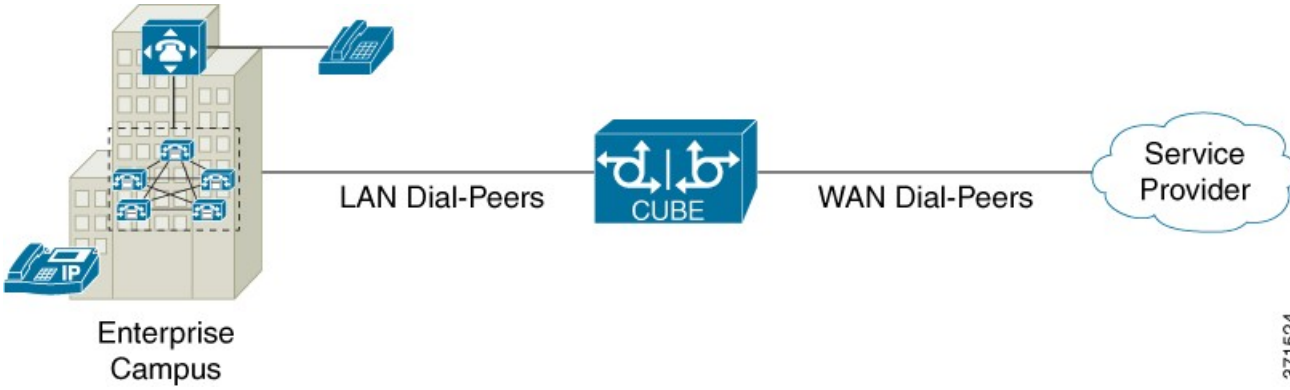
A dial peer is a static routing table, mapping phone numbers to interfaces or IP addresses.

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.

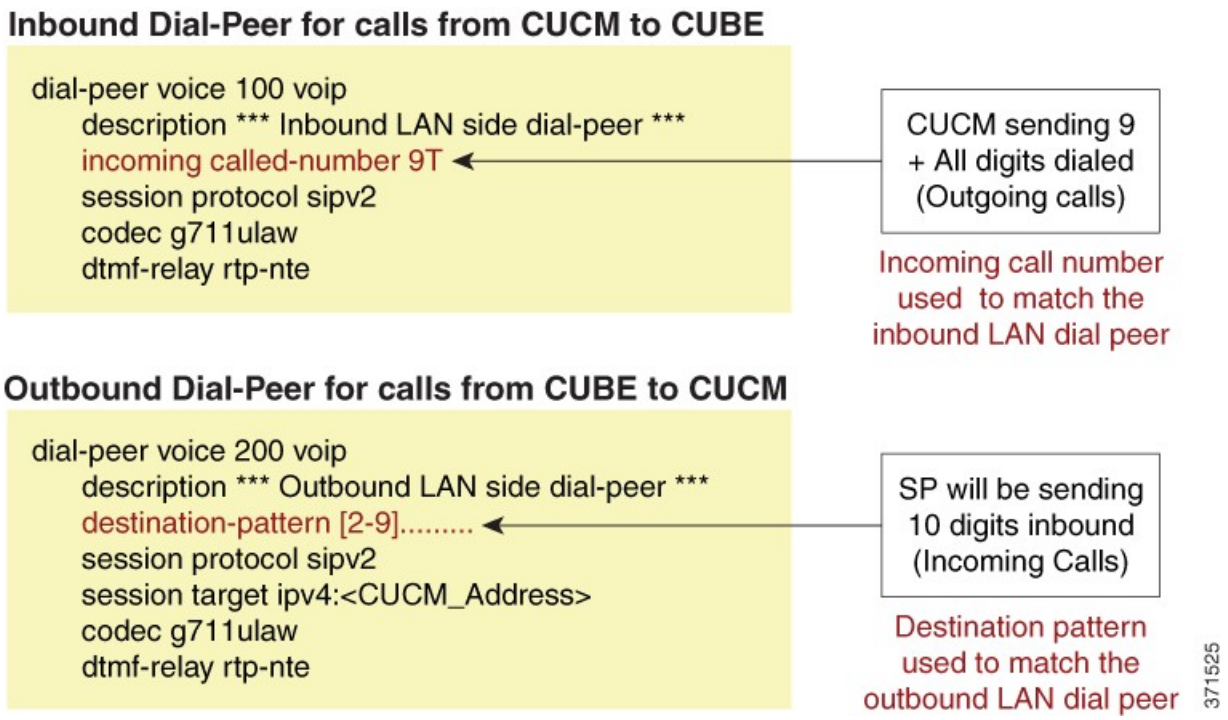
In CUBE, dial peers can also be classified as LAN dial peers and WAN dial peers based on the connecting entity from which CUBE sends or receives calls.

Figure 7: LAN and WAN Dial Peers



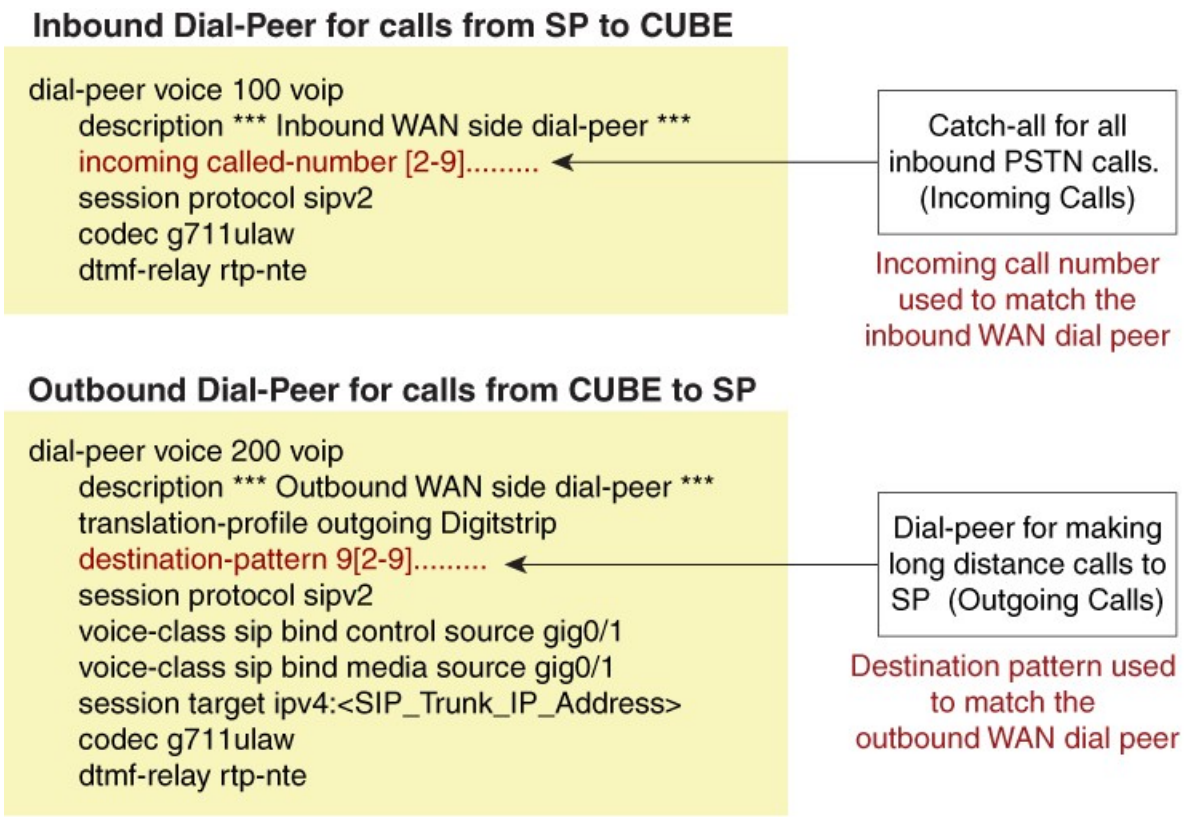
A LAN dial peer is used to send or receive calls between CUBE and the Private Branch Exchange (PBX)—a system of telephone extensions within an enterprise. Given below are examples of inbound and outbound LAN dial peers.

Figure 8: LAN Dial Peers



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.

Figure 9: 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 3: Incoming Dial-Peer Matching

Command in Dial-Peer Configuration	Description	Call Setup Element
incoming called-number *DNIS-string*	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.	DNIS number

Command in Dial-Peer Configuration	Description	Call Setup Element
answer-address *ANI-string*	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.	ANI string
destination-pattern *ANI-string*	This command uses the inbound call leg to the inbound dial peer.	ANI string for inbound
{**incoming called | incoming calling**} **e164-pattern-map** *pattern-map-group-id*	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.	E.164 Patterns
voice class uri *URI-class-identifier* with **incoming uri** {**from | request | to | via**} *URI-class-identifier*	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**	Directory URI
incoming uri {**called | callling**} *URI-class-identifier*	This command uses the directory URI (Uniform Resource Identifier) number to match the outgoing H.323 call leg to an outgoing dial peer. The command calls a globally defined voice class identifier where the directory URI is configured.	Directory URI

Table 4: Outgoing Dial-Peer Matching

Dial-Peer Command	Description	Call Setup Element
destination-pattern *DNIS-string*	This command uses DNIS string to match the outbound call leg to the outbound dial peer.	DNIS string for outbound ANI string for inbound
destination *URI-class-identifier*	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.	Directory URI
destination e164-pattern-map *pattern-map-group-id*	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.	E.164 patterns

Preference for Dial-Peer Matching

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

- **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*

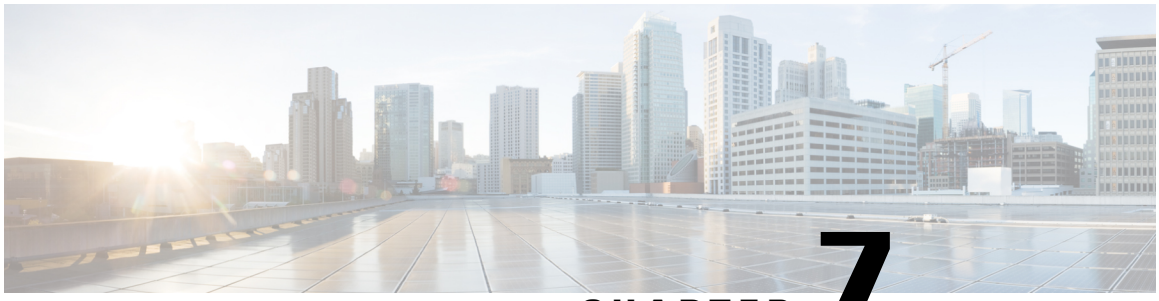
The following is the order in which inbound dial-peer is matched for H.323 call-legs:

- **incoming uri** {**called**} *URI-class-identifier*
- **incoming uri** {**callling**} *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*



CHAPTER 7

DTMF Relay

The DTMF Relay feature allows CUBE to send dual-tone multi-frequency (DTMF) digits over IP.

This chapter talks about DTMF tones, DTMF relay mechanisms, how to configure DTMF relays, and interoperability and priority with multiple relay methods.

Feature Information for DTMF Relay

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 5: Feature Information for DTMF Relay

Feature Name	Releases	Feature Information
DTMF Relay	Cisco IOS 12.1(2)T Cisco IOS XE 2.1	The DTMF Relay feature allows CUBE to send DTMF digits over IP. The **dtmf-relay** command was added.

Information About DTMF Relay

DTMF Tones

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

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

DTMF Relay

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

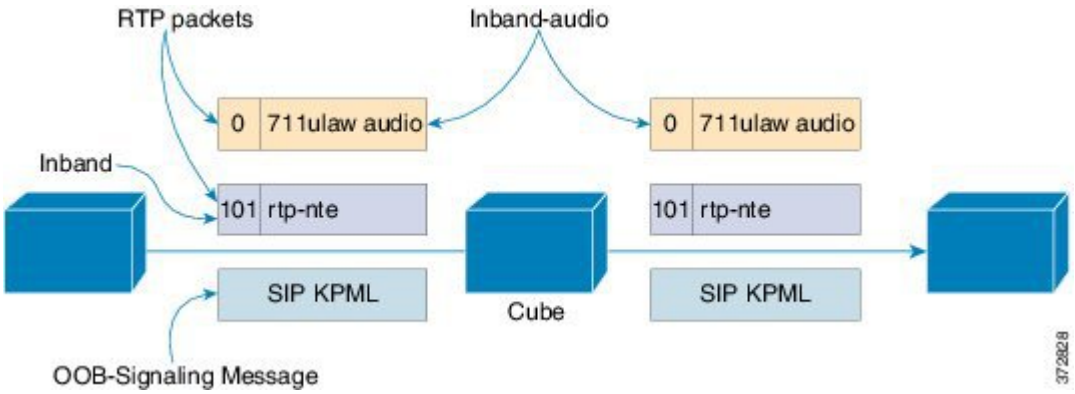
In-band DTMF-Relay passes the DTMF digits using the RTP media stream and uses a special payload type identifier in the RTP header to distinguish DTMF digits from actual voice communication. This method is more likely to work on lossless codecs, such as G.711.

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

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

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

Figure 10: DTMF Relay Mechanism



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

- **h245-alphanumeric and h245-signal—**These two methods are available only on H.323 dial peers. This is an out-of-band DTMF relay mechanism that transports the DTMF signals using H.245, which is the media control protocol of the H.323 protocol suite.
  The H245-signal method carries more information about the DTMF event (such as its actual duration) than the H245-Alphanumeric method. This addresses a potential problem with the alphanumeric method when interworking with other vendors' systems.

- **sip-notify—**This method is available on SIP dial peers only. This is a Cisco proprietary out-of-band DTMF relay mechanism that transports DTMF signals using SIP-Notify message. The SIP Call-Info header is used to indicate the use of the SIP-Notify DTMF relay mechanism. The message is acknowledged with a 18x or 200 response message containing a similar SIP Call-Info header.
  The Call-Info header for NOTIFY-based out-of-band relay is as follows:

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

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

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

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

- **sip-kpml—**This method is available only on SIP dial peers. This is an out-of-band DTMF relay mechanism defined by RFC 4730 that registers the DTMF signals using SIP-Subscribe messages and transports the DTMF signals using SIP-Notify messages containing an XML-encoded body. This method is also known as Key Press Markup Language.
  If you configure KPML on the dial peer, the gateway sends INVITE messages with kpml in the Allow-Events header.

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

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

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

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

- **sip-info—**The sip-info method is available only on SIP dial peers. This is an out-of-band DTMF relay mechanism that registers the DTMF signals using SIP-Subscribe messages and transports the DTMF signals using SIP-Info messages. The body of the SIP message consists of signaling information and uses the content-type application/dtmf-relay.
  The method is always enabled for SIP dial peers, and is invoked when a SIP INFO message is received with DTMF relay content.

  This following sample message shows a SIP INFO message received by the gateway with specifics about the DTMF tone to be generated. The combination of the From, To, and Call-ID headers identifies

the call leg. The signal and duration headers specify the digit, in this case 1, and duration, 160 milliseconds in the example, for DTMF tone play.

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

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

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

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

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

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

- **G711 audio—**This is an inband DTMF relay mechanism that is enabled by default and requires no configuration. Digits are transmitted within the audio of the phone conversation, that is, it is audible to the conversation partners; therefore, only uncompressed codecs like g711 alaw or ulaw can carry inband DTMF reliably. Female voices are known to, sometimes, trigger the recognition of a DTMF tone. Digits are passed along just like the rest of your voice as normal audio tones with no special coding or markers using the same codec as your voice does and are generated by your phone.

Configuring DTMF Relays

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

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

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

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

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

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

Table 6: Supported H.323 and SIP DTMF Relay Methods

	H.323 Gateway	SIP Gateway
In-band	**cisco-rtp, rtp-nte**	**rtp-nte**
Out-of-band	**h245-alphanumeric, h245-signal**	**sip-notify, sip-kpml, sip-info**

Interoperability and Priority with Multiple DTMF Relay Methods

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

- If you configure rtp-nte, sip-notify, and sip-kpml, the outgoing INVITE contains a SIP Call-Info header, an Allow-Events header with KPML, and an 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** (highest priority)
  - **rtp-nte**
  - **None—**Send DTMF in-band

  H.323 gateways select DTMF relay mechanisms using the following priority:
  - **cisco-rtp**
  - **h245-signal**
  - **h245-alphanumeric**
  - **rtp-nte**
  - **None—**Send DTMF in-band

DTMF Interoperability Table

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

- RTP-RTP Flow-Through
- RTP-RTP with transcoder Flow-Through
- RTP-RTP Flow Around
- RTP-RTP with high-density transcoder Flow Through
- SRTP-RTP Flow Through

Table 7: RTP-RTP Flow-Through

	dial-peer protocol	H.323			SIP				Inband
dial-peer protocol	**DTMF Relay Type**	**h245-alpha numeric**	**h245-signal**	**rtp-nte**	**rtp-nte**	**sip-kpml**	**sip-notify**	**sip-info**	**Voice Inband (G.711)**
H.323	**h245-alpha numeric**	Supported		Supported	Supported	Supported	Supported		
	h245-signal		Supported	Supported	Supported	Supported	Supported		
	rtp-nte	Supported	Supported	Supported	Supported		Supported		Supported*
SIP	**rtp-nte**	Supported	Supported	Supported	Supported		Supported	Supported	Supported*
	sip-kpml	Supported	Supported			Supported			
	sip-notify	Supported	Supported	Supported	Supported		Supported		
	sip-info				Supported			Supported	
Inband	**Voice Inband (G.711)**			Supported*	Supported*				Supported

* media resource is required (Transcoder) for IOS versions

Table 8: RTP-RTP with transcoder Flow-Through

	dial-peer protocol	H.323			SIP				Inband
dial-peer protocol	**DTMF Relay Type**	**h245-alpha numeric**	**h245-signal**	**rtp-nte**	**rtp-nte**	**sip-kpml**	**sip-notify**	**sip-info**	**Voice Inband (G.711)**
H.323	**h245-alpha numeric**	Supported		Supported	Supported	Supported	Supported		
	h245-signal		Supported	Supported	Supported	Supported	Supported		
	rtp-nte	Supported	Supported	Supported	Supported				Supported

	dial-peer protocol	H.323			SIP				Inband
dial-peer protocol	**DTMF Relay Type**	**h245-alpha numeric**	**h245-signal**	**rtp-nte**	**rtp-nte**	**sip-kpml**	**sip-notify**	**sip-info**	**Voice Inband (G.711)**
SIP	**rtp-nte**	Supported	Supported	Supported	Supported			Supported	Supported
	sip-kpml	Supported	Supported			Supported			
	sip-notify	Supported	Supported	Supported			Supported		
	sip-info				Supported			Supported	
Inband	**Voice Inband (G.711)**			Supported	Supported				

Table 9: RTP-RTP Flow Around

	dial-peer protocol	H.323			SIP				Inband
dial-peer protocol	**DTMF Relay Type**	**h245-alpha numeric**	**h245-signal**	**rtp-nte**	**rtp-nte**	**sip-kpml**	**sip-notify**	**sip-info**	**Voice Inband (G.711)**
H.323	**h245-alpha numeric**	Supported							
	h245-signal		Supported						
	rtp-nte			Supported					Supported*
SIP	**rtp-nte**				Supported				Supported*
	sip-kpml					Supported			
	sip-notify						Supported		
	sip-info							Supported	
Inband	**Voice Inband (G.711)**			Supported*	Supported*				Supported

* media resource is required (Transcoder) for IOS versions

Table 10: RTP-RTP with high-density transcoder Flow Through

	dial-peer protocol	H.323			SIP				Inband
dial-peer protocol	**DTMF Relay Type**	**h245-alpha numeric**	**h245-signal**	**rtp-nte**	**rtp-nte**	**sip-kpml**	**sip-notify**	**sip-info**	**Voice Inband (G.711)**
H.323	**h245-alpha numeric**	Supported				Supported	Supported		
	h245-signal		Supported			Supported	Supported		
	rtp-nte			Supported	Supported				Supported
SIP	**rtp-nte**			Supported	Supported				Supported
	sip-kpml	Supported	Supported			Supported			
	sip-notify	Supported	Supported				Supported		
	sip-info							Supported	
Inband	**Voice Inband (G.711)**			Supported	Supported				

Table 11: SRTP-RTP Flow Through

	dial-peer protocol	H.323			SIP				Inband
dial-peer protocol	**DTMF Relay Type**	**h245-alpha numeric**	**h245-signal**	**rtp-nte**	**rtp-nte**	**sip-kpml**	**sip-notify**	**sip-info**	**Voice Inband (G.711)**
H.323	**h245-alpha numeric**								
	h245-signal								
	rtp-nte								
SIP	**rtp-nte**				Supported				Supported
	sip-kpml					Supported			
	sip-notify						Supported		
	sip-info							Supported	

	dial-peer protocol	H.323			SIP				Inband
dial-peer protocol	DTMF Relay Type	h245-alpha numeric	h245-signal	rtp-nte	rtp-nte	sip-kpml	sip-notify	sip-info	Voice Inband (G.711)
Inband	Voice Inband (G.711)				Supported				Supported

Verifying DTMF Relay

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **show sip-ua calls**

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

Example:

```
Device# show sip-ua calls

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

```
     Media Source IP Addr:Port: 192.0.2.5:17576
     Media Dest IP Addr:Port  : 192.0.2.6:17468
     Orig Media Dest IP Addr:Port : 0.0.0.0:0
   Number of SIP User Agent Client(UAC) calls: 1
SIP UAS CALL INFO
   Number of SIP User Agent Server(UAS) calls: 0
```

Step 2 **show sip-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                : 9598A547-5C1311E2-8008F709-2470C996@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                : 9598A547-5C1311E2-8008F709-2470C996@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
```

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

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
No.         Timestamp         Digit           Duration
=======================================================

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

SIP UAS CALL INFO
Call 1
SIP Call ID                : 22BC36A5-F01411E3-81808A6A-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
No.         Timestamp         Digit           Duration
=======================================================

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

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

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

Example:

```
Device# show sip-ua history dtmf-relay sip-notify

Total SIP call legs:2, User Agent Client:1, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID                : 29BB98C-F01311E3-8297DE9F-78C438FF@10.86.176.119
   State of the call       : STATE_ACTIVE (7)
   Calling Number          : 2017
```

```
   Called Number           : 1011
   CC Call ID              : 252
No.         Timestamp         Digit           Duration
==========================================================

Call 2
SIP Call ID                : 550E973B-F01311E3-817A8A6A-5FE95113@10.86.176.142
   State of the call       : STATE_ACTIVE (7)
   Calling Number          : 2017
   Called Number           : 1011
   CC Call ID              : 251
No.         Timestamp         Digit           Duration
==========================================================

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

SIP UAS CALL INFO
Call 1
SIP Call ID                : 550E973B-F01311E3-817A8A6A-5FE95113@10.86.176.142
   State of the call       : STATE_ACTIVE (7)
   Calling Number          : 2017
   Called Number           : 1011
   CC Call ID              : 251
No.         Timestamp         Digit           Duration
==========================================================

Call 2
SIP Call ID                : 29BB98C-F01311E3-8297DE9F-78C438FF@10.86.176.119
   State of the call       : STATE_ACTIVE (7)
   Calling Number          : 2017
   Called Number           : 1011
   CC Call ID              : 252
No.         Timestamp         Digit           Duration
==========================================================

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

CHAPTER 8

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 and 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 of digital voice samples using codecs and how to configure them.

Why CUBE Needs Codecs

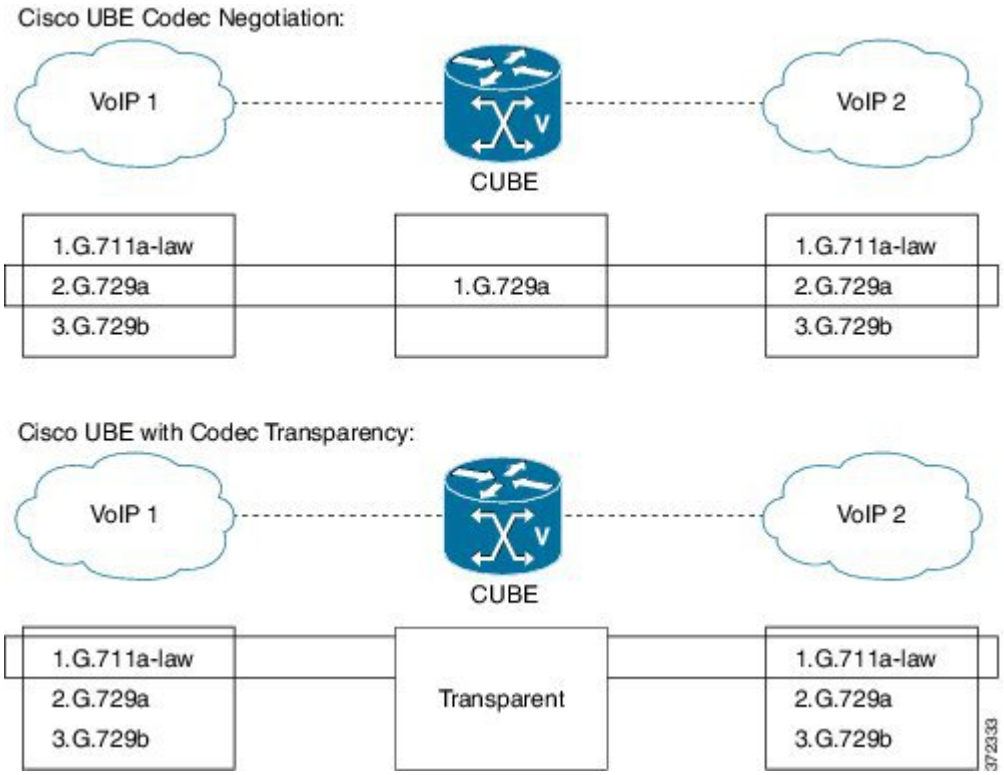
CUBE uses codecs to compress digital voice samples to reduce bandwidth usage per call. Refer to Table 13: Codec and Bandwidth Information, on page 50 to see the relationship between codec and bandwidth utilization.

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.

Figure 11: Codec Negotiation on CUBE



In the first example, the CUBE router is configured to use the G.729a codec. This can be done by using the appropriate codec command on both VoIP dial peers. When a call is set up, CUBE will accept only G.729a calls, thus influencing the codec negotiation.

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

Voice Media Transmission

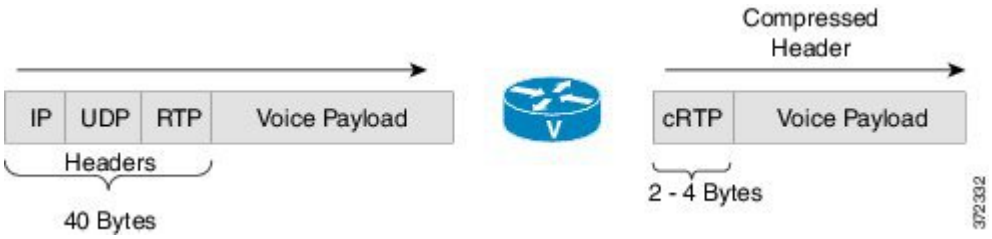
When a VoIP call is established, using the signaling protocols, the digitized voice samples need to be transmitted. These voice samples are often called the voice media. Voice media protocols found in a VoIP environment are the following:

- Real-Time Transport Protocol (RTP)—RTP is a Layer 4 protocol that is encapsulated inside UDP segments. RTP carries the actual digitized voice samples in a call.
- Real-Time Control Protocol (RTcP)—RTcP is a companion protocol to RTP. Both RTP and RTcP operate at Layer 4 and are encapsulated in UDP. RTP and RTCP typically use UDP ports 16384 to 32767, though these ranges may vary according to hardware platform. However, RTP uses the even port

numbers in that range, whereas RTcP uses the odd port numbers. While RTP is responsible for carrying the voice stream, RTcP carries information about the RTP stream such as latency, jitter, packets, and octets sent and received.

- Compressed RTP (cRTP)—One of the challenges with RTP is its overhead. Specifically, the combined IP, UDP, and RTP headers are approximately 40 bytes in size, whereas a common voice payload size on a VoIP network is only 20 bytes, which includes 20 ms of voice by default. In that case, the header is twice the size of the payload. cRTP is used for RTP header compression and can reduce the 40-byte header to 2 or 4 bytes in size (depending on whether UDP checksums are in use), as shown in the figure below.

Figure 12: Compressed RTP



- Secure RTP (sRTP)—To help prevent an attacker from intercepting and decoding or possibly manipulating voice packets, sRTP supports encryption of RTP packets. In addition, sRTP provides message authentication, integrity checking, and protection against replay attacks.
  VPN technology like IP Security (IPSec) may be used to protect traffic between sites. Encrypting sRTP traffic at the source of transmission results in encrypting already encrypted traffic, adding significant overhead and bandwidth needs. So it is recommended that sRTP is used for voice traffic, and that this traffic is excluded from IPSec encapsulation. sRTP uses lesser bandwidth, has the same level of security, and can be used by devices at any location because the payload is originated and terminated at the voice endpoint. Because endpoints can be mobile, the security follows the phone.

Voice Activity Detection

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

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

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

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

VoIP Bandwidth Requirements

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

Table 12: Protocol Header Size Assumptions

Protocol	Header size
IP	20 bytes
UDP	8 bytes
RTP	12 bytes
cRTP	Reduces size of IP, UDP, RTP to 2 or 4 bytes

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 13: Codec and Bandwidth Information

Codec & Bit Rate (kbps)	Codec Sample Size (Bytes)	Codec Sample Interval (ms)	Mean Opinion Score (MOS)	Voice Payload Size (Bytes)	Voice Payload Size (ms)	Payload Size (ms) Packets Per Second (PPS)	Bandwidth MP or FRF.12 (kbps)	Bandwidth w/cRTP MP or FRF.12 (kbps)	Bandwidth Ethernet (kbps)
G.711 (64 kbps)	80	10	4.1	160	20	50	82.8	67.6	87.2
G.729 (8 kbps)	10	10	3.92	20	20	50	26.8	11.6	31.2
G.723.1 (6.3 kbps)	24	30	3.9	24	30	33.3	18.9	8.8	21.9
G.723.1 (5.3 kbps)	20	30	3.8	20	30	33.3	17.9	7.7	20.8
G.726 (32 kbps)	20	5	3.85	80	20	50	50.8	35.6	55.2
G.726 (24 kbps)	15	5		60	20	50	42.8	27.6	47.2

Codec & Bit Rate (kbps)	Codec Sample Size (Bytes)	Codec Sample Interval (ms)	Mean Opinion Score (MOS)	Voice Payload Size (Bytes)	Voice Payload Size (ms)	Payload Size (ms) Packets Per Second (PPS)	Bandwidth MP or FRF.12 (kbps)	Bandwidth w/cRTP MP or FRF.12 (kbps)	Bandwidth Ethernet (kbps)
G.728 (16 kbps)	10	5	3.61	60	30	33.3	28.5	18.4	31.5
G722_64k(64 kbps)	80	10	4.13	160	20	50	82.8	67.6	87.2
ilbc_mode_20(15.2 kbps)	38	20	NA	38	20	50	34.0	18.8	38.4
ilbc_mode_30(13.33 kbps)	50	30	NA	50	30	33.3	25.867	15.73	28.8

Table 14: Explanation of Terms

Codec Bit Rate (kbps)	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).
Codec Sample Size (Bytes)	Size (Bytes) Based on the codec, this is the number of bytes captured by the digital signal processor (DSP) at each codec sample interval. For example, the G.729 coder operates on sample intervals of 10 ms, corresponding to 10 bytes (80 bits) per sample at a bit rate of 8 kbps. (codec bit rate = codec sample size / codec sample interval).
Codec Sample Interval (ms)	This is the sample interval at which the codec operates. For example, the G.729 coder operates on sample intervals of 10 ms, corresponding to 10 bytes (80 bits) per sample at a bit rate of 8 kbps. (codec bit rate = codec sample size / codec sample interval).
MOS	MOS is a system of grading the voice quality of telephone connections. With MOS, a wide range of listeners judge the quality of a voice sample on a scale of one (bad) to five (excellent). The scores are averaged to provide the MOS for the codec.
Voice Payload Size (Bytes)	The voice payload size represents the number of bytes (or bits) that are filled into a packet. The voice payload size must be a multiple of the codec sample size. For example, G.729 packets can use 10, 20, 30, 40, 50, or 60 bytes of voice payload size.

Voice Payload Size (ms)	Payload Size (ms) The voice payload size can also be represented in terms of the codec samples. For example, a G.729 voice payload size of 20 ms (two 10 ms codec samples) represents a voice payload of 20 bytes [(20 bytes * 8) / (20 ms) = 8 kbps]
PPS	PPS represents the number of packets that need to be transmitted every second in order to deliver the codec bit rate. For example, for a G.729 call with voice payload size per packet of 20 bytes (160 bits), 50 packets need to be transmitted every second [50 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 can be supported:

Table 15: Audio Codecs Supported on CUBE

Codec Keyword	Codec
aacld	AACLD 90000 bps
clear-channel	Clear Channel 64000 bps (No voice capabilities: data transport only)
g711alaw	G.711 A Law 64000 bps
g711ulaw	G.711 u Law 64000 bps
g722-48	G722-48K 64000 bps - Only supported for H.320<->H.323 calls
g722-56	G722-56K 64000 bps - Only supported for H.320<->H.323 calls
g722-64	G722-64K 64000 bps
g723ar53	G.723.1 ANNEX-A 5300 bps (contains built-in VAD that cannot be disabled) Not supported on PVDM3.
g723ar63	G.723.1 ANNEX-A 6300 bps (contains built-in VAD that cannot be disabled) Not supported on PVDM3.
g723r53	G.723.1 5300 bps Not supported on PVDM3.

Codec Keyword	Codec
g723r63	G.723.1 6300 bps Not supported on PVDM3.
g726r16	G.726 16000 bps
g726r24	G.726 24000 bps
g726r32	G.726 32000 bps
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 upto 128 kbps
transparent	Transparent; uses the endpoint codec

Table 16: 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*]]
   - **codec mp4-latm** [**profile** *tag*]
5. Do the following to configure a video codec:
   - **video codec** *codec*
6. (Optional) Do one of the following to configure RTP payload type:
   - **rtp payload-type cisco-codec-isac** *value*
   - **rtp payload-type cisco-codec-ilbc** *value*
   - **rtp payload-type cisco-codec-video-h263+** *value*
   - **rtp payload-type cisco-codec-video-h264** *value*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
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	Purpose
Step 3	**dial-peer voice** *number* **voip** **Example:** `Device(config)# dial-peer voice 1 voip`	Enters dial peer configuration mode for the specified VoIP dial peer.

Step 4	Enter one of the following to configure an audio codec:
	• **codec** *codec* [**bytes** *payload-size* **fixed-bytes**]
	• **codec isac** [**mode {adaptive | independent}** [**bit-rate** *value* **framesize { 30 | 60 } [fixed]**]
	• **codec ilbc** [**mode** *frame-size* [**bytes** *payload-size*]]
	• **codec mp4-latm** [**profile** *tag*]
	
	Example:
	For g711alaw Codec
	`Device(config-dial-peer)# codec g711alaw`
	
	Example:
	For ISAC Codec
	`Device(config-dial-peer)# codec isac mode independent`
		Configures an audio codec at the dial peer level.
		• g729r8, 20-byte payload is configured by default.

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	Do one of the following to configure RTP payload type: • **rtp payload-type cisco-codec-isac** *value* • **rtp payload-type cisco-codec-ilbc** *value* • **rtp payload-type cisco-codec-video-h263+** *value* • **rtp payload-type cisco-codec-video-h264** *value*	(Optional) 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*[**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

	Command or Action	Purpose
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	**voice class codec** *tag* **Example:** `Device(config)# voice class codec 10`	Enters voice-class configuration mode for the specified codec voice class.
Step 4	Do the following for each audio codec you want to configure in the voice class: • **codec preference** *value codec-type*[**bytes** *payload-size* **fixed-bytes**]	Configure a codec within the voice class and specifies a preference for the codec. This becomes part of a preference list

	Command or Action	Purpose
	• **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*]	
Step 5	**exit** **Example:** `Device(config-class)# exit`	Exits the current mode. • Enter your password if prompted.
Step 6	**dial-peer voice** *number* **voip** **Example:** `Device(config)# dial-peer voice 1 voip`	Enters dial peer configuration mode for the specified VoIP dial peer.
Step 7	**voice-class codec** *tag* **offer-all** **Example:** `Device(config-dial-peer)# voice-class codec 10`	Applies the previously configured voice class and associated codecs to a dial peer. • The **offer-all** keyword allows the device to offer all codecs configured in a codec voice class.
Step 8	**end** **Example:** `Device(config-dial-peer)# end`	Returns to privileged EXEC mode.

Configuring Video Codecs Using Codec Voice Class

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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	**voice class codec** *tag* **Example:** `Device(config)# voice class codec 10`	Enters voice-class configuration mode for the specified codec voice class.
Step 4	**video codec** *codec* **Example:** `video codec h261`	Configures a video codec within the voice class.
Step 5	**exit** **Example:** `Device(config-class)# exit`	Exits the current mode. • Enter your password if prompted.
Step 6	**dial-peer voice** *number* **voip** **Example:** `Device(config)# dial-peer voice 1 voip`	Enters dial peer configuration mode for the specified VoIP dial peer.
Step 7	**voice-class codec** *tag* **offer-all** **Example:** `Device(config-dial-peer)# 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.
Step 8	**end** **Example:** `Device(config-dial-peer)# end`	Returns to privileged EXEC mode.

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
```

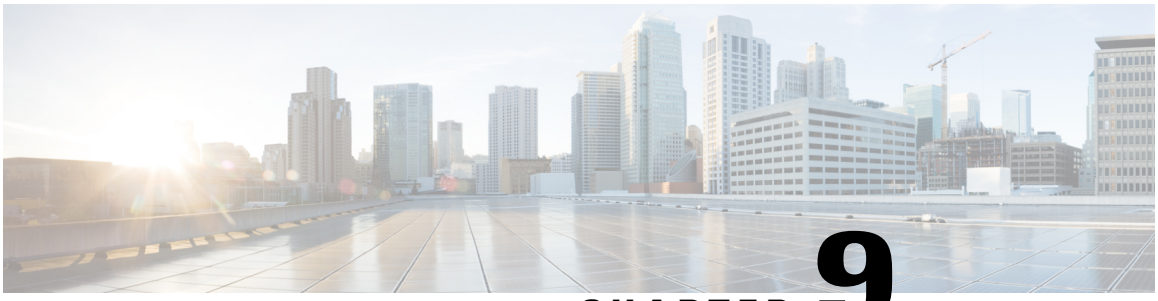
Configuration Examples for Codecs

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
```



CHAPTER 9

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

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 17: Feature Information for SIP Binding

Feature Name	Releases	Feature Information
SIP Gateway Support for the bind Command	Cisco IOS 12.2(2)XB, 12.2(2)XB2, 12.2(8)T, 12.2(11)T, and 12.3(4)T Cisco IOS XE 3.1.0S	The SIP Gateway Support for the **bind** Command feature allows you to configure the source IP address of signaling packets and media packets. In 12.2(2)XB, this feature was introduced. In 12.3(4)T, this feature was expanded to provide the flexibility to specify different source interfaces for signaling and media, and allow network administrators a finer granularity of control on the network interfaces used for voice traffic. The following commands were introduced or modified: **bind**, **show dial-peer voice**, **show ip sockets**, **show sip-ua connections**, and **show sip-ua status**.
Support for Ability to Configure Source IP Address for Signaling and Media per SIP Trunk	15.1(2)T	This feature allows you to configure a separate source IP address per SIP trunk. This source IP address is embedded in all SIP signaling and media packets that traverse the SIP trunk. This feature enables service providers for better profiling and billing policies. It also enables greater security for enterprises by the use of distinct IP addresses within and outside the enterprise domain. The following command was introduced or modified: **voice-class sip bind**.

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.

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 Qualtiy of Service (QoS) to be monitored.
- The **bind media** command relaxes the constraints imposed by the **bind control** and **bind all** commands, which cannot be set during an active call. The **bind media** command works with active calls.

Source Address

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

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

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

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

- Bind configuration at dial peer level

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

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

Table 18: State of the System for the bind Address

Bind State	System Status
No global bind	The best local address is used in all outbound SIP messages. Only one SIP listen socket with a wildcard source address.
Global bind	Global bind address used in all outbound SIP messages. Only one SIP listen socket with global bind address.
No global bind Dial peer bind	Dial peer bind address is used in outbound SIP messages of this dial peer. The remaining SIP messages use the best local address. One SIP listen socket with a wildcard source address. Additional SIP listen socket for each different dial peer bind listening on the specific dial peer bind address.
Global bind Dial peer bind	Dial peer bind address is used in outbound SIP messages of this dial peer. The remaining SIP messages use the global bind address. One SIP listen socket with global bind address. Additional SIP listen socket for each different dial peer bind command listening on the specific dial peer bind address.

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

Table 19: State of the Interface for the bind Command

Interface State	Result Using Bind Command
Shut down With or without active calls	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 **bind** command enabled and has an active call, the call becomes a one-way call with media flowing from the outgoing gateway to the terminating gateway. The dial peer bind socket listeners of the interface are closed and the configuration turns inactive for all subsequent SIP messages.
No shut down No active calls	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 **bind** command. The sockets accept packets destined for the bound address only. The dial peer bind socket listeners of the interface are reopened and the configuration turns active for all subsequent SIP messages.
No shut down Active calls	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.

Interface State	Result Using Bind Command
Bound-interface 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. 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 **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 is 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 **no shutdown** interfaces. 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. 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.

Interface State	Result Using Bind Command
Note If there are active calls, the **bind** command does not take effect if it is issued for the first time or if another **bind** command is in effect. A message reminds you that there are active calls and that the change cannot take effect.	

The **bind** command applied at the dial peer level can be modified only in the following situations:

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

Voice Media Stream Processing

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

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

Table 20: Interaction Between Previously Set and New bind Commands

Interface State	bind Command	Result Using bind Command
Without active calls	**bind all**	Generated **bind control** and **bind media** commands to override existing **bind control** and **bind media** commands.
	bind control	Overrides existing **bind control** command.
	bind media	Overrides existing **bind media** command.

Interface State	bind Command	Result Using bind Command
With active calls	**bind all** or **bind control**	Blocks the command, and the following messages are displayed: • 00:16:39: There are active calls • 00:16:39: configure_sip_bind_command: The bind command change will not take effect
	bind media	Succeeds and overrides any existing **bind media** command.

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

Note The **bind all** command only applies to global level, whereas the **bind control** and **bind media** command apply to global and dial peer. The table below applies to **bind media** only if the media interface is the same as the **bind control** interface. If the two interfaces are different, media behavior is independent of the interface state.

Table 21: bind all and bind control Functions, Based on Interface State

Interface State	Result Using bind all or bind control Commands
Shut down With or without active calls	TCP, TLS, and UDP socket listeners are initially closed. (Socket listeners receive datagrams addressed to the socket.) Then the sockets are opened to listen to any IP address. If the outgoing gateway has the **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.

Interface State	Result Using bind all or bind control Commands
Not shut down Without active calls	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 **bind** command. The sockets accept packets destined for the bound address only. The dial peer bind socket listeners of the interface are reopened and the configuration turns active for all subsequent SIP messages.
Not shut down With active calls	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.
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.

Interface State	Result Using bind all or bind control Commands
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.

Configuring SIP Binding

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface fastethernet0/0`	Configures an interface type and enters the interface configuration mode. • *type number* —Type of interface to be configured and the port, connector, or interface card number.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 192.168.200.33 255.255.255.0`	Configures a primary or secondary IP address for an interface.
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits the current mode.

Step 6	Use one of the following commands to configure SIP binding:
	• **bind** {**control** | **media** | **all**} **source-interface** *interface-id* [**ipv6-address** *ipv6-address*] in SIP configuration mode.
	• **voice-class sip bind** {**control** | **media**} **source interface** *interface-id* [**ipv6-address** *ipv6-address*] in dial-peer configuration mode.
	Example:
	SIP binding in SIP configuration mode:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# sip`
	`Device(conf-serv-sip)# bind control source-interface FastEthernet0/0`
	`Device(conf-serv-sip)# exit`
	Example:
		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.
		• **control** —Binds signaling packets.
		• **media** —Binds media packets.
		• **all** —Binds signaling and media packets.
		• **source interface** *interface-id* —Type of interface and its ID.
		• **ipv6-address** *ipv6-address* —Configures the IPv6 address. Ensure that the IPv6 address has been applied to an interface.

	Command or Action	Purpose

	SIP binding in dial-peer configuration mode:
	
	`Device(config)# dial-peer voice 100 voip`
	`Device(config-dial-peer)# session protocol sipv2`
	`Device(config-dial-peer)# voice-class sip bind control source-interface fastethernet0/0`
	`Device(config-dial-peer)# exit`	

Step 7	**end**	Exits to privileged EXEC mode.

Verifying SIP Binding

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **show ip sockets**

Use this command to display IP socket information and indicate whether the bind address of the receiving gateway is set.

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

Example:

```
Device# show ip sockets

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

Example:

Step 2 **show sip-ua status**

Use this command to display SIP user-agent status and indicate whether bind is enabled.

The following sample output indicates that signaling is disabled and media on 172.18.192.204 is enabled:

Example:

```
Device# show sip-ua status
SIP User Agent Status
SIP User Agent for UDP : ENABLED
SIP User Agent for TCP : ENABLED
SIP User Agent for TLS over TCP : ENABLED
SIP User Agent bind status(signaling): DISABLED
SIP User Agent bind status(media): ENABLED 172.18.192.204
SIP early-media for 180 responses with SDP: ENABLED
SIP max-forwards : 70
SIP DNS SRV version: 2 (rfc 2782)
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 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 udptl
```

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

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

Example:

```
Device# show sip-ua connections udp detail

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

Step 4 **show dial-peer voice**

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

Example:

```
Device# show dial-peer voice 101

VoiceOverIpPeer1234
        peer type = voice, system default peer = FALSE, information type = voice,
        description = `',
        tag = 1234, destination-pattern = `',
        voice reg type = 0, corresponding tag = 0,
        allow watch = FALSE
        answer-address = `', preference=0,
        CLID Restriction = None
        CLID Network Number = `'
        CLID Second Number sent
        CLID Override RDNIS = disabled,
        rtp-ssrc mux = system
        source carrier-id = `', target carrier-id = `',
        source trunk-group-label = `',  target trunk-group-label = `',
        numbering Type = `unknown'
        group = 1234, Admin state is up, Operation state is down,
        incoming called-number = `', connections/maximum = 0/unlimited,
        DTMF Relay = disabled,
        modem transport = system,
        URI classes:
            Incoming (Request) =
            Incoming (Via) =
            Incoming (To) =
            Incoming (From) =
            Destination =
        huntstop = disabled,
        in bound application associated: 'DEFAULT'
        out bound application associated: ''
        dnis-map =
        permission :both
        incoming COR list:maximum capability
        outgoing COR list:minimum requirement
        outgoing LPCOR:
        Translation profile (Incoming):
        Translation profile (Outgoing):
        incoming call blocking:
        translation-profile = `'
        disconnect-cause = `no-service'
        advertise 0x40 capacity_update_timer 25 addrFamily 4 oldAddrFamily 4
        mailbox selection policy: none
        type = voip, session-target = `',
        technology prefix:
        settle-call = disabled
        ip media DSCP = ef, ip media rsvp-pass DSCP = ef
        ip media rsvp-fail DSCP = ef, ip signaling DSCP = af31,
        ip video rsvp-none DSCP = af41,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-ld=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 = 19
```

```
fax rate = voice,   payload size =  20 bytes
fax protocol = system
fax-relay ecm enable
Fax Relay ans enabled
Fax Relay SG3-to-G3 Enabled (by system configuration)
fax NSF = 0xAD0051 (default)
codec = g729r8,   payload size =  20 bytes,
video codec = None
voice class codec = `'
voice class sip session refresh system
voice class sip rsvp-fail-policy voice post-alert mandatory keep-alive interval 30
voice class sip rsvp-fail-policy voice post-alert optional keep-alive interval 30
voice class sip rsvp-fail-policy video post-alert mandatory keep-alive interval 30
voice class sip rsvp-fail-policy video post-alert optional keep-alive interval 30
text relay = disabled
Media Setting = forking (disabled) flow-through (global)
Expect factor = 10, Icpif = 20,
Playout Mode is set to adaptive,
Initial 60 ms, Max 1000 ms
Playout-delay Minimum mode is set to default, value 40 ms
Fax nominal 300 ms
Max Redirects = 1, signaling-type = cas,
VAD = enabled, Poor QOV Trap = disabled,
Source Interface = NONE
voice class sip url = system,
voice class sip tel-config url = system,
voice class sip rel1xx = system,
voice class sip anat = system,
voice class sip outbound-proxy = "system",
voice class sip associate registered-number =
                system,
voice class sip asserted-id system,
voice class sip privacy system
voice class sip e911 = system,
voice class sip history-info = system,
voice class sip reset timer expires 183 = system,
voice class sip pass-thru headers = system,
voice class sip pass-thru content unsupp = system,
voice class sip pass-thru content sdp = system,
voice class sip copy-list = system,
voice class sip g729 annexb-all = system,
voice class sip early-offer forced = system,
voice class sip negotiate cisco = system,
voice class sip block 180 = system,
voice class sip block 183 = system,
voice class sip block 181 = system,
voice class sip preloaded-route = system,
voice class sip random-contact = system,
voice class sip random-request-uri validate = system,
voice class sip call-route p-called-party-id = system,
voice class sip call-route history-info = system,
voice class sip privacy-policy send-always = system,
voice class sip privacy-policy passthru = system,
voice class sip privacy-policy strip history-info = system,
voice class sip privacy-policy strip diversion = system,
voice class sip map resp-code 181 = system,
voice class sip bind control = enabled, 9.42.28.29,
voice class sip bind media = enabled, 9.42.28.29,
voice class sip bandwidth audio = system,
voice class sip bandwidth video = system,
voice class sip encap clear-channel = system,
voice class sip error-code-override options-keepalive failure = system,
voice class sip calltype-video = false
voice class sip registration passthrough = System
voice class sip authenticate redirecting-number  = system,
redirect ip2ip = disabled
local peer = false
probe disabled,
Secure RTP: system (use the global setting)
voice class perm tag = `'
Time elapsed since last clearing of voice call statistics never
```

```
Connect Time = 0, Charged Units = 0,
Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
Accepted Calls = 0, Refused Calls = 0,
Last Disconnect Cause is "",
Last Disconnect Text is "",
Last Setup Time = 0.
Last Disconnect Time = 0.
```

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

Step 5 **show running-config**

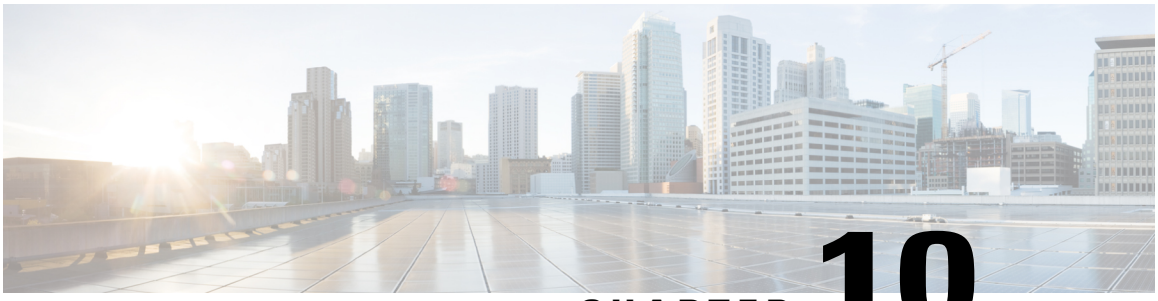
Although the bind all command is an accepted configuration, it does not appear in **show running-config** command output. Because the **bind all** command is equivalent to issuing the commands **bind source** and **bind media**, those are the commands that appear in the **show running-config** command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



CHAPTER 10

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

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.

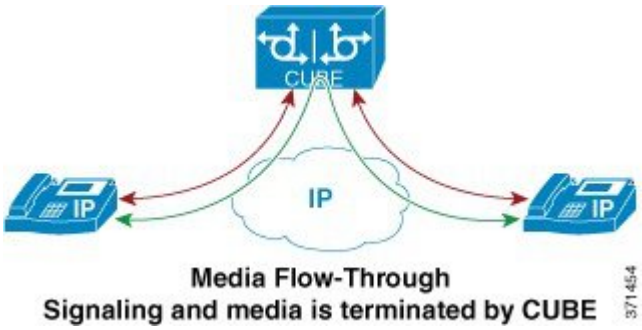
Table 22: Feature Information for Configuring Path of Media

Feature Name	Releases	Feature Information
Configuring Media Path	12.4(3), 12.4(24)T, 15.0(1)M	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**.

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 13: Media Flow-Through Mode



Restrictions for Media Flow-Through

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

Configuring Media Flow-Through

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

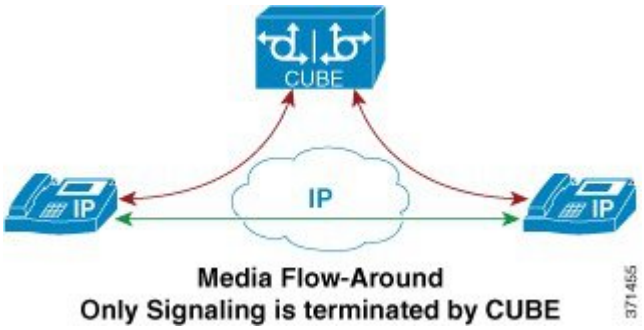
Step 3	Use one of the following commands to configure media flow-through:
	• **media flow-through** in dial-peer configuration mode
	• **media flow-through** in global VoIP configuration mode
	
	Example:
	In dial-peer configuration mode
	`! Applying SIP profiles to one dial peer only`
	`Device (config)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

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 14: Media Flow-Around



Configuring Media Flow-Around

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

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

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

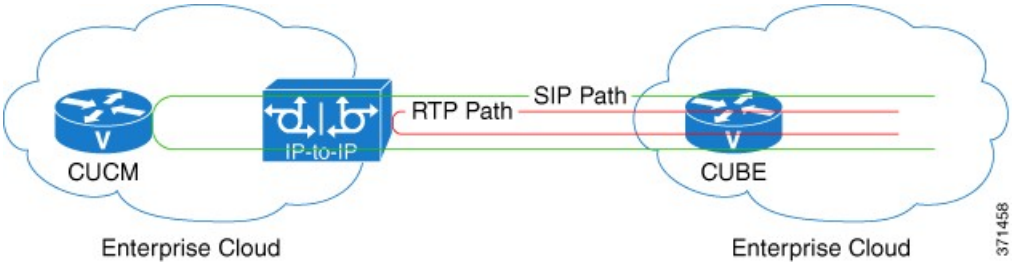
Media Anti-Trombone

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

The user agent may initiate call forwards and call transfers that are sent towards CUBE as a new SIP INVITE dialog. CUBE considers the original call and the forwarded call as separate unrelated calls. Media anti-tromboning allows CUBE to detect the relation between the calls and resolve the media loop by sending SDP packets back to the sender.

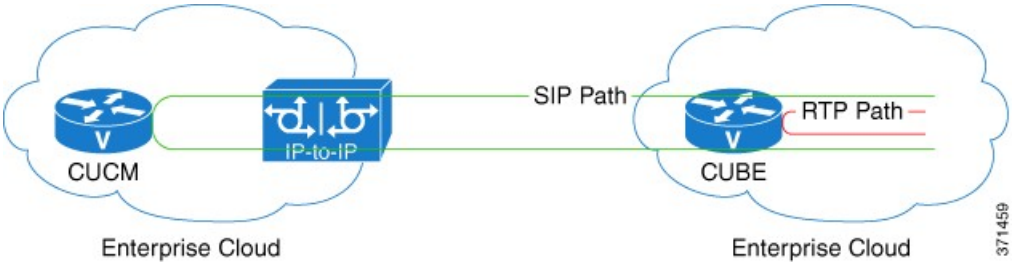
The figure below illustrates how CUBE needlessly loops RTP packets towards the User Agent because it fails to detect the loop.

Figure 15: Tromboning - Needless looping of Media Packets



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

Figure 16: Anti-Tromboning - Avoiding Media Loops



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

SUMMARY STEPS

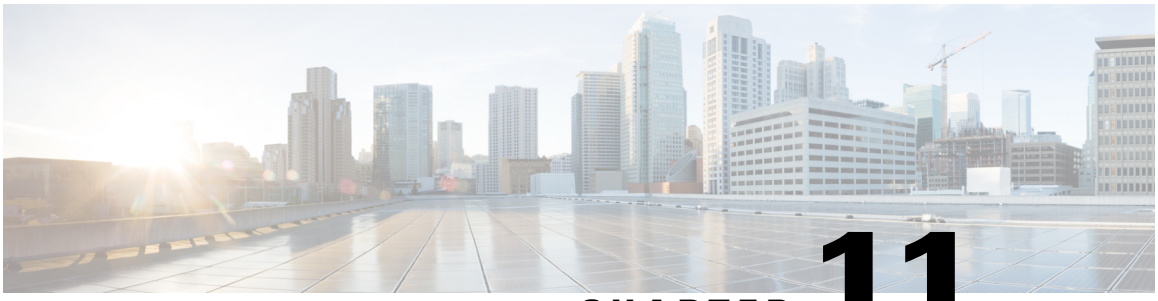
1. **enable**
2. **configure terminal**
3. Enter one of the following commands to configure media anti-tromboning:
   - **media anti-trombone** in dial-peer configuration mode
   - **media anti-trombone** in global VoIP configuration mode
4. **end**

DETAILED STEPS

	Command or Action	Purpose
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 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`
	
	Example:
	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`
		Enables media anti-trombone for all calls.

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



CHAPTER 11

SIP Profiles

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

SIP profiles can be configured 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 also be 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 17: Incoming and Outgoing messages where SIP Profiles can be applied

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 23: Feature Information for SIP Profiles

Feature Name	Releases	Feature Information
SIP Profiles (for inbound messages)	Cisco IOS 15.4(2)T Cisco IOS XE 3.12S	This feature extends support to inbound messages. This feature modifies the following commands: The **inbound** keyword was added to the **sip-profiles** and **voice-class sip profiles** commands.
Support for Rotary calls and Media Forking	Cisco IOS 15.3(1)T	With CSCty41575, this feature was enhanced to support forked and rotary calls.
Configuring SIP Profile (Add, Delete or Modify)	Cisco IOS 12.4(15)XZ Cisco IOS 12.4(20)T Cisco IOS XE 2.5	This feature allows users to change (add, delete, or modify) the standard SIP messages that are sent or received for better interworking with different SIP entities. This feature introduces the following commands: **voice class sip-profiles**, **response**, **request**.
Support for Non-Standard SIP Headers	Cisco IOS 15.5(2)T	This feature allows users to add, copy, delete, or modify non-standard (for example, X-Cisco-Recording-Participant) using SIP profiles. The **word** keyword was added to the **sip-profiles** command to allow the user to configure any non-standard SIP header.

Feature Name	Releases	Feature Information
Support for tagging rules in a SIP profile configuration	Cisco IOS 15.5(2)T Cisco IOS XE 3.15S	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: **rule** 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: **voice sip sip-profiles upgrade**, **voice sip sip-profiles downgrade**
Support for Copying Unsupported SDP Headers	Cisco IOS 15.6(1)T Cisco IOS XE 3.17S	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: **pass-thru content custom-sdp** .

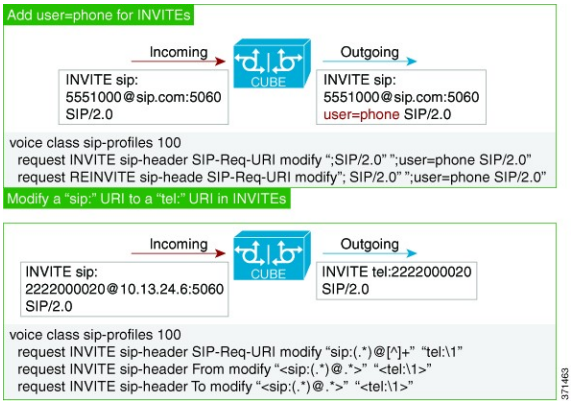
Information About SIP Profiles

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 18: SIP Profile



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

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

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

The SIP profiles feature on CUBE provides a solution to these incompatibilities and customization issues.

SIP profiles can also be used to change a header name from the long form to the compact form. For example, From to f. This can be used as a way to reduce the length of a SIP message. By default, the device never sends the compact form of the SIP messages although it receives either the long or the short form.

Important Characteristics of SIP Profiles

Given below are a few important notes for SIP Profiles:

- Copy Variables u01 to u99 are shared by inbound and outbound SIP Profiles.
- Session Initiation Protocol (SIP) and Session Description Protocol (SDP) headers are supported. SDP can be either a standalone body or part of a Multipurpose Internet Mail Extensions (MIME) message.

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

Inbound SIP Profile:

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

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

Restrictions for SIP Profiles

- Removal or addition of mandatory headers is not supported. You can only modify mandatory headers Mandatory SIP headers include To, From, Via, CSeq, Call-Id, and Max-Forwards. Mandatory SDP headers include v, o, s, t ,c, and m.
- Addition or removal of entire Multipurpose Internet Mail Extensions (MIME) or (Session Description Protocol) SDP bodies from SIP messages is not supported.
- Syntax checking is not performed on SIP messages after SIP profile rules have been applied. Changes specified in the SIP profile should result in valid SIP protocol exchanges.
- The header length (including header name) after modification should not exceed 300 characters. Max header length for add value is approximately 220 characters. Max SDP length is 2048 characters. If any header length exceeds this maximum value after applying SIP profiles, then the profile is not applied.
- If a header-name is changed to its compact form, SIP profile rules cannot be applied on that header. Thus a SIP profile rule modifying a header name to its compact form must be the last rule on that header.
- We cannot modify the "image" m-line attributes (m=image 16850 udptl t38) using SIP profiles. SIP profiles can be applied only on audio and video m-lines in SDP.
- In a high-availability (HA) scenario, SIP profiles copy variable data is not check-pointed to standby.
- Existing limitations and restrictions of outbound SIP profiles apply to inbound SIP profiles as well.
- You cannot configure more than 99 variables for the SIP profiles copy option.
- Once a SIP profile is configured using rule tag, you cannot add rules without tags in the same profile and vice-versa.

How to Configure SIP Profiles

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

Configuring a SIP Profile to Manipulate SIP Request or Response Headers

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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* **Example:** `Device(config)# voice class sip-profiles 10`	Creates a SIP Profiles and enters voice class configuration mode.
Step 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**	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.

	Command or Action	Purpose
	• **request** *message* {**sip-header** | **sdp-header**} *header-to-modify* **modify** *header-value-to-match header-value-to-replace*	• If **ANY** is used in place of a header, it indicates that a rule must be applied to any message within the specified category. • For *header-value-to-add* used to add a header, *header-value-to-match* or *header-value-to-replace* used to modify a header: ◦ If a whitespace occurs, the entire value must be included between double quotes. For example, "User-Agent: CISCO CUBE" ◦ If double quotes occurs, a back slash must prefix the double quotes. For example, "User-Agent: \"CISCO\" CUBE" ◦ Regular expressions are supported.
Step 5	Enter one of the following to add, remove, or modify SIP response headers: • **response** *message* [**method** *method-type*] {**sip-header** | **sdp-header**} *header-to-add* **add** *header-value-to-add* • **response** *message* [**method** *method-type*] {**sip-header** | **sdp-header**} *header-to-remove* **remove** • **response** *message* [**method** *method-type*] {**sip-header** | **sdp-header**} *header-to-modify* **modify** *header-value-to-match header-value-to-replace*	According to your choice, this step does one of the following: • Adds a SIP or SDP header to a SIP response. • Removes a SIP or SDP header to a SIP response. • Modifies a SIP or SDP header to a SIP response. • All notes from the previous step are applicable here.
Step 6	**end**	Exits to privileged EXEC mode

Configuring SIP Profiles for Copying Unsupported SDP Headers

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

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. To enable copying of unsupported SDP attribute from incoming leg to outbound leg, you need to enable one of the following commands:
   - In Global VoIP SIP configuration mode

     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** *indexheader-name* **ADD** *copy-variable*
   - **request/response ANY sdp-header mline-index** *indexheader-name* **MODIFY** *copy-variable* + *replace-pattern*
   - **request/response ANY sdp-header mline-index** *indexheader-name* **REMOVE**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
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	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)	Enables copying of unsupported SDP attributes per m-line to the peer leg so that it can be used in outgoing SIP messages. **Note** Enabling this command does not enable the SDP Passthrough feature.

	Command or Action	Purpose

	voice-class sip pass-thru content custom-sdp
	
	Example:
	In Global VoIP SIP configuration mode:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# sip`
	`Device(conf-serv-sip)# pass-thru content custom-sdp`
	
	Example:
	In Dial-peer configuration mode:
	`Device(config)# dial-peer voice 2 voip`
	`Device(config-dial-peer)# voice-class sip pass-thru content custom-sdp`	

Step 4	**voice class sip-profiles** *profile-id* **Example:** `Device(config)# voice class sip-profiles 10`	Voice class sip-profile is configured on the outbound dial-peer or as a global configuration. Creates a SIP Profile and enters voice class configuration mode.
Step 5	Enter one of the following to copy an unsupported SDP line or attribute from peer leg's SDP and add, modify, or remove in the outgoing SDP: • **request/response ANY peer-header sdp mline-index** *index* **COPY** *match-pattern copy-variable* • **request/response ANY sdp-header mline-index** *indexheader-name* **ADD** *copy-variable* • **request/response ANY sdp-header mline-index** *indexheader-name* **MODIFY** *copy-variable + replace-pattern* • **request/response ANY sdp-header mline-index** *indexheader-name* **REMOVE**	M-line Index values: • 0 - A value of zero represents the session level. • 1 to 6 - A value in the range of one to six represents the m-line number in SDP. Copy: Enables copying of SDP line or attribute from peer leg SDP. Add: Enables adding the copied SDP line or attribute in the outgoing SDP. Modify: Enables modifying SDP line or attribute in the outgoing SDP. Remove: Enables removing SDP line or attribute in the outgoing SDP.
Step 6	**end**	Exits to privileged EXEC mode.

Example: Configuring SIP Profile Rules (Attribute Passing)

```
response ANY peer-header sdp mline-index 4 copy “(a=ixmap:0.*)” u01
response ANY sdp-header mline-index 4 a=ixmap add “\u01”
```

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

	Command or Action	Purpose
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* **Example:** `Device(config)# voice class sip-profiles 10`	Creates a SIP Profile and enters voice class configuration mode.
Step 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*	According to your choice, this step tags the SIP request or response header with a unique number.

	Command or Action	Purpose
Step 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*	According to your choice this steps inserts the rule at the position specified in the **before** *tag*. The subsequent rules in the existing SIP profile configuration is incremented sequentially.
Step 6	Enter the following to delete a rule: • **no rule** *tag*	According to your choice, this step tags the SIP request or response with a unique number.
Step 7	**end**	Exits voice class sip-profiles configuration mode.

Configuring a SIP Profile for Non-standard SIP Header

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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* **Example:** `Device(config)# voice class sip-profiles 10`	Creates a SIP Profiles and enters voice class configuration mode.
Step 4	Enter one of the following to add, copy, remove, or modify non-standard SIP request headers:	According to your choice, this step does one of the following: • Adds a non-standard SIP header to a SIP request.

	Command or Action	Purpose
	• **request** *message* {**sip-header** } *non-standard-header-to-add* **add** *non-standard-header-value-to-add* • **request** *message* {**sip-header** } *non-standard-header-to-copy* **copy** *non-standard-header-value-to-match copy-variable* • **request** *message* {**sip-header** } *non-standard-header-to-remove* **remove** • **request** *message* {**sip-header** } *non-standard-header-to-modify* **modify** *non-standard-header-value-to-match non-standard-header-value-to-replace*	• Copies contents from a non-standard SIP header to a SIP request. • Removes a non-standard SIP header to a SIP request. • Modifies a non-standard SIP header to a SIP request. • If **ANY** is used in place of a header, it indicates that a rule must be applied to any message within the specified category. • For *non-standard-header-value-to-add* used to add a non-standard header, *non-standard-header-value-to-match* or *non-standard-header-value-to-replace* used to modify a non-standard header: ◦ If a whitespace occurs, the entire value must be included between double quotes. For example, “User-Agent: CISCO CUBE” ◦ If double quotes occurs, a back slash must prefix the double quotes. For example, “User-Agent: \”CISCO\” CUBE” ◦ Regular expressions are supported.
Step 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-copy* **copy** *non-standard-header-value-to-match copy-variable* • **response** *message* [**method** *method-type*] {**sip-header**} *non-standard-header-to-remove* **remove** • **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*	According to your choice, this step does one of the following: • Adds a non-standard SIP to a SIP response. • Copies contents from a non-standard SIP header to a SIP response. • Removes a non-standard header to a SIP response. • Modifies a non-standard SIP header to a SIP response. • All notes from the previous step are applicable here.
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

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	Enter the following to upgrade SIP profiles configurations to rule-format: • **voice sip sip-profiles upgrade** **Example:** In EXEC(#) mode: `Device#voice sip sip-profiles upgrade`	Upgrades all SIP Profiles to rule-format configurations.
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`	Downgrades all SIP Profiles from rule-format configurations to non-rule format configurations.

	Command or Action	Purpose
Step 4	**end**	Exits privileged EXEC mode.

What to Do Next

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

Configuring a SIP Profile as an Outbound Profile

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Apply the SIP profile to a dial peer:
   - **voice-class sip profiles** *profile-id* in the dial-peer configuration mode.
   - **sip-profiles** *profile-id* in the global VoIP configuration mode
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal**	Enters global configuration mode.

Step 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
	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)# voice-class sip profiles 30`
	`Device (config-dial-peer)# end`
	Example:	

	Command or Action	Purpose

	In global VoIP SIP mode
	`! 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`	

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

Configuring a SIP Profile as an Inbound Profile

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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal**	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters global VoIP configuration mode.

	Command or Action	Purpose
Step 4	**sip** **Example:** `Device(config-voi-serv)# sip`	Enters global VoIP SIP configuration mode.
Step 5	**sip-profiles inbound** **Example:** `Device(config-voi-sip)# sip-profiles inbound`	Enables inbound SIP profiles feature.

Step 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
	
	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)# voice-class sip profiles 30 inbound`
	`Device (config-dial-peer)# end`
	
	Example:
	In global VoIP SIP mode
	`! Applying SIP profiles globally`
	`Device(config)# voice service voip`
	`Device (config-voi-serv)# sip`
	`Device (config-voi-sip)# sip-profiles 20 inbound`
	`Device (config-voi-sip)# end`	

Step 7	**end**	Exits to privileged EXEC mode

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

debug ccsip all
This command displays the applied SIP profiles.

Example:
Applied SIP profile is highlighted in the example below.

```
Device# debug ccsip all
...
Oct 12 06:51:53.619: //-1/735085DC8F3D/SIP/Info/sipSPIGetShrlPeer:
                     Try match incoming dialpeer for Calling number:
                     : sippOct 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/
                    sip_profiles_update_content_length:
                    Content length header after modification :
                    Content-Length: 279
```

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 Messagesin 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:

```
a=ptime:20
```

Outbound ptime:

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

Example: Modify Audio direction "Audio-Attribute"

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

Case 1:

```
Inbound Audio-Attribute

a=inactive

Outbound Audio-Attribute

a=sendrecv
```

Case 2:

```
Inbound Audio-Attribute

a=recvonly

Outbound Audio-Attribute

a=sendrecv
```

Case 3

```
Inbound Audio-Attribute

a=sendonly

Outbound Audio-Attribute

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

Device(config-class)# response any sdp-header Audio-Attribute modify "a=inactive" "a=sendrecv"
Device(config-class)# response any sdp-header Audio-Attribute modify "a=recvonly" "a=sendrecv"
Device(config-class)# response any sdp-header Audio-Attribute modify "a=sendonly" "a=sendrecv"
```

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

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

Applying the SIP Profiles to Dial Peers

```
! Applying SIP Profiles globally
Device(config)# voice service voip
Device (config-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 sdp-header Audio-Attribute modify "a=inactive"
"a=sendrecv"
Device(config-class)# rule 2 request any sdp-header Audio-Attribute modify "a=recvonly"
"a=sendrecv"
Device(config-class)# end
```

Removing the rule using rule tag

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

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

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

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

Example: Inserting SIP Profile Rules

Example: Inserting a SIP Profile Rule

Inserting a SIP profile rule to a SIP Profile

```
Device(config)#voice class sip-profiles 1
 Device(config-class)#rule 1 request INVITE sip-header Contact Modify "(.*)" "\1;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 "(.*)" "\1;temp=abc"
```

The SIP Profile after inserting the new rule

```
Device(config)#voice class sip-profiles 1
 Device(config-class)#rule 1 request INVITE sip-header Contact Modify "(.*)" "\1;temp=xyz"
 Device(config-class)#rule 2 request INVITE sip-header To Modify "(.*)" "\1;temp=abc"
 Device(config-class)#rule 3 request INVITE sip-header Supported Add "Supported: "
```

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
```

```
Device(config)# voice class sip-profiles 101
Device(config-class)# request Invite sip-header Diversion modify "<sip:5(...)@"
"<sip:9789365\1@"
Device(config-class)# end
```

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(.*)@(.*)"
"From:\1<sip:978\2@\3\x0ADiversion:
<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 sdp-header Audio-Bandwidth-Info add "b=AS:1600“
  request ANY sip-header Cisco-Guid remove
  request INVITE sdp-header Session-Owner modify "CiscoSystems-SIP-GW-UserAgent" "-“
```

The SIP INVITE message before the SIP profile has been applied is 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:1111000010@9.13.40.249>;tag=F11AE0-1D8D
To: <sip:2222000020@9.13.40.250>
Date: Mon, 29 Oct 2007 19:02:04 GMT
Call-ID: 4561B116-858811DC-804DEF2E-4CF2D71B@9.13.40.249
Cisco-Guid: 1163870326-2240287196-2152197934-1290983195
Content-Length: 290

v=0
o=CiscoSystemsSIP-GW-UserAgent 6906 8069 IN IP4 9.13.40.249
s=SIP Call
c=IN IP4 9.13.40.249
t=0 0
m=audio 17070 RTP/AVP 0
c=IN IP4 9.13.40.249
a=rtpmap:0 PCMU/8000
a=ptime:20
```

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

- The Cisco-Guid has been removed.

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

```
INVITE sip:2222000020@9.13.40.250:5060 SIP/2.0
Via: SIP/2.0/UDP 9.13.40.249:5060;branch=z9hG4bK1A203F
From: "sipp " <sip:1111000010@9.13.40.249>;tag=F11AE0-1D8D
To: <sip:2222000020@9.13.40.250>
Date: Mon, 29 Oct 2007 19:02:04 GMT
Call-ID: 4561B116-858811DC-804DEF2E-4CF2D71B@9.13.40.249
Content-Length: 279

v=0
o=- 6906 8069 IN IP4 9.13.40.249
s=SIP Call
c=IN IP4 9.13.40.249
t=0 0
m=audio 17070 RTP/AVP 0
c=IN IP4 9.13.40.249
a=rtpmap:0 PCMU/8000
a=ptime:20
b=AS:1600
```

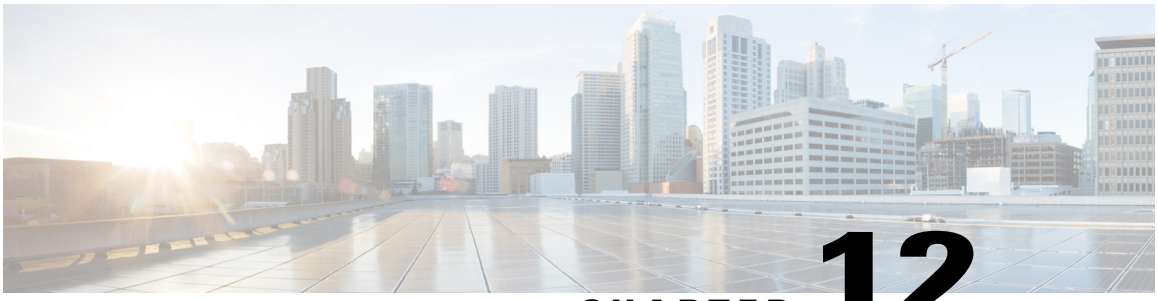
Example: Sample SIP Profile for Non-Standard SIP Headers

Prior to 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.

```
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
```

PART II

Dial Peer Enhancements



CHAPTER 12

Matching Inbound Dial Peers by URI

The Matching Inbound Dial Peers by URI feature allows you to configure selection of inbound dial peers by matching parts of the URI sent by 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, enforcement of specific call-treatment, security, and routing policies on each SIP trunk by originating SIP entity.

In a scenario where multiple SIP hops are involved in a call, there would be multiple via headers involved, and the topmost via header of an incoming SIP invite represents the last hop that forwarded the SIP request, and the bottom-most via header would represent the originator of the SIP request. This feature supports matching by the last hop that forwarded the request (neighboring SIP entity), which is the topmost via header.

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.

Feature Information for Matching Inbound Dial Peers by URI

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

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

Feature Name	Releases	Feature Information
Inbound Dial-peer Match Based on Remote IP Address on SIP Trunks	15.1(2)T Cisco IOS XE Release 3.8S	The Inbound Dial-peer Match Based on Remote IP Address on SIP Trunks feature supports the expansion of inbound dial-peer matching logic to include matching based on the source IP address of inbound signaling on a SIP trunk. This feature enables enforcement of specific call-treatment, security, and routing policies on each SIP trunk. In Cisco IOS Release 15.1(2)T this feature was implemented on the Cisco Unified Border Element. The following commands were introduced or modified: **dial-peer voice**, **voice-class uri**.

Configuring an Inbound Dial Peer to Match on URI

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class uri** *voice-class-uri-tag*
4. Specify a URI field for the voice class:
   - **host** *hostname-pattern*
   - **host ipv4:** *ipv4-address*
   - **host ipv6:** *ipv6-address*
   - **host dns:** *dns-address*
   - **pattern** *uri-pattern*
   - **user-id** *username-pattern*
5. **exit**
6. **dial-peer voice** *tag* **voip**
7. **session protocol sipv2**
8. **incoming uri** { **from** | **request** | **to** | **via**} *voice-class-uri-tag*
9. **end**

DETAILED STEPS

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

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-voice-uri-class)# host server1`
	
	Example:
	
	`Device(config-voice-uri-class)# host ipv4:10.0.0.0`
	
	Example:
	
	`Device(config-voice-uri-class)# host dns:xxx.yyy.com`
		• You can specify up to ten instances of the **host ipv4:**, **host ipv6:**, and **host dns:** commands.
		• You can specify only one instance of the **host** *hostname-pattern* commands.
		• Length of *uri-pattern*, *username-pattern*, and *hostname-pattern* should be less than 32.
		• *username-pattern* is matched against the username field of the URI.
		• *hostname-pattern* is matched against the host field of the URI.
		• *uri-pattern* is matched against the entire URI.
		• Only one instance of the **pattern** and **host** commands are possible.

Step 5	**exit** **Example:** `Device(config-voice-uri-class)# exit`	Enters global configuration mode.

	Command or Action	Purpose
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:123@1.2.3.4:5060 SIP/2.0
Via: SIP/2.0/TCP 10.10.10.1:5093;branch=z9hG4bK-17716-1-0
Via: SIP/2.0/TCP 10.10.14.20:5093;branch=z9hG4bK-28280-1-0
```

Matching Against DNS Name and VIA

CUBE is configured to use incoming dial-peer 102 for incoming SIP calls from sample.com or an IP address that represents one of the resolved IP address of sample.com.

```
voice class uri 202 sip
host dns:sample.com
```

```
dial-peer voice 101 voip
 session protocol sipv2
 incoming uri via 202
```

Incoming INVITE that can be matched against this dial peer.

```
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.

Matching Against Multiple Attributes and VIA

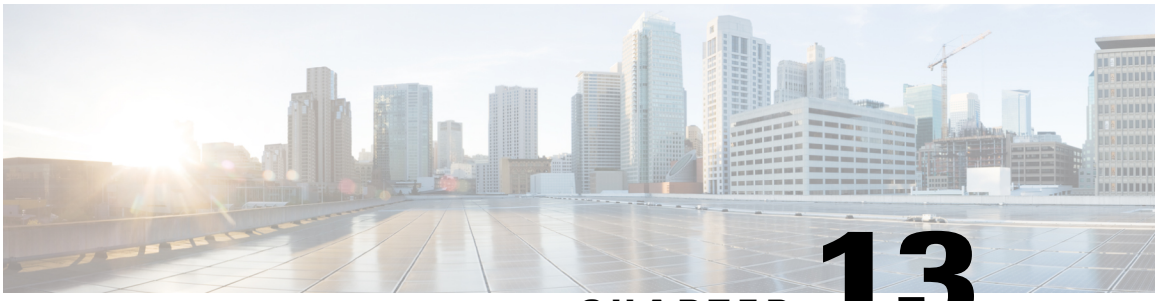
CUBE is configured to use incoming dial-peer 103 for incoming SIP calls from xxx.yyy.com, abc.def.com and IP addresses 10.10.10.10, 10.9.10.11 and 10.10.10.10.

```
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 10.10.10.10:5093;branch=z9hG4bK-17716-1-0
Via: SIP/2.0/TCP 10.10.14.20:5093;branch=z9hG4bK-28280-1-0
```

10.10.10.25 is a resolved IP address of sample.com.



CHAPTER 13

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 (Cisco UBE) for Session Initiation Protocol (SIP) calls. The URI-Based Dialing Enhancements feature includes support for call routing on Cisco UBE when the user part of the incoming Request-URI is non-E164 (for example, INVITE sip:user@abc.com).

Finding Feature Information

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

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

Information About URI-Based Dialing Enhancements

Cisco Unified Communications Manager (CUCM) supports dialing using directory Uniform Resource Identifiers (URIs) for call addressing. Directory URIs follow the username@host format where the host portion is an IPv4 address or a fully qualified domain name. A directory URI is a string of characters that can be used to identify a directory number. If that directory number is assigned to a phone, CUCM can route calls to that phone using the directory URI. URI dialing is available for Session Initiation Protocol (SIP) and Signaling Connection Control Part (SCCP) endpoints that support directory URIs.

The primary use of URI-based dialing is peer-to-peer calling between enterprises using complete URI addresses (that is, 'username@host'). The host part of the URI identifies the destination to which the call should be routed. In earlier Cisco Unified Border Element (Cisco UBE) URI routing, the URI was replaced in the SIP header with the destination server IP address. Then routing of calls was based on the following restrictions:

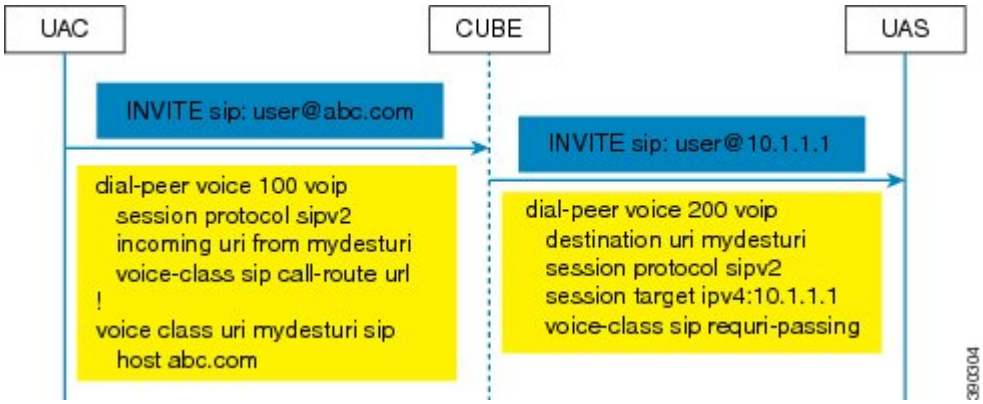
- The user part of the incoming Request-URI must be an E164 number.
- The outgoing Request-URI is always set to the session target information of the outbound dial peer.

The URI-Based Dialing Enhancements feature extends support for Cisco UBE URI-based routing of calls. With these enhancements Cisco UBE supports:

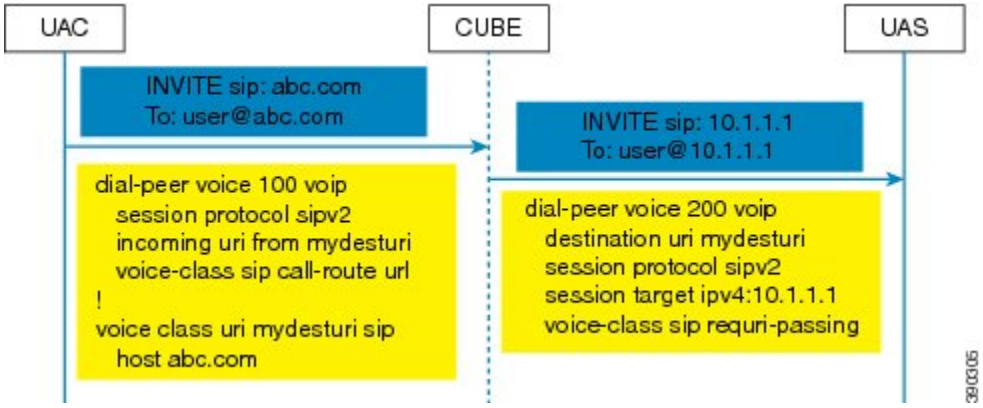
- URI-based routing when the user part of the incoming Request-URI is non-E164 (for example, INVITE sip:user@abc.com).
- URI-based routing when the user part is not present. The user part is an optional parameter in the URI (for example, INVITE sip:abc.com).
- Copying the outgoing Request-URI and To header from the inbound Request-URI and To header respectively.
- Deriving (optionally) the session target for the outbound dial peer from the host portion of the inbound URI.
- URI-based routing for 302, Refer, and Bye Also scenarios.
- Call hunting where the subsequent dial peer is selected based on URI.
- Pass through of 302, with the host part of Contact: unmodified.

Call Flows for URI-Based Dialing Enhancements

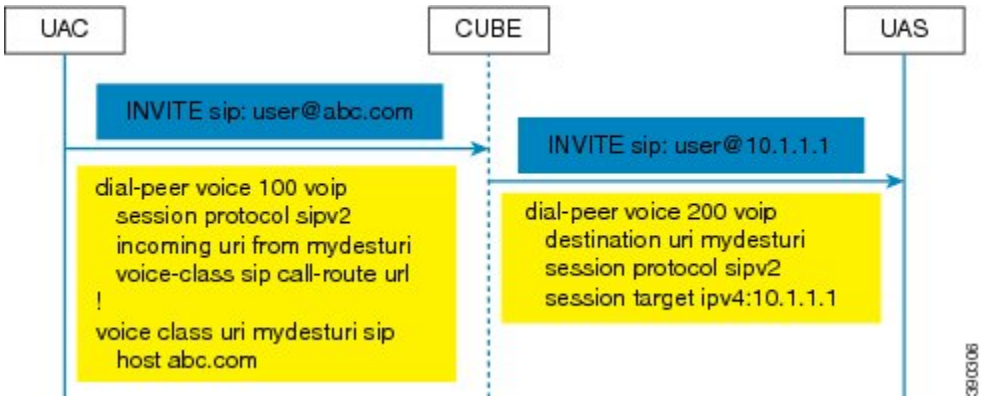
Case1: URI dialing with username being E164 or non-E164 number and Request-URI host copied from the inbound leg.



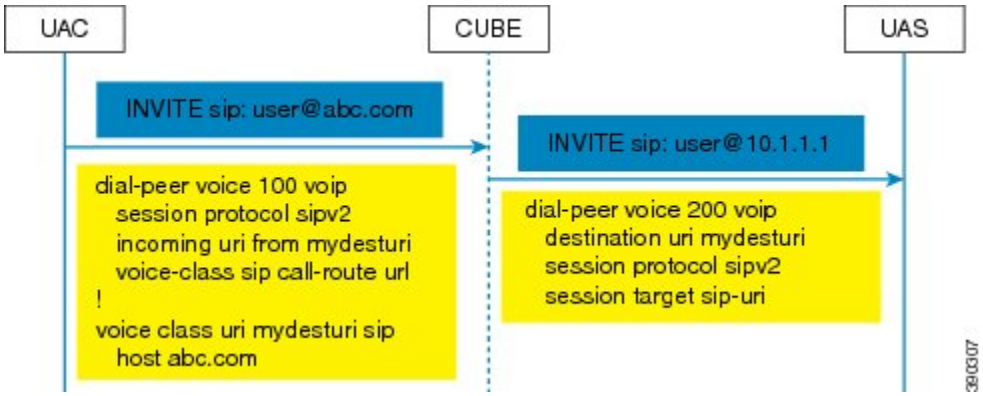
Case 2: Incoming Request-URI does not contain user part. The To: header information is also copied from the peer leg when the **requri-passing** command is enabled.



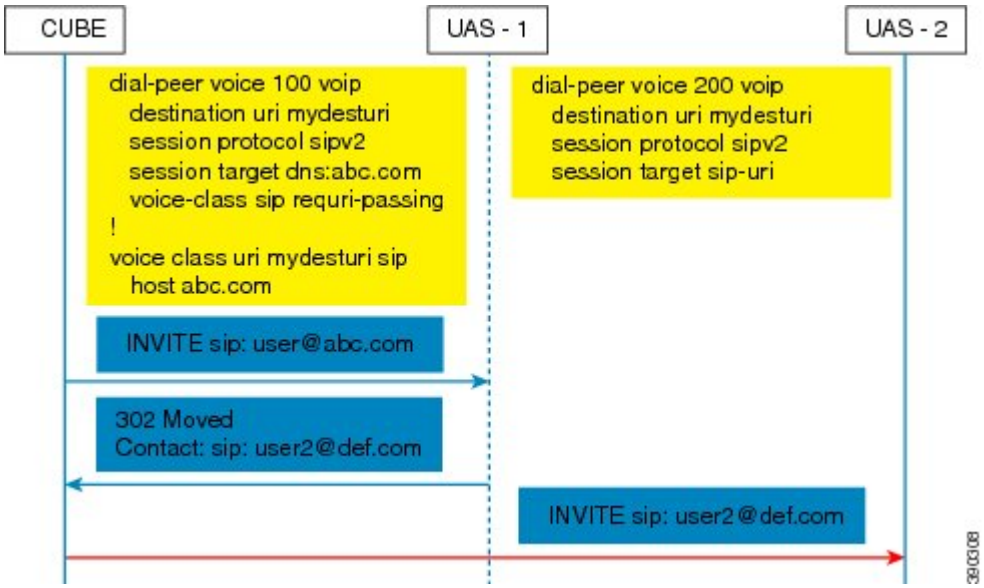
Case 3: The old behavior of setting the outbound Request-URI to session target is retained when the **requri-passing** command is not enabled.



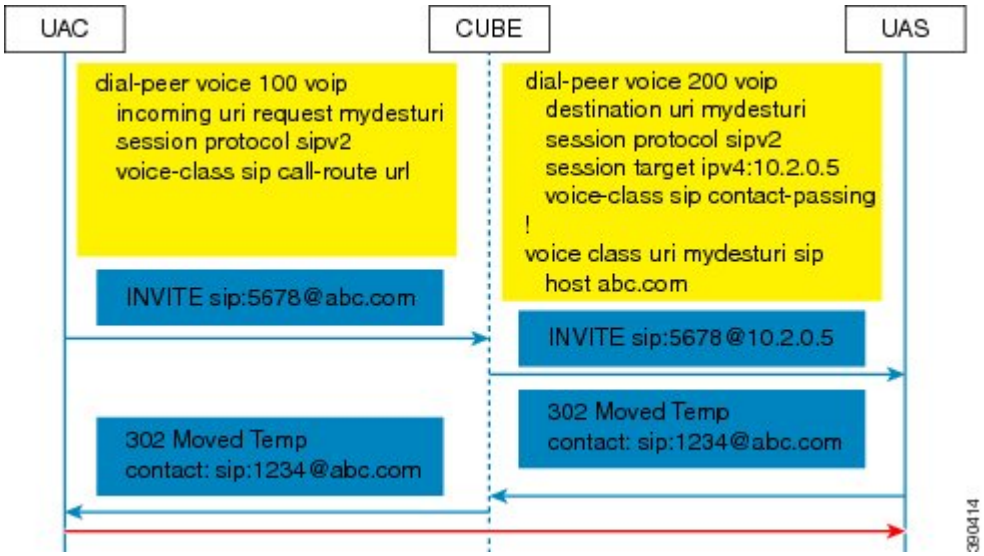
Case 4: The session target derived from the host part of the URI. The outgoing INVITE is sent to resolved IP address of the host part of the URI.



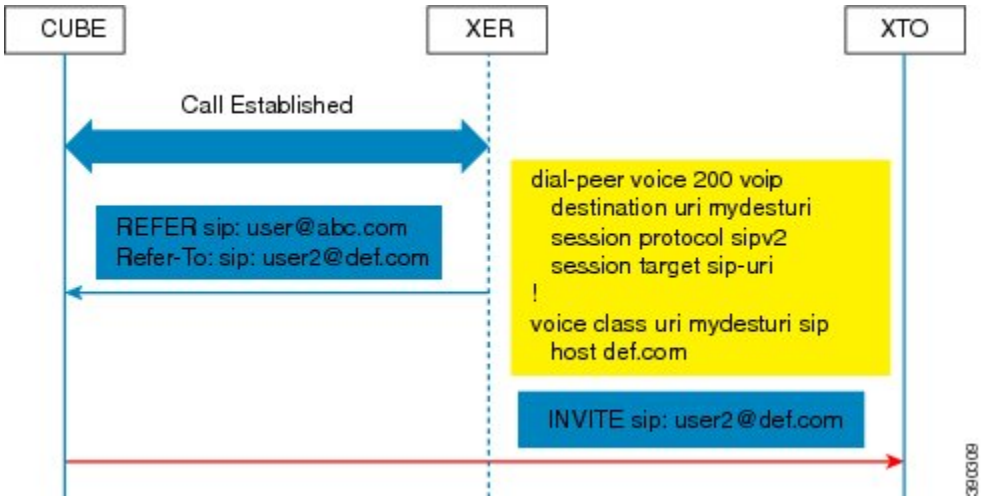
Case 5: Pass through of contact URI to request URI.



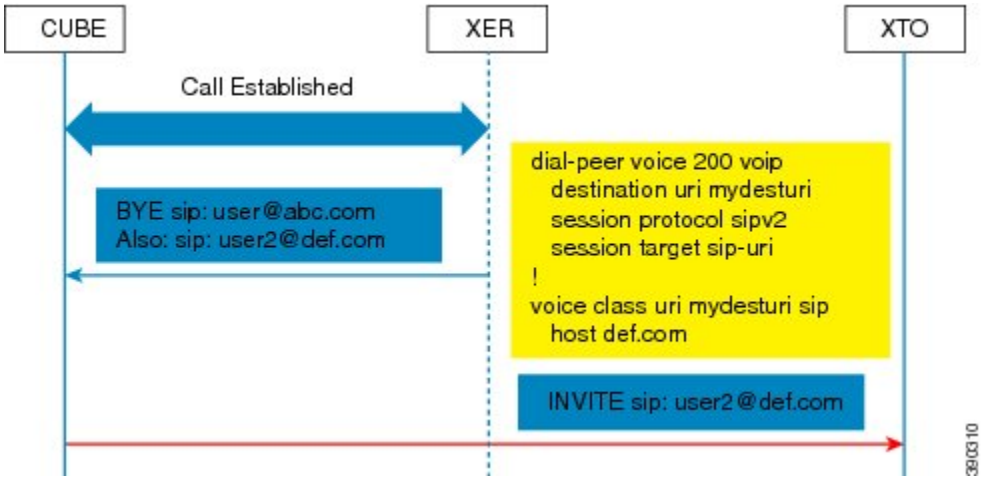
Case 6: In 302 pass-through, contact header can be passed through from one leg to another by using the **contact-passing** command.



Case 7: Pass through of refer-to URI to request URI.



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 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

	Command or Action	Purpose
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	**voice service voip** **Example:** `Device(config)# voice service voip`	Specifies VoIP encapsulation and enters voice service configuration mode.
Step 4	**sip** **Example:** `Device(conf-voi-serv)# sip`	Enters the Session Initiation Protocol (SIP) configuration mode.
Step 5	**requri-passing** **Example:** `Router(conf-serv-sip)# requri-passing`	Enables pass through of the host part of the Request-URI and To SIP headers. By default, Cisco UBE sets the host part of the URI to the value configured under the session target of the outbound dial peer.
Step 6	**end** **Example:** `Router(conf-serv-sip)# end`	Ends the current configuration session and returns to privileged EXEC mode.

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 requri-passing** [**system**]
11. **end**

DETAILED STEPS

	Command or Action	Purpose
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	**voice class uri** *tag* **sip** **Example:** `Device(config)# voice class uri mydesturi sip`	Creates a voice class for matching dial peers to a Session Initiation Protocol (SIP) and enters voice URI class configuration mode.
Step 4	**host** *hostname-pattern* **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.

	Command or Action	Purpose
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 requri-passing** [**system**] **Example:** `Device(config-dial-peer)# voice-class sip requri-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

	Command or Action	Purpose
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	**voice service voip** **Example:** `Device(config)# voice service voip`	Specifies VoIP encapsulation and enters voice service configuration mode.
Step 4	**sip** **Example:** `Device(conf-voi-serv)# sip`	Enters voice service SIP configuration mode.
Step 5	**contact-passing** **Example:** `Router(conf-serv-sip)# contact-passing`	Enables pass through of the contact header from one leg to the other leg in 302 pass through scenario.
Step 6	**end** **Example:** `Router(conf-serv-sip)# end`	Ends the current configuration session and returns to privileged EXEC mode.

Configuring Pass Through of 302 Contact Header (Dial Peer Level)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class uri** *destination-tag* **sip**
4. **user-id** *id-tag*
5. **exit**
6. **voice service voip**
7. **allow-connections sip to sip**
8. **dial-peer voice** *tag* **voip**
9. **session protocol sipv2**
10. **destination uri** *destination-tag*
11. **voice-class sip contact-passing**
12. **end**

DETAILED STEPS

	Command or Action	Purpose
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	**voice class uri** *destination-tag* **sip** **Example:** `Device(config)# voice class uri mydesturi sip`	Creates a voice class for matching dial peers to a Session Initiation Protocol (SIP) and enters voice URI class configuration mode.
Step 4	**user-id** *id-tag* **Example:** `Device(config-voice-uri-class)# user-id 5678`	Matches a call based on the User ID portion of the Uniform Resource Identifier (URI).
Step 5	**exit** **Example:** `Device(config-voice-uri-class)# exit`	Exits voice URI class configuration mode.
Step 6	**voice service voip** **Example:** `Device(config)# voice service voip`	Specifies Voice over IP (VoIP) as the voice encapsulation type and enters voice service configuration mode.

	Command or Action	Purpose
Step 7	**allow-connections sip to sip** **Example:** `Device(conf-voi-serv)# allow-connections sip to sip`	Allows connections between SIP endpoints in a VoIP network.
Step 8	**dial-peer voice** *tag* **voip** **Example:** `Device(config)# dial-peer voice 200 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 a session protocol for calls between local and remote routers using the Internet Engineering Task Force (IETF) SIP.
Step 10	**destination uri** *destination-tag* **Example:** `Device(config-dial-peer)# destination uri mydesturi`	Specifies the voice class used to match a dial peer to the destination URI of an outgoing call.
Step 11	**voice-class sip contact-passing** **Example:** `Device(config-dial-peer)# voice-class sip contact-passing`	Enables pass through of the contact header from one leg to the other leg in 302 pass through scenario.
Step 12	**end** **Example:** `Device(config-dial-peer)# end`	Ends the current configuration session and returns to privileged EXEC mode.

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

	Command or Action	Purpose
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	**voice class uri** *destination-tag* **sip** **Example:** `Device(config)# voice class uri mydesturi sip`	Creates or modifies a voice class for matching dial peers to a Session Initiation Protocol (SIP) or telephone (TEL) Uniform Resource Identifier (URI) and enters voice URI class configuration mode.
Step 4	**host** *hostname-pattern* **Example:** `Device(config-voice-uri-class)# host destination.com`	Matches a call based on the host field in a SIP 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 25 voip`	Defines a VoIP dial peer and enters dial peer configuration mode.

	Command or Action	Purpose
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** *destination-tag* **Example:** `Device(config-dial-peer)# 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 sip-uri** **Example:** `Device(config-dial-peer)# session target sip-uri`	Derives session target from incoming URI.
Step 10	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits dial peer voice configuration mode.
Step 11	**voice class uri** *source-tag* **sip** **Example:** `Device(config)# voice class uri mysourceuri sip`	Creates or modifies a voice class for matching dial peers to a SIP or TEL URI and enters voice URI class configuration mode.
Step 12	**host** *hostname-pattern* **Example:** `Device(config-voice-uri-class)# host abc.com`	Matches a call based on the host field in a SIP URI.
Step 13	**end** **Example:** `Device(config-voice-uri-class)# end`	Ends the current configuration session and returns to privileged EXEC mode.

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(conf-voi-serv)# sip
```

```
Device(conf-serv-sip)# requri-passing
Device(conf-serv-sip)# end
```

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

```
! 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(conf-voi-serv)# sip
Device(conf-serv-sip)# contact-passing
Device(conf-serv-sip)# end
```

Example: Configuring Pass Through of 302 Contact Header (Dial Peer Level)

```
! Configuring URI voice class destination
Device> enable
Device# configure terminal
Device(config)# voice class uri mydesturi sip
Device(config-voice-uri-class)# user-id 5678
Device(config-voice-uri-class)# exit

! Configuring outbound dial peer
Device(config)# voice service voip
Device(conf-voi-serv)# allow-connections sip to sip
Device(conf-voi-serv)# dial-peer voice 200 voip
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# destination uri mydesturi
Device(config-dial-peer)# voice-class sip contact-passing
Device(config-dial-peer)# end
```

Example: Deriving Session Target from URI

```
Device> enable
Device# configure terminal
Device(config)# voice class uri mydesturi sip
Device(config-voice-uri-class)# host destination.com
Device(config-voice-uri-class)# exit
!
Device(config)# dial-peer voice 25 voip
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# destination uri mydesturi
Device(config-dial-peer)# session target sip-uri
Device(config-dial-peer)# exit
!
Device(config)# voice class uri mysourceuri sip
```

```
Device(config-voice-uri-class)# host abc.com
Device(config-voice-uri-class)# end
```

Additional References for URI-Based Dialing Enhancements

Related Documents

Related Topic	Document Title
Voice commands	Cisco IOS Voice Command Reference
Cisco IOS commands	Cisco IOS Master Command List, All Releases
SIP configuration tasks	SIP Configuration Guide, Cisco IOS Release 15M&T

Technical Assistance

Description	Link
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.	http://www.cisco.com/support

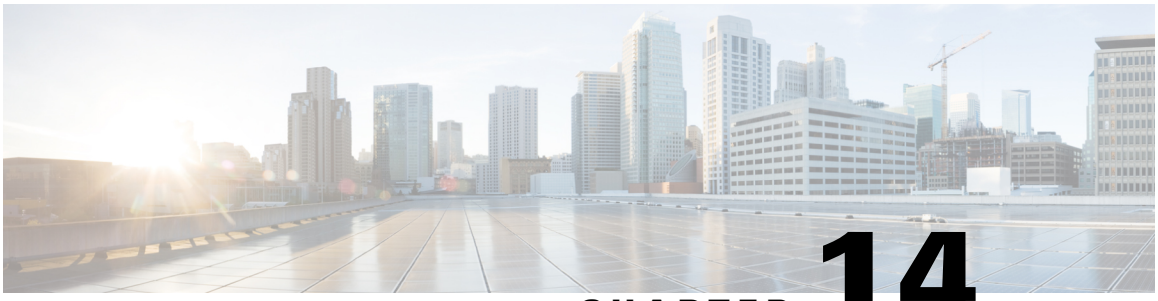
Feature Information for URI-Based Dialing Enhancements

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 24: Feature Information for URI-Based Dialing Enhancements

Feature Name	Releases	Feature Information
URI-Based Dialing Enhancements	15.4(1)T Cisco IOS XE Release 3.11S	The URI-Based Dialing Enhancements feature includes support for call routing on Cisco UBE when the user-part of the incoming Request-URI is non-E164 (for example, INVITE sip:user@abc.com). The following commands were introduced or modified: **contact-passing**, **requri-passing**, **session target sip-uri** and **voice-class sip requri-passing**



CHAPTER 14

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

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

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

Table 25: Feature Information for Multiple Pattern Support on a Voice Dial Peer

Feature Name	Releases	Feature Information
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**
Configuring Multiple Pattern Support on a Voice Dial Peer (Outbound Calls)	Cisco IOS 15.2(4)M Cisco IOS XE 3.7S	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**.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice class e164-pattern-map** *pattern-map-id* **Example:** `Device(config)# voice class e164-pattern-map 1111`	Creates a pattern map for configuring one or multiple E.164 patterns on a dial peer and enters voice class configuration mode.

	Command or Action	Purpose

Step 4	Do one of the following:
	• **e164** *pattern-map-tag*
	• **url** *url*
	Example:
	Using URL text file:
	`Device(voice-class)# url http://http-host/config-files/pattern-map.cfg`
	Directly specifying match patterns:
	`Device(voice-class)# e164 5557123`
		Configure one or more E.164 telephone number prefix match patterns for the pattern map.
		• Repeat this step for each pattern if you are using the **e164** command.
		• You can specify a file URL containing the patterns for this dial peer using the **url** *url* command. You must then load the E.164 telephone prefixes using Step 10. The file can be internal (on the device) or external.

Step 5	**description** *string* **Example:** `Device(voice-class)# description It has 1 entry`	(Optional) Provides a description for the pattern map.
Step 6	**exit** **Example:** `Device(voice-class)# exit`	Exits voice class configuration mode and enters global configuration mode.
Step 7	**dial-peer voice** *dial-peer-id* **voip** **Example:** `Device(config)# dial-peer voice 2222 voip`	Defines a VoIP dial peer and enters dial peer configuration mode.
Step 8	{**destination** | **incoming called** | **incoming calling**} **e164-pattern-map** *pattern-map-group-id* **Example:** `Device(config-dial-peer)# incoming calling e164-pattern-map 1111`	Links a pattern-map group with a dial peer. • Use the **destination** keyword for outbound dial peers. • Use the **incoming called** or **incoming calling** keywords for inbound dial peers using called or calling numbers.
Step 9	**end** **Example:** `Device(config-dial-peer)# end`	Exits dial peer configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 10	**voice class e164-pattern-map load** *pattern-map-group-id* **Example:** `Device# voice class e164-pattern-map load 1111`	(Optional) Loads the specified pattern map with E.164 match patterns from a text file configured in the pattern map. • This step is required only if patterns have been defined for the specified pattern map using a file URL in Step 4.
Step 11	**show dial-peer voice** [**summary** | *dial-peer-id*] **Example:** `Device# show dial-peer voice 1111`	Displays the status of a pattern map when the pattern map is associated with a dial peer.

Verifying Multiple Pattern Support on a Voice Dial Peer

SUMMARY STEPS

1. **show voice class e164-pattern-map** [**summary** | *pattern-map-id*]
2. **show dial-peer voice** [**summary** | *dial-peer-id*]
3. **show dialplan incall {sip | h323} {calling | called}** *e164-pattern*

DETAILED STEPS

Step 1 **show voice class e164-pattern-map** [**summary** | *pattern-map-id*]
Displays the status and contents of a specified pattern map or a status summary of all pattern maps.

Example:

```
Device# show voice class e164-pattern-map 200

e164-pattern-map 200
----------------------------------------
  It has 1 entries
  It is not populated from a file.
  Map is valid.

E164 pattern
-------------------
200
```

Step 2 **show dial-peer voice** [**summary** | *dial-peer-id*]
Displays the status of pattern maps associated with all or a specified dial peer.

Example:

```
Device# show dial-peer voice | include  e164-pattern-map
```

```
        incoming calling e164-pattern-map tag = `200' status = valid,
        destination e164-pattern-map tag = 3000 status = valid,

Device# show dial-peer voice 2222| include  e164-pattern-map

        incoming calling e164-pattern-map tag = `200' status = valid,
```

Step 3 **show dialplan incall {sip | h323} {calling | called}** *e164-pattern*

Displays inbound dial peer details and associated pattern maps based on an incoming calling or called number.

Example:

```
Device#  show dialplan incall voip calling 23456

VoiceOverIpPeer1234567
        peer type = voice, system default peer = FALSE, information type = voice,
        description = `',
        tag = 1234567, destination-pattern = `',
        destination e164-pattern-map tag = 200 status = valid,
        destination dpg tag = 200 status = valid,
        voice reg type = 0, corresponding tag = 0,
        allow watch = FALSE
        answer-address = `', preference=0,
        incoming calling e164-pattern-map tag = `200' status = valid,
        CLID Restriction = None
```

Configuration Examples for Multiple Pattern Support on a Voice Dial Peer

Example: Configuring Multiple Patterns for Outbound Dial Peers Using a File URL

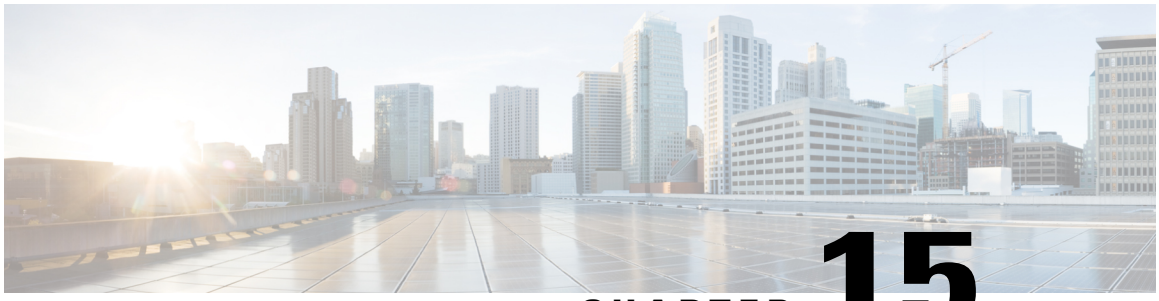
```
Device# voice class e164-pattern-map 1111
Device(voice-class)# url http://http-host/config-files/pattern-map.cfg
Device(voice-class)# description For Outbound Dial Peer
Device(voice-class)# exit
Device(config)# dial-peer voice 2222 voip
Device(voice-dial-peer)# destination e164-pattern-map 1111
Device(voice-dial-peer)# exit
Device(config)# voice class e164-pattern-map load 1111
Device(config)# end
```

Example: Configuring Multiple Patterns for Outbound Dial Peers 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 15

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

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

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

Table 26: Feature Information for Outbound Dial-Peer Group as an Inbound Dial-Peer Destination

Feature Name	Releases	Feature Information
Support for POTS dial-peer	Cisco IOS 15.5(1)T Cisco IOS XE 3.14S	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.

Feature Name	Releases	Feature Information
Outbound Dial-Peer Group as an Inbound Dial-Peer Destination	Cisco IOS 15.4(1)T Cisco IOS XE 3.11S	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: **voice class dpg**, **description**, **dial-peer preference**, **destination dpg**, **show voice class dpg**.

Restrictions

- If a dial-peer group is in the shutdown state, regular dial-peer search occurs.
- If all dial peers in an active dial-peer group are unavailable, call is disconnected.
- The number of matched digits is zero.
- The **destination-pattern** command is required on the outbound dial peer even matching is not done based on this command.
- The outgoing call setup is deferred until inter-digit timer expires or a terminator is entered.

For POTS dial peers:

- Two-stage dialing is not supported.
- Overlapping dialing is not supported.
- TCL and VXML routing changes are not supported.
- Digit-stripping is not supported.

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

You can group up to 20 outbound (H.323, SIP or POTS) dial peers into a dial-peer group and configure this dial-peer group as the destination of an inbound dial peer. Once an incoming call is matched by an inbound dial peer with an active destination dial-peer group, dial peers from this group are used to route the incoming call. No other outbound dial-peer provisioning to select outbound dial peers is used.

A preference can be defined for each dial peer in a dial-peer group. This preference is used to decide the order of selection of dial peers from the group for the setup of an outgoing call.

You can also specify various dial-peer hunt mechanism using the existing **dial-peer hunt** command.

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

Perform this task to configure a dial-peer group with multiple outbound peers and an inbound dial peer referencing this dial-peer group as a destination.

Before You Begin

- Configure SIP, H.323 or POTS outbound dial peers to be associated with a dial-peer group.
- For an outbound POTS dial peer, ensure that **destination-pattern .T** and **no digit-strip** are configured to avoid unexpected dialed digit strip.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *outbound-dial-peer-id* [**voip** | **pots**]
4. **destination-pattern** *pattern*
5. **no digit-strip** for POTS dial peers.
6. **exit**
7. (Optional) **dial-peer hunt** *hunt-order-number*
8. **voice class dpg** *dial-peer-group-id*
9. **dial-peer** *outbound-dial-peer-id* [**preference** *preference-order*]
10. (Optional) **description** *string*
11. **exit**
12. **dial-peer voice** *inbound-dial-peer-id* [**voip** | **pots**]
13. **destination dpg** *dial-peer-group-id*
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose

Step 3	**dial-peer voice** *outbound-dial-peer-id* [**voip** | **pots**]
	
	Example:
	For VoIP dial peer:
	`Device(config)# dial-peer voice 123 voip`
	
	Example:
	For POTS dial peer:
	`Device(config)# dial-peer voice 345 pots`
		Defines a dial peer and enters dial peer configuration mode.

Step 4	**destination-pattern** *pattern*
	
	Example:
	For VoIP Dial Peers
	`Device(config-dial-peer)# destination-pattern 1004`
	
	Example:
	For POTS Dial Peers
	`Device(config-dial-peer)# destination-pattern .T`
		Configures a destination pattern. This step is required even though the value is not used for dial-peer matching.

Step 5	**no digit-strip** for POTS dial peers. **Example:** `Device(config-dial-peer)# no digit-strip`	Disable unexpected dialed digit strip.
Step 6	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits to global configuration mode.
Step 7	**dial-peer hunt** *hunt-order-number* **Example:** `Device(config)# dial-peer hunt 0`	(Optional) Specifies a hunt selection mechanism for dial peers. • The default mechanism is random selection.
Step 8	**voice class dpg** *dial-peer-group-id* **Example:** `Device(config)# voice class dpg 181`	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.
Step 9	**dial-peer** *outbound-dial-peer-id* [**preference** *preference-order*]	Associates a configured outbound dial peer with this dial-peer group and configures a preference value.

	Command or Action	Purpose
	Example: `Device(config-class)# dial-peer 123 preference 1`	• Repeat this step for all outbound dial-peers that need to be added to this dial-peer group. • If preference is not specified, the order of selection is random or as specified by the **dial-peer hunt** command.
Step 10	**description** *string* **Example:** `Device(config-class)# description Boston Destination`	(Optional) 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

 Peer Tag        Pref
 --------        ----
 1001            1
 1002            2
 1004            0
 1003            1
-------------------------------------
```

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:

```
*Jul 19 10:15:53.310 IST: //-1/ED647BD1B0F9/DPM/dpMatchCore:
   Dial String=4001, Expanded String=4001, Calling Number=
   Timeout=TRUE, Is Incoming=TRUE, Peer Info Type=DIALPEER_INFO_SPEECH
*Jul 19 10:15:53.310 IST: //-1/xxxxxxxxxxxx/DPM/vepm_match_pattern_map:
   DEPM 1000 use caching dialstring 4001 status 0
*Jul 19 10:15:53.310 IST: //-1/ED647BD1B0F9/DPM/MatchNextPeer:
```

Incoming dial peer is first matched:

```
Result=Success(0); Incoming Dial-peer=600 Is Matched
*Jul 19 10:15:53.310 IST: //-1/ED647BD1B0F9/DPM/dpMatchPeertype:exit@6602
*Jul 19 10:15:53.310 IST: //-1/ED647BD1B0F9/DPM/dpAssociateIncomingPeerCore:
   Result=Success(0) after DP_MATCH_INCOMING_DNIS; Incoming Dial-peer=600
*Jul 19 10:15:53.310 IST: //-1/ED647BD1B0F9/DPM/dpMatchSafModulePlugin:
   dialstring=NULL, saf_enabled=0, saf_dndb_lookup=0, dp_result=0
*Jul 19 10:15:53.310 IST: //-1/ED647BD1B0F9/DPM/dpAssociateIncomingPeerSPI:exit@7181
*Jul 19 10:15:53.311 IST: //-1/ED647BD1B0F9/DPM/dpMatchPeersCore:
   Calling Number=, Called Number=4001, Peer Info Type=DIALPEER_INFO_SPEECH
```

The dial-peer group associated with a dial peer is selected:

```
*Jul 19 10:15:53.311 IST: //-1/ED647BD1B0F9/DPM/dpMatchPeersCore:
   Outbound Destination DPG Group Request; Destination DPG=1
*Jul 19 10:15:53.311 IST: //-1/ED647BD1B0F9/DPM/dpMatchDestDPGroup:
   Result=0
*Jul 19 10:15:53.311 IST: //-1/ED647BD1B0F9/DPM/dpMatchPeersCore:
   Result=SUCCESS(0) after DestDPGroup
*Jul 19 10:15:53.311 IST: //-1/ED647BD1B0F9/DPM/dpMatchSafModulePlugin:
   dialstring=4001, saf_enabled=0, saf_dndb_lookup=1, dp_result=0
```

List of active Dial-peers configured within the DPG, sorted by preference:

```
*Jul 19 10:15:53.311 IST: //-1/ED647BD1B0F9/DPM/dpMatchPeersMoreArg:
   Result=SUCCESS(0)
 List of Matched Outgoing Dial-peer(s):
     1: Dial-peer Tag=1004
     2: Dial-peer Tag=1001
     3: Dial-peer Tag=1003
     4: Dial-peer Tag=1002
```

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
```

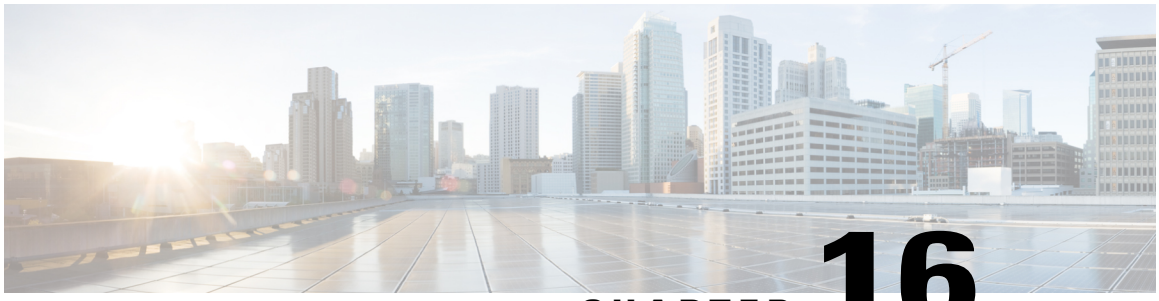
Verifying Outbound Dial-Peer Group Configuration

```
Device# show voice class dpg 200

Voice class dpg: 200     AdminStatus: Up
 Description: Boston Destination
 Total dial-peer entries: 4

 Peer Tag       Pref
```

```
 --------        ----
 1001            1
 1002            2
 1004            0
 1003            1
-------------------------------------
```

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,
```



CHAPTER 16

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. An account on Cisco.com is not required.

Table 27: Feature Information for Inbound Leg Headers for Outbound Dial-Peer Matching

Feature Name	Releases	Feature Information
Inbound Leg Headers for Outbound Dial-Peer Matching	15.4(2)T, Cisco IOS XE Release 3.12S	TheInbound 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**.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice class dial-peer provision-policy** *tag* **Example:** `Device(config)# voice class dial-peer provision-policy 200`	Creates a provision policy profile in which a set of attributes for dial-peer matching can be defined. • You can use the **shutdown** command to deactivate the provision policy and allow normal outbound dial-peer provisioning.
Step 4	**description** *string* **Example:** `Device(voice-class)# description match both calling and called`	(Optional) Provides a description for the provision policy profile.
Step 5	**preference** *preference-order first-attribute second-attribute*	Configures a provision policy rule.

	Command or Action	Purpose
	<table><tr><th>First Attribute</th><th>Second Attribute</th></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, uri, from</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></table> **Example:** `Device(voice-class)# preference 2 calling called`	• You can configure up to two rules. This means up to four attributes can be configured for matching outbound dial peers. • If rules are not configured, outbound dial-peer provisioning is disabled, and an incoming call matched to an inbound dial peer associated with this profile is disconnected by CUBE or voice gateway with cause code "unassigned number (1)".
Step 6	**exit** **Example:** `Device(voice-class)# exit`	Exits voice class configuration mode and enters global configuration mode.
Step 7	**dial-peer voice** *inbound-dial-peer-tag* **voip**	Enters dial peer configuration mode for an inbound dial peer.

Step 8	**destination provision-policy** *tag*
	Example:
	`Device(config)# dial-peer voice 100 voip`
	`Device(config-dial-peer)# destination provision-policy 200`
	`Device(config)# exit`
		Associates a provision policy profile with an inbound dial peer.

Step 9	**exit**	Exits dial peer configuration mode.
Step 10	**dial-peer voice** *outbound-dial-peer-tag* **voip**	Enters dial peer configuration mode for an outbound dial peer.

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td>Step 11</td><td>Configure a match command for an outbound dial peer according to the provision policy rule attribute configured.
<table><tr><th>Provision Policy Rule Attribute</th><th>Dial-peer Match command</th></tr><tr><td>called</td><td>destination-pattern pattern
destination e164-pattern-map pattern-map-class-id</td></tr><tr><td>calling</td><td>destination calling e164-pattern-map pattern-map-class-id</td></tr><tr><td>carrier-id</td><td>carrier-id target</td></tr><tr><td>uri</td><td>destination uri uri-class-tag</td></tr><tr><td>via</td><td>destination uri-via uri-class-tag</td></tr><tr><td>to</td><td>destination uri-to uri-class-tag</td></tr><tr><td>from</td><td>destination uri-from uri-class-tag</td></tr><tr><td>diversion</td><td>destination uri-diversion uri-class-tag</td></tr><tr><td>referred-by</td><td>destination uri-referred-by uri-class-tag</td></tr></table>Example:
<pre>Device(config)# dial-peer voice 300 voip
Device(config-dial-peer)# destination uri-from 200
Device(config)# exit</pre></td><td>Configure a match command based on any of the four attributes defined in the provision policy rule.</td></tr>
<tr><td>Step 12</td><td>end
Example:
<pre>Device(config-dial-peer)# end</pre></td><td>Exits dial peer configuration mode and enters privileged EXEC mode.</td></tr>
</table>

Verifying Inbound Leg Headers for Outbound Dial-Peer Matching

SUMMARY STEPS

1. **show dialplan incall {sip | h323} {calling | called}** *e164-pattern* | **include voice**
2. **show dialplan dialpeer** *inbound-dial-peer-id* **number** *e164-pattern* **[timeout] | include Voice**
3. **show voice class dial-peer provision-policy**

DETAILED STEPS

Step 1 **show dialplan incall {sip | h323} {calling | called}** *e164-pattern* | **include voice**

Displays inbound dial peers based on an incoming calling or called number. Once you have the dial peer number, you can use it to search for the complete dial-peer details in the running-config.

Example:

```
Device# show dialplan incall sip calling 3333 | include Voice

VoiceOverIpPeer1

Device# show dialplan incall sip calling 4444 | include Voice

VoiceOverIpPeer1

Device# show running-config | section dial-peer voice 1 voip

dial-peer voice 1 voip
 destination dpg 10000
 incoming calling e164-pattern-map 100
 dtmf-relay rtp-nte
 codec g711ulaw

Device# show dialplan incall sip called 6000 timeout | include Voice

VoiceOverIpPeer100

Device# show running-config | section dial-peer voice 100 voip

dial-peer voice 100 voip
 incoming called e164-pattern-map 1
 incoming calling e164-pattern-map 1
 dtmf-relay rtp-nte
 codec g711ulaw

Device#  show dialplan incall voip calling 23456

VoiceOverIpPeer1234567
        peer type = voice, system default peer = FALSE, information type = voice,
        description = `',
        tag = 1234567, destination-pattern = `',
        destination e164-pattern-map tag = 200 status = valid,
        destination 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 **show dialplan dialpeer** *inbound-dial-peer-id* **number** *e164-pattern* **[timeout] | include Voice**

Displays a list of outbound dial peers based on a specified inbound dial peer. This command line will be helpful find a list of outbound dial peer of a destination dial-peer group.

Example:

```
Device# show dialplan dialpeer 1 number 23457 timeout | include Voice

VoiceOverIpPeer100013
VoiceOverIpPeer100012
```

Example:

```
voice class dial-peer provision-policy 2000
 preference 2 diversion to
!
...
!
dial-peer voice 32555 voip
 session protocol sipv2
 session target ipv4:1.5.14.9
 destination uri-diversion 1
 destination uri-to test2
!
dial-peer voice 32991 voip
 destination provision-policy 2000
 incoming called-number 1234
!

Device# show dialplan dialpeer 32991 number 2234 timeout

Macro Exp.: 2234
Enter Diversion header:sip:1234@cisco.com
Enter To header:sip:2234@10.0.0.0
VoiceOverIpPeer32134
        peer type = voice, system default peer = FALSE, information type = voice,
        description = `',
```

Step 3 **show voice class dial-peer provision-policy**

Displays a list of configured provision policies and associated rules.

Example:

```
Device# show voice class dial-peer provision-policy

Voice class dial-peer provision-policy: 100  AdminStatus: Up
 Description: match only called

 Pref  Policy Rule
 ----  -----------
 1     called

Voice class dial-peer provision-policy: 101  AdminStatus: Up
 Description: match both calling and called

 Pref  Policy Rule
 ----  -----------
 1     called calling

Voice class dial-peer provision-policy: 102  AdminStatus: Up
 Description: match calling first; if no match then match called

 Pref  Policy Rule
 ----  -----------
 1 calling
 2 called

Voice class dial-peer provision-policy: 200  AdminStatus: Up
 Description: match referred-by and via uri; if no match then match request- uri
```

```
 Pref  Policy Rule
 ----  -----------
 1 referred-by via
 2 uri

voice class dial-peer provision-policy: 300  AdminStatus: Up
 Description: match only request-uri

 Pref  Policy Rule
 ----  -----------
 1 uri

Voice class dial-peer provision-policy: 400  AdminStatus: Up
 Description: match only request uri; if no match then match called

 Pref  Policy Rule
 ----  -----------
 1 uri
 2 called
```

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

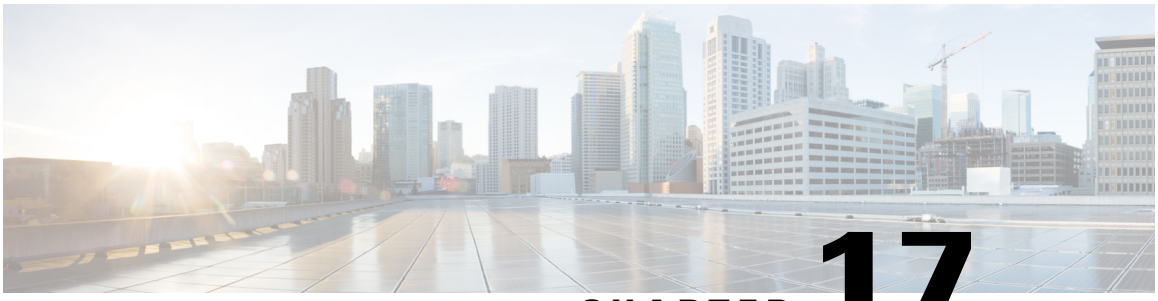
Example: Configuring Inbound Called or Calling Numbers Used for Outbound Dial-Peer Matching

```
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(voice-class)# description patterns
Device(voice-class)# e164 5557123
Device(voice-class)# e164 5558123
Device(voice-class)# e164 5559123
Device(voice-class)# exit

!Associating the Provision Policy with an Inbound Dial Peer
Device(config)# dial-peer voice 100 voip
Device(config-dial-peer)# destination provision-policy 200
Device(config-dial-peer)# end

!Associates a Pattern Map with an Outbound Dial Peer.
! The called number in the SIP headers of the inbound leg is matched to select the below
outbound dial peer.
Device(config)# dial-peer voice 200 voip
Device(config-dial-peer)# destination e164-pattern-map 300
Device(config-dial-peer)# end
```

Example: Configuring Inbound SIP Headers for Outbound Dial-Peer Matching

```
Device> enable
Device# configure terminal

Device(config)# voice class dial-peer provision-policy 200
Device(voice-class)# description match both calling and called
Device(voice-class)# preference 2 via from
Device(voice-class)# exit

!Associating the Provision Policy with an Inbound Dial Peer
```

```
Device(config)# dial-peer voice 100 voip
Device(config-dial-peer)# destination provision-policy 200
Device(config-dial-peer)# end

!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
```



CHAPTER 17

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

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 Configuring Server Groups in Outbound Dial Peers

Feature Name	Releases	Feature Information
Configuring Server Groups in Outbound Dial Peers	Cisco IOS XE Release 3.11S 15.4(1)T	This feature configures server groups (groups of IPv4 and IPv6 addresses) which can be referenced from an outbound SIP dial peer. The following commands were introduced or modified: **voice class server-group**, **description**, **ipv4 port preference**, **ipv6 port preference**, **hunt-scheme**, **show voice class server-group**, **shutdown (Server Group)**.

Information About Server Groups in Outbound Dial Peers

You can now group IPv4 and IPv6 addresses of servers and configure it as in an outbound SIP dial-peer destination. A server group is first created and associated with a SIP outbound dial peer. When a call matches an outbound dial peer, a connection is made to one of the servers of the group, which can then route the call to the destination.

You can associate a preference with each of the servers in the group, which decides the order of selection for outgoing call setup.

You can also configure the round-robin selection of servers in the group.

You can configure up to five dial peers to a server group.

If neither round robin nor preference order is configured, the selection of servers is random.

If the specified server-group is in shutdown mode, the referenced dial peers are not selected to route outgoing calls.

How to Configure Server Groups in Outbound Dial Peers

Configuring Server Groups in Outbound Dial Peers

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class server-group** *server-group-id*
4. {**ipv4** | **ipv6**} *address* [**port** *port*] [**preference** *preference-order*]
5. (Optional) **hunt-scheme round-robin**
6. (Optional) **description** *string*
7. **exit**
8. **dial-peer voice** *dial-peer-id* **voip**
9. **session protocol sipv2**
10. **destination-pattern [+]** *string* **[T]**
11. **session server-group** *server-group-id*
12. **end**
13. **show voice class server-group** *server-group-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enters privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice class server-group** *server-group-id* **Example:** `Device(config)# voice class server-group 171`	Configures a voice class server group and enters voice class configuration mode. • You can use the **shutdown** command to make the server group inactive.
Step 4	{**ipv4** | **ipv6**} *address* [**port** *port*] [**preference** *preference-order*] **Example:** `Device(config-class)# ipv4 10.1.1.1 preference 3`	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.
Step 5	**hunt-scheme round-robin** **Example:** `Device(config-class)# hunt-scheme round-robin`	(Optional) 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 values is configured, the selection of servers is random.
Step 6	**description** *string* **Example:** `Device(config-class)# description It has 3 entries`	(Optional) Provides a description for the server group.
Step 7	**exit** **Example:** `Device(config-class)# exit`	Exits voice class mode and enters global configuration mode.

	Command or Action	Purpose
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:** `Device# show voice class server-group 171`	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 voice class server-group dialpeer** *dial-peer-id*

DETAILED STEPS

Step 1 **show voice class server-group** [*server-group-id*]
Displays the configurations for all configured server groups or a specified server group.

Example:

```
Device# show voice class server-group 171

Voice class server-group: 171
 AdminStatus: Up                OperStatus: Up
 Hunt-Scheme: preference        Last returned server: 10.1.1.1
 Description:
 Total server entries: 3

 Pref   Type   IP Address                              IP Port
 ----   ----   ----------                              -------
 1      ipv4   10.1.1.1
 2      ipv4   10.1.1.2                                34515
 3      ipv4   2001:DB8:12:1::26
-------------------------------------
```

Step 2 **show voice class server-group dialpeer** *dial-peer-id*
Displays the configurations for dial peers associated with server groups.

Example:

```
Device# show voice class server-group dialpeer 181

Voice class server-group: 171    AdminStatus: Up
 Hunt-Scheme: round-robin
 Total Remote Targets: 3

 Pref   Type   IP Address                              IP Port
 ----   ----   ----------                              -------
 0      ipv4   10.1.1.2
 1      ipv4   10.5.0.1
 1      ipv4   10.1.1.1
```

Configuration Examples for Server Groups in Outbound Dial Peers

Server Groups in Outbound Dial Peers (Preference-Based Selection)

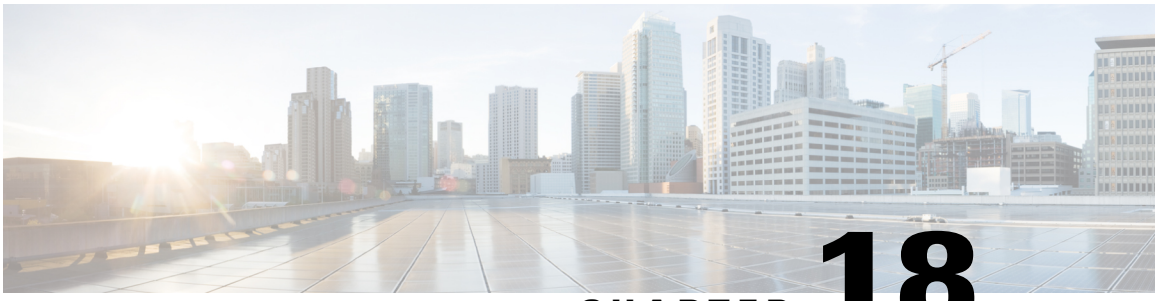
```
! Configuring the Server Group
Device(config)# voice class server-group 171
Device(config-class)# ipv4 10.1.1.1 preference 1
Device(config-class)# ipv4 10.1.1.2 preference 2
Device(config-class)# ipv4 10.1.1.3 preference 3
Device(config-class)# description It has 3 entries
Device(config-class)# exit


! Configuring an outbound SIP dial peer.
Device(config)# dial-peer voice 181 voip system
!Associate a destination pattern
Device(config-dial-peer)# destination-pattern 3001
Device(config-dial-peer)# session protocol sipv2
!Associate a server group with the dial peer
```

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

! Displays the configurations made for the outbound dial peer 181 associated with a server
 group
Device# show voice class server-group dialpeer 181

Voice class server-group: 171    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 171

Voice class server-group: 171
 AdminStatus: Up                 OperStatus: Up
 Hunt-Scheme: preference         Last returned server: 10.1.1.1
 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

-------------------------------------
```

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

```
! Configuring the Server Group
Device(config)# voice class server-group 171
Device(config-class)# ipv4 10.1.1.1
Device(config-class)# ipv4 10.1.1.2
Device(config-class)# ipv4 10.1.1.3
Device(config-class)# hunt-scheme round-robin
Device(config-class)# description It has 3 entries
Device(config-class)# exit

! Configuring an outbound SIP dial peer.
Device(config)# dial-peer voice 181 voip system
! Associate a destination pattern
Device(config-dial-peer)# destination-pattern 3001
Device(config-dial-peer)# session protocol sipv2
! Associate a server group with the dial peer
Device(config-dial-peer)# session server-group 171
Device(config-dial-peer)# end

! Displays the configurations made for the outbound dial peer 181 associated with a server
 group
Device# show voice class server-group dialpeer 181

Voice class server-group: 171    AdminStatus: Up
 Hunt-Scheme: round-robin
 Total Remote Targets: 3

 Pref   Type   IP Address                              IP Port
 ----   ----   ----------                              -------
 0      ipv4   10.1.1.1
 0      ipv4   10.1.1.2
 0      ipv4   10.1.1.3

! Displays the configurations made for the server group.
```

```
Device# show voice class server-group 171

Voice class server-group: 171
 AdminStatus: Up                 OperStatus: Up
 Hunt-Scheme: round-robin        Last returned server: 10.1.1.1
 Description: It has 3 entries
 Total server entries: 3

 Pref   Type   IP Address                                IP Port
 ----   ----   ----------                                -------
 0      ipv4   10.1.1.1
 0      ipv4   10.1.1.2
 0      ipv4   10.1.1.3

-------------------------------------
```



CHAPTER 18

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

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

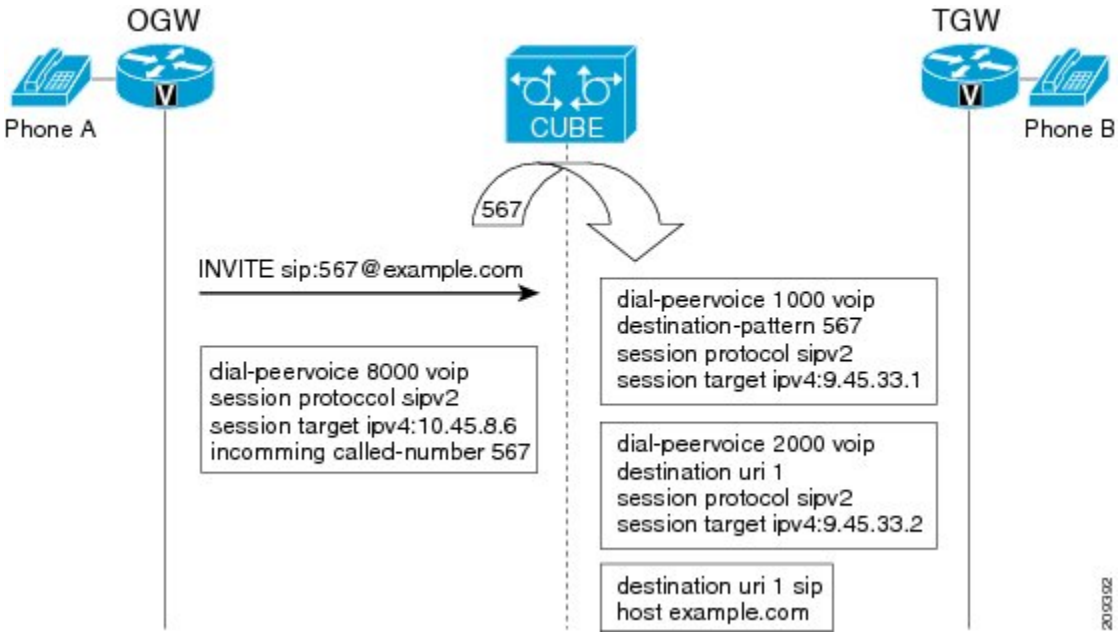
Feature Name	Releases	Feature Information
Domain Based Routing Support on the Cisco UBE	15.2(1)T	The domain-based routing enables for calls to be routed on the outbound dial peer based on the domain name or IP address provided in the request URI (Uniform Resource Identifier) of incoming SIP message. 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**.

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.

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

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters voice service configuration mode.
Step 4	**sip** **Example:** `Device(conf-voi-serv)# sip`	Enters voice service SIP configuration mode.
Step 5	**call-route url** **Example:** `Device(conf-serv-sip)# call-route url` **Example:**	Routes calls based on the URL.
Step 6	**exit** **Example:** `Device(conf-serv-sip)# exit`	Exits the current mode.

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

	Command or Action	Purpose
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** *dial-peer tag* **voip** **Example:** `Device(config)#` **dial-peer voice** *2* **voip**	Enter dial peer voice configuration mode.
Step 4	**voice-class sip call-route url** **Example:** `Device(config-dial-peer)#` **Example:** Routes calls based on the URL	
Step 5	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits the current mode.

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@[2208:1:1:1:1:1:1:1118]:5060 SIP/2.0
Via: SIP/2.0/UDP [2208:1:1:1:1:1:1:1115]:5060;branch=z9hG4bK83AE3
Remote-Party-ID: <sip:2222222222@[2208:1:1:1:1:1:1:1115]>;party=calling;screen=no;privacy=off
From: <sip:2222222222@[2208:1:1:1:1:1:1:1115]>;tag=627460F0-1259
To: <sip:5555555555@[2208:1:1:1:1:1:1:1118]>
Date: Tue, 01 Mar 2011 08:49:48 GMT
Call-ID: B30FCDEB-431711E0-8EDECB51-E9F6B1F1@2208:1:1:1:1:1:1:1115
Supported: 100rel,timer,resource-priority,replaces
Require: sdp-anat
Min-SE:  1800
Cisco-Guid: 2948477781-1125585376-2396638033-3925258737
User-Agent: Cisco-SIPGateway/IOS-15.1(3.14.2)PIA16
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1298969388
Contact: <sip:2222222222@[2208:1:1:1:1:1:1:1115]:5060>
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 495
v=0
o=CiscoSystemsSIP-GW-UserAgent 7880 7375 IN IP6 2208:1:1:1:1:1:1:1115
s=SIP Call
c=IN IP6 2208:1:1:1:1:1:1:1115
t=0 0
a=group:ANAT 1 2
m=audio 17836 RTP/AVP 0 101 19
c=IN IP6 2208:1:1:1:1:1:1:1115
a=mid:1
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=rtpmap:19 CN/8000
a=ptime:20
m=audio 18938 RTP/AVP 0 101 19
c=IN IP4 9.45.36.111
a=mid:2
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=rtpmap:19 CN/8000
a=ptime:20
“Received:
INVITE sip:2222222222@[2208:1:1:1:1:1:1:1117]:5060 SIP/2.0
Via: SIP/2.0/UDP [2208:1:1:1:1:1:1:1116]:5060;branch=z9hG4bK38ACE
Remote-Party-ID: <sip:5555555555@[2208:1:1:1:1:1:1:1116]>;party=calling;screen=no;privacy=off
From: <sip:5555555555@[2208:1:1:1:1:1:1:1116]>;tag=4FE8C9C-1630
To: <sip:2222222222@[2208:1:1:1:1:1:1:1117]>;tag=1001045C-992
Date: Thu, 10 Feb 2011 12:15:08 GMT
Call-ID: 5DEDB77E-ADC11208-808BE770-8FCACF34@2208:1:1:1:1:1:1:1117
Supported: 100rel,timer,resource-priority,replaces,sdp-anat
Min-SE:  1800
Cisco-Guid: 1432849350-0876876256-2424621905-3925258737
User-Agent: Cisco-SIPGateway/IOS-15.1(3.14.2)PIA16
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
```

```
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1297340108
Contact: <sip:5555555555@[2208:1:1:1:1:1:1:1116]:5060>
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Length: 424
v=0
o=CiscoSystemsSIP-GW-UserAgent 8002 7261 IN IP6 2208:1:1:1:1:1:1:1116
s=SIP Call
c=IN IP6 2208:1:1:1:1:1:1:1116
t=0 0
m=image 17278 udptl t38
c=IN IP6 2208:1:1:1:1:1:1:1116
a=T38FaxVersion:0
a=T38MaxBitRate:14400
a=T38FaxFillBitRemoval:0
a=T38FaxTranscodingMMR:0
a=T38FaxTranscodingJBIG:0
a=T38FaxRateManagement:transferredTCF
a=T38FaxMaxBuffer:200
a=T38FaxMaxDatagram:320
a=T38FaxUdpEC:t38UDPRedundancy"
```

Step 3 **debug voip dialpeer inout**

The **debug 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
*May  1 19:32:11.731: //-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:
   Result=Success(0) after DP_MATCH_INCOMING_DNIS; Incoming Dial-peer=100
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersCore:
   Calling Number=, Called Number=3600, Peer Info Type=DIALPEER_INFO_SPEECH
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersCore:
   Match Rule=DP_MATCH_DEST; Called Number=3600
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersCore:
   Result=Success(0) after DP_MATCH_DEST
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersMoreArg:
   Result=SUCCESS(0)
```

The following event shows the matched dial peers in the order of priority:

Example:

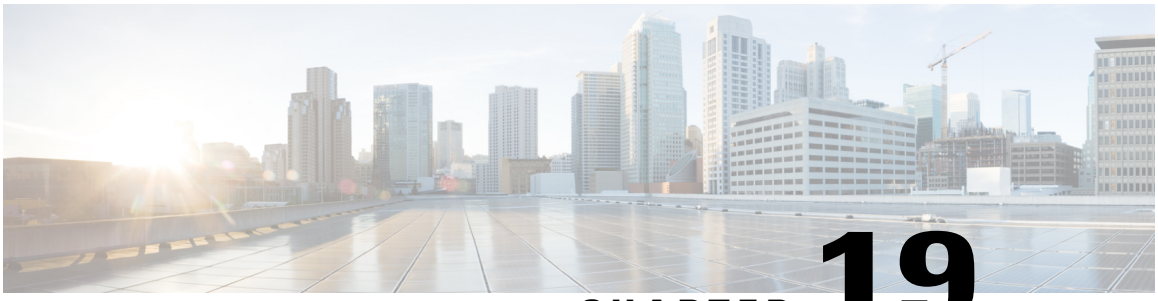
```
List of Matched Outgoing Dial-peer(s):
     1: Dial-peer Tag=3600
     2: Dial-peer Tag=36
```

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

Example Configuring Domain-Based Routing Support on the Cisco UBE

The following example shows how to enable domain-based routing support on the Cisco UBE:

```
Device> enable
Device# configure terminal
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# call-route url
Device(conf-serv-sip)# exit
Device(config)# dial-peer voice 2 voip
Device(config-dial-peer)# voice-class sip call-route url
Device(config-dial-peer)# exit
```



CHAPTER 19

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 URIs. 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 pseduo 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

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 30: Feature Information for ENUM Enhancement per Kaplan Draft RFC

Feature Name	Releases	Feature Information
ENUM Enhancement per Kaplan Draft RFC	Cisco IOS XE 3.14S Cisco IOS 15.5(1)T	The ENUM enhancement per Kaplan draft RFC provides source-based routing, that is, SIP-to-SIP calls can be routed based on the source SIP requests. To provide this source-based routing, an EDNS0 OPT pseudo resource record with source URI is added to the ENUM DNS query, according to **draft-kaplan-enum-sip-routing-04**. This feature also provides the flexibility to disable route caching.
Support to include inbound call ID, outbound call ID and Call Session Identification to ENUM DNS query	Cisco IOS 15.5(2)T Cisco IOS XE 3.15S	This feature allows you to add incoming SIP call ID, outbound SIP call ID, and Call Session Identification to an EDNS0 OPT pseduo resource record in the ENUM DNS query.

Restrictions for ENUM Enhancement per Kaplan Draft RFC

- Supported only for SIP-to-SIP calls.
- The full command of **voice enum-match-table**, including the options, needs to be specified whenever being referenced by its subcommand. If not, the defaults, **no source-uri** and no no-cached (or caching) will take effect.
- As the maximum number of characters of the host shown in the **show host** command is 25, the source URI may not be displayed completely.
- The source URI is displayed in a separate line below, starting with “source-uri=”. Refer to the **show** command outputs in this chapter.
- If **no-cache** is configured in the **voice enum-match-table**, no cache table look-up would be made and hence an ENUM query would be made regardless of what is in the cache table.

- Both the target and source, where the source can be null/undefined or defined, need to be matched when looking up the cache table.
- The OPT RR will be added to the query for a SIP-to-SIP call only if the **source-uri** is configured for the outbound **enum-match-table**.
- The route will not be cached if the server does not support the OPT RR (it is recommended to remove the **source-uri** for this scenario if caching is preferred).
- The source URL can be prefixed with a host/target in the host name field in a double quote in the **show host** *host* command to display routes for the host specific with this source.
- A wild card, "*", can be used to denote "all" hosts in the **show host** command. It can be by itself or any host matched with its prefix. The prefix can be a host name, partial or complete, or a domain name with partial or complete source URL.

Refer to the document titled *Unified Border Element ENUM Support Configuration Example* for a detailed message format.

Information About ENUM Enhancement per Kaplan Draft RFC

SIP-to-SIP calls can be routed based on the source SIP requests, using the ENUM enhancement feature. To provide source-based routing and to interact with Policy Server, an EDNS0 OPT pseudo resource record with source URI, incoming SIP call ID, outbound SIP call ID, and Call session Identification are added to the ENUM DNS query. The DNS server filters its response based on the source URI and call ID information and returns the appropriate NAPTR entries. To enable this feature, you must use the **source-uri** option in the **voice enum-match-table** <*table-number*> command. In addition, you can use the **no-cache** option to disable caching.

Refer to RFC 3761 and **draft-kaplan-enum-sip-routing-04** for more information about routing SIP requests with ENUM.

How to Configure ENUM Enhancement per Kaplan Draft RFC

Enabling Source-Based Routing

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice enum-match-table** *match-table-index* [**source-uri**] [**no-cache**]
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice enum-match-table** *match-table-index* [**source-uri**] [**no-cache**] **Example:** `Device(config)# voice enum-match-table 5 source-uri no-cache`	Enables source URI filtering for the enum match table entry. You can use the **no-cache** option to disable the caching to the **voice enum** command.
Step 4	**end** **Example:** `Device(config-enum)# end`	Returns to privileged EXEC mode.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**test enum** *match-table-index input -pattern* **source-url** *source-url* **more** *parameter* **Example:** `Device# test enum 1117777 source sip:1116666@10.1.50.16 more`	Tests the source-based routing ENUM. • The source routing or no caching features depend on the **voice enum-match-table** command. If the **source-uri** command is not configured, the **source-url** *source-url* in the test command is ignored.

	Command or Action	Purpose
	"ibcall-id=1-23735@10.1.50.16; obcall-id=7190DF-F1AA3CF1@10.1.110.222;sbc-id=1	
Step 3	**end** **Example:** `Device# end`	Returns to privileged EXEC mode.

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 /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.1.9.3.e164-test sip:540 NA     (temp, OK)   0  NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.0.9.3.e164-test sip:540 NA     (temp, OK)   0  NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```

Step 2 **show host 1.0.9.3.e164-test***

Example:

```
Device# show host  1.0.9.3.e164-test*

Host                                Port     Flags          Age Type       Address(es)
1.0.9.3.e164-test sip:540 NA      (temp, OK)  0     NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
```

```
1.0.9.3.e164-test sip:540 NA      (temp, OK)  0     NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```

Step 3 **show host 1***

Example:

```
Device# show host 1*

Host                                  Port     Flags          Age Type       Address(es)
1.0.9.3.e164-test sip:540 NA      (temp, OK)  0     NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.1.9.3.e164-test sip:540 NA      (temp, OK)  0     NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.0.9.3.e164-test sip:540 NA      (temp, OK)  0     NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```

Step 4 **show host "1.0.9.3.e164-test sip*"**

Example:

```
Device# show host "1.0.9.3.e164-test sip*"

Host                                  Port     Flags          Age Type       Address(es)
ns.e164-test                        None   (temp, OK)  0     IP          127.0.0.1
1.0.9.3.e164-test sip:540 NA      (temp, OK)  0     NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:5403@1.4.65.5"
1.0.9.3.e164-test sip:540 NA      (temp, OK)  0     NAPTR 0 0 U sip+E2U /^.*$/sip:3901@10.1.18.28/
Source-uri="sip:3401@1.4.65.5"
```

Troubleshooting Tips

Use the following commands for debugging information:

- **debug voip enum detail**
- **debug ip domain**
- **debug ccsip message**
- **debug voip ccapi 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 psedo resource record fields as per Kaplan Draft that is helpful in debugging. In the below query the values corresponding to ibcall-id, obcall-id, and sbc-id represent the incoming SIP call ID, outbound SIP call ID and Call Session Identification respectively.

```
7.7.7.7.1.1.1.e164.arpa sip:1116666@10.1.50.16enum_dns_query: name = 7.7.7.7.1.1.1.e164.arpa
sip:1116666@10.1.50.16 type = 35, ns_server = 0x0 no_cache 1 more_data
```

```
;ibcall-id=1-23735@10.1.50.16;
obcall-id=7190DF-39DD11E4-8008EDAD-F1AA3CF1@10.1.110.222;sbc-id=1
```

Configuration Examples for ENUM Enhancement per Kaplan Draft RFC

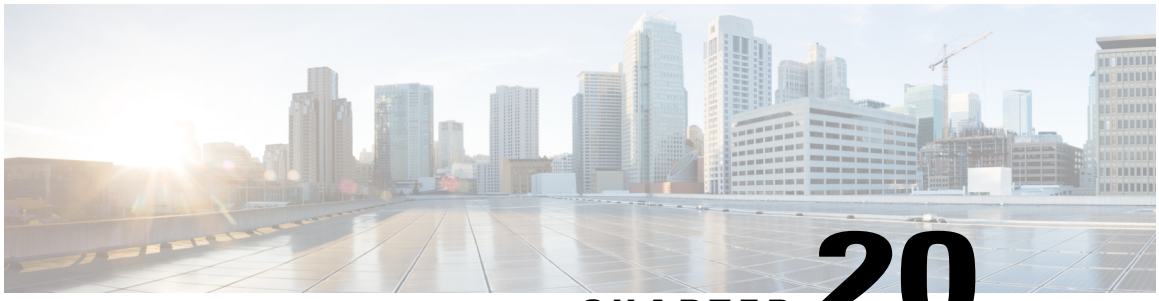
```
voice enum-match-table 1 source-uri //The source URI is sent to the DNS server to filter
the route.//
  description enable source-uri
  rule 2 1 /^\(.*\)$/ /\1/ e164.arpa

voice enum-match-table 2 source-uri no-cache
rule 1 1 /^\(.*\)$/ /\1/ e164-test

voice enum-match-table 3 no-cache //The cache table is not looked up and the route is not
cached.//
 rule 1 1 /^\(.*\)$/ /\1/ e164-test
```

The following is a sample configuration for the ENUM enhancement feature:

```
dial-peer voice 1 voip
 description ENUM Inbound dialpeer
 session protocol sipv2
 incoming called-number 1116666

dial-peer voice 2 voip
 description ENUM Outbound dialpeer
 destination-pattern 1117777
 session protocol sipv2
 session target enum:1 //Session target configured to look up ENUM table 1.//
```

PART III

Multi-Tenancy



CHAPTER 20

Support for Multi-VRF

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 sub-interface 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 Single-VRF feature for reference purpose only. For detailed information on Single-VRF feature, see http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t15/vrfawvgw.html.

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. An account on Cisco.com is not required.

Table 31: Feature Information for VRF

Feature Name	Releases	Feature Information
Support for Single-VRF (VRF-Aware)	Cisco IOS 12.4(11)XJ	This feature provides support to configure a VRF specific to voice traffic.
Support for Multi-VRF	Cisco IOS 15.6(2)T	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: **media-address voice-vrf** *name* **port-range** *min-max*, **show voice vrf**

Information About Single-VRF

Support for single-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. Single-VRF can be configured using **voice vrf** *vrf-name* command. For more information on Single-VRF, see http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t15/vrfawvgw.html.

If you are currently using single-VRF feature on CUBE and intend to upgrade to multi-VRF feature, you must perform the following:

- Remove the configurations specific to single-VRF
- Configure Multi-VRF support

Information About Multi-VRF

The Multi-VRF feature allows you to configure and maintain more than one instance of routing and forwarding tables within the same CUBE device and segregate voice traffic based on the VRF.

Multi-VRF uses input interfaces to distinguish calls for different VRFs and forms VRF tables by associating with one or more Layer 3 interfaces. Interface can be physical interface (such as FastEthernet ports, Gigabit Ethernet ports) or sub-interface. CUBE supports bridging calls on both intra-VRF and inter-VRF.

Note One physical interface or sub-interface can be associated with one VRF only. One VRF can be associated with multiple interfaces.

As per the Multi-VRF feature, the dial-peer configuration must include the use of the interface bind functionality. This is mandatory. It allows dial-peers to be mapped to a VRF via the interface bind.

The calls received on a dial-peer are processed based on the interface to which it is associated with. The interface is in turn associated with the VRF. So, the calls are processed based on the VRF table associated with that particular interface.

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 and (Unified SRST) and CUBE co-located with VRF is not supported.
- Multi-VRF call across CUBE is supported in Flow-through mode only.
- IPv6 on VRF is not supported.
- SDP pass-through is not supported with Multi-VRF.
- Calls are not supported when incoming dial-peer matched is default dial-peer (dial-peer 0).
- If DNS queries from CUBE are associated with VRF, then such queries are not supported.
- 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.
- Single-VRF (configuring VRF using **voice vrf** *vrf-name*) and Multi-VRF cannot co-exist. You must remove **voice vrf** *vrf-name* command in order to use Multi-VRF feature.

Recommendations

- For new deployments, we recommend a reboot of the router once all VRFs' are configured under interfaces.
- No VRF Route leaks are required on CUBE to bridge VoIP calls across different VRFs.
- High Availability(HA) with VRF is supported where VRF IDs are check-pointed in the event of fail-over. Ensure that same VRF configuration exists in both the HA boxes.
- Whenever destination server group is used with VRF, ensure that the server group should have the session targets, belonging to the same network as that of sip bind on the dial-peer, where the server-group is configured. This is because, dial-peer bind is mandatory with VRF and only one sip bind can be configured on any given dial-peer.

- If there are no VRF configuration changes at interface level, then reload of the router is not required.

Configuring VRF

Note We recommend you NOT to modify VRF settings on the interfaces in a live network as it requires CUBE reload to resume VRF functionality.

This section provides the generic configuration steps for creating a VRF. For detailed configuration steps specific to your network scenario (Multi-VRF and Multi-VRF with HA), refer to Configuration Examples section.

Create a VRF

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **rd** *route-distinguisher*
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
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	**ip vrf** *vrf-name* **Example:** `Device(config)# ip vrf VRF1`	Creates a VRF with the specified name. In the example, VRF name is VRF1. **Note** Space is not allowed in VRF name.
Step 4	**rd** *route-distinguisher* **Example:** `Device(config)# rd 1:1`	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)

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config)# exit`	Exits present mode.

Assign Interface to VRF

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

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

	Command or Action	Purpose
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	**interface***interface-name* **Example:** `Device(config)# interface GigabitEthernet 0/1`	Enters the interface configuration mode.

	Command or Action	Purpose
Step 4	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)# ip vrf forwarding VRF1`	Associates VRF with the interface. **Note** 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.
Step 5	**ip address** *ip address subnet mask* **Example:** `Device(config-if)# ip address 10.0.0.1 255.255.255.0`	IP address is assigned to the interface.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Exits present mode.

Create Dial-peers

Note Incoming dial-peer match criterion is not based on VRF. Configuration should ensure that appropriate incoming dial-peer is selected.

If there are overlapping DN's across different VRF's, URI based via header matching can be used for inbound dial-peer selection.

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

	Command or Action	Purpose
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** *number* **voip** **Example:** `Device(config)# dial-peer voice 1111 voip`	Creates the dial-peer with the specified number.
Step 4	**session protocol** *protocol* **Example:** `Device(config-dial-peer)# session protocol sipv2`	Specifies the protocol associated with the dial-peer.

Step 5	Create dial-peer:
	• To create inbound dial-peer:
	incoming called number *number*
	• To create outbound dial-peer:
	destination pattern *number*
	
	Example:
	Inbound dial-peer:
	`Device(config-dial-peer)# incoming called-number 1111`
	
	Example:
	Outbound dial-peer:
	`Device(config-dial-peer)# destination pattern 3333`
		Creates inbound and outbound dial-peer.

Step 6	**codec** *codec-name* **Example:** `Device(config-dial-peer)# codec g711ulaw`	Specifies the codec associated with this dial-peer.

	Command or Action	Purpose
Step 7	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits present mode.

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(conf-voi-serv)# sip
CUBE(conf-serv-sip)# call service stop
CUBE(conf-serv-sip)# no call service stop
CUBE(conf-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

	Command or Action	Purpose
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	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*
	
	Example:
	At dial-peer level:
	`Device(config)#dial-peer voice 1111 voip`
	`Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1`
	`Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1`
	
	Example:
	At global configuration level:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# sip`
	`Device(conf-voi-sip)# bind control source-interface GigabitEthernet0/1`
	`Device(conf-voi-sip)# bind media source-interface GigabitEthernet0/1`
		Interface bind associates VRF to the specified dial-peer.

Step 4	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits present mode.

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

	Command or Action	Purpose
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	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters voice service voip mode.
Step 4	**media-address voice-vrf** *vrf-name* **port-range** *min max* **Example:** **Example:**	Associates the RTP Port range with the VRF.

	Command or Action	Purpose

	Example 1
	`Device(conf-voi-serv)#media-address voice-vrf VRF1 port 16000 32000`
	
	The output:
	`Device#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 2
	`Device(conf-voi-serv)#media-address voice-vrf VRF1`
	`Device(cfg-media-addr-vrf)#port-range 6000 7000`
	`Device(cfg-media-addr-vrf)#port-range 8000 10000`
	`Device(cfg-media-addr-vrf)#port-range 11000 20000`
	
	The output:
	`Device#voice`
	`voice-card 0/3`
	` dsp services dspfarm`
	`voice service voip`
	` no ip address trusted authenticate`
	` media-address voice-vrf VRF1`
	`  port-range 6000 7000`
	`  port-range 8000 10000`
	`  port-range 11000 20000`
	*In this case, multiple port range lines are configured under the media address.
	You can configure up to ten port ranges per media address.	

Step 5	**exit** **Example:** `Device(conf-voi-serv)# exit`	Exits present mode.

Example: VRF with overlapping and non-overlapping RTP Port Range

Example 1 - Non-overlapping Port Range

The following is 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

Found 2 active RTP connections
```

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
                                                          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                                    25000 28000 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       25000      16392      11.0.0.1     11.0.0.2        NO      vrf2

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.

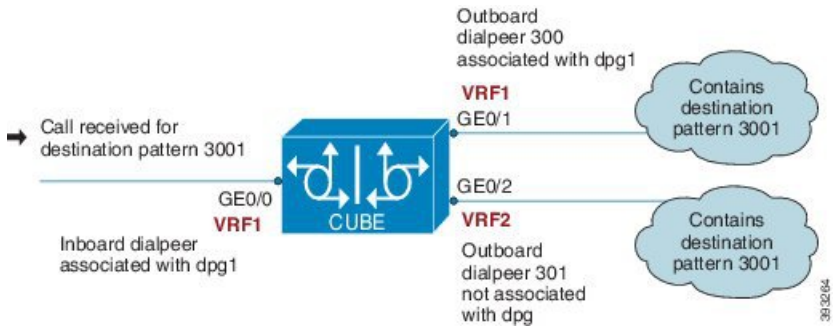
To use dial-peer group feature, configure dial-peers such that there is a unique inbound dial-peer match for calls related to each VRF. Configuring dial-peer group, limits the outbound dial-peer search within the VRF.

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 19: Associating Dial-peer Group to overcome DN overlap



The following scenario is considered in the below example:

- VRF1 associated with Gigabitethernt Interface 0/0 and 0/1
- VRF 2 associated with Gigabitethernet Inetrface 0/2
- Dial-peer Group: dpg1
- VRF1 is associated with dial-peer group - dpg 1
- Outbound dial-peer 300 is selected as preference 1
- Inbound dial-peer 3000 associated with VRF 1 and dial-peer group 1 (dpg1)
- Outbound Dial-peer: 300 – destination pattern “3001” associated with VRF1
- Outbound dial-peer: 301 – destination pattern “3001” associated with VRF2

Configure a dial-peer group and set the outbound dial-peer preference.

```
Device# enable
Device# configure terminal
Device(config)# voice class dpg 1
Device(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

Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1
Device(config-dial-peer)# dtmf-relay sip-kpml
Device(config-dial-peer)# srtp fallback
Device(config-dial-peer)# codec g711ulaw
```

Creating outbound dial-peer with destination pattern ‘3001’ associated with VRF1.

```
Device(config)# dial-peer voice 300 voip
Device(config-dial-peer)# destination-pattern 3001
Device(config-dial-peer)# video codec h264
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:10.0.0.1
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1
Device(config-dial-peer)# dtmf-relay sip-kpml
Device(config-dial-peer)# codec g711ulaw
```

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

```
Device(config)# dial-peer voice 301 voip
Device(config-dial-peer)# destination-pattern 3001
Device(config-dial-peer)# video codec h264
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target ipv4:11.0.0.1
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/2
Device(config-dial-peer)# dtmf-relay sip-kpml
Device(config-dial-peer)# codec g711ulaw
```

With above dial-peer group configuration, whenever dial-peer “3000” is matched as inbound dial-peer, CUBE will always route call using dial-peer “300” (VRF1). Without dial-peer group, CUBE would have picked dial-peers “300”(VRF1) and “301”(VRF2) in random to route the call.

```
Device# show vrf brief
  Name                   Default RD              Protocols   Interfaces
  VRF1                      1:1                      ipv4         Gi0/0
                                                                  Gi0/1
  VRF2                      2:2                      ipv4         Gi0/2

Device# show dial-peer voice summary
dial-peer hunt 0
                    AD                          PRE  PASS                      OUT
TAG    TYPE  MIN  OPER PREFIX DEST-PATTERN  FER  THRU      SESS-TARGET    STAT PORT
KEEPALIVE    VRF
3000   voip   up     up                      0     syst
         VRF1
300    voip   up     up        3001          0     syst   ipv4: 10.0.0.1
         VRF1
301    voip   up               3001          0     syst   ipv4: 11.0.0.1
         VRF2
```

IP Overlap with VRF

Generally, on a router, two interfaces cannot be configured with the same IP address. With VRF feature, you can configure two or more interfaces with the same IP address. This is possible 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 GigbitEthernet 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 GigabitEthernet 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 needs to be configured in the network.

Even though GigabitEthernet 0/0 and GigabitEthernet 0/1 have 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 overlapping IP address:

```
Device# show ip interface brief
Interface                  IP-Address      OK? Method Status           Protocol
Embedded-Service-Engine0/0 unassigned      YES NVRAM  administratively down down
GigabitEthernet0/0         10.0.0.0        YES NVRAM  up                    up
GigabitEthernet0/1         10.0.0.0        YES NVRAM  up                    up
GigabitEthernet0/1.1       unassigned      YES NVRAM  up                    up
GigabitEthernet0/2         unassigned      YES NVRAM  up                    up
```

show voip rtp connections command shows a video call established on CUBE across different interfaces belonging to different VRF's 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
                                        Min   Max   Ports     Ports     Ports
Media-Address Range                     Port  Port  Available Reserved  In-use
-----------------------------------------------------------------------------
Global Media Pool                       20000 22000 900       101       0

VRF ID Based Media Pool
-----------------------------------------------------------------------------
POD2                                    30002 32000 1000      0         0
POD1                                    20000 30000 4900      101       2
POD3                                    20000 30000 4900      101       2
-----------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId     dstCallId  LocalRTP RmtRTP   LocalIP    RemoteIP   MPSS  VRF
1     37         39         20000    18164  10.0.0.0   11.0.0.3   NO    VRF1
2     38         40         20002    18166  10.0.0.0   11.0.0.3   NO    VRF1
3     39         37         20002    16388  10.0.0.0   11.0.0.3   NO    VRF2
4     40         38         20000    16390  10.0.0.0   11.0.0.3   NO    VRF2
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
Interface              IP-Address  OK?   Method   Status            Protocol
GigabitEthernet0/0/0   10.0.0.1     YES  NVRAM     up                  up
GigabitEthernet0/0/1   11.0.0.1     YES  NVRAM     up                  up
GigabitEthernet0/0/2   12.0.0.1     YES  NVRAM     up                  up
```

- dial-peer 200 is bind to GigabitEthernet0/0/1
- server-group 1 (belonging to VRF1) is applied to dial-peer 200

```
Device(config)# dial-peer voice 200 voip
Device(config-dialpeer)# destination-pattern 4.....
Device(config-dialpeer)# session protocol sipv2
Device(config-dialpeer)# session transport udp
Device(config-dialpeer)# session server-group 1
Device(config-dialpeer)# voice-class sip bind control source-interface GigabitEthernet0/0/1
Device(config-dialpeer)# voice-class sip bind media source-interface GigabitEthernet0/0/1
Device(config-dialpeer)# codec g711ulaw
```

As dial-peer 200 is bind to GigabitEthernet0/0/1 , the session targets configured in the “server-group 1” should belong to the network which is reachable by the bind source interface GigabitEthernet0/0/1 as shown below:

```
Device(config)# voice class server-group 1
 Device(config-class)# ipv4 11.0.0.22
 Device(config-class)# ipv4 11.0.0.8 preference 2
```

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.

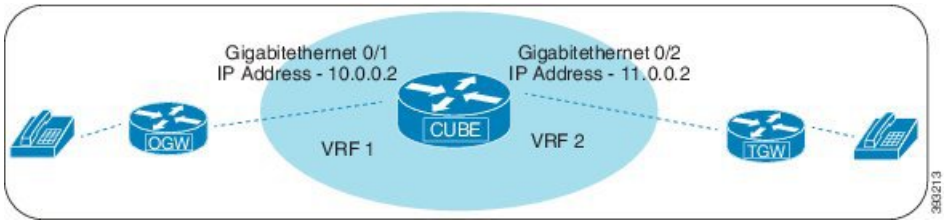
Configuration Examples

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 20: Multi-VRF in Standalone Mode



Configuring VRF

```
Device# enable
Device# configure terminal
Device(config)# ip vrf VRF1
Device(config)# rd 1:1
```

```
Device(config)# ip vrf  VRF2
Device(config)# rd 2:2
Device(config)# exit
```

Associating interfaces with VRF

```
Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip vrf forwarding VRF1
Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip vrf forwarding VRF2
```

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

Configure Interface GigabitEthernet0/1

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet0/1
Device(config-if)# ip address 10.0.0.2 255.255.255.0
Device(config-if)# speed auto
Device(config-if)# exit
```

Configure Interface GigabitEthernet0/2

```
Device(config)# interface GigabitEthernet0/2
Device(config-if)# ip address 11.0.0.2 255.255.255.0
Device(config-if)# speed auto
Device(config-if)# exit
```

Creating Dial-peer

Creating Inbound Dial-peer:

```
Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# incoming called-number 1111
Device(cofig-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

                  AD                          PRE  PASS                      OUT
TAG   TYPE  MIN OPER PREFIX DEST-PATTERN FER  THRU   SESS-TARGET   STAT PORT KEEPALIVE  VRF
1111  voip  up   up             -         0   syst  ipv4:10.0.0.2
VRF1
2222  voip  up   up             -         0   syst  ipv4:11.0.0.2
VRF2
```

Configure 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.
- Whenever global sip bind interface associated with a VRF is added, modified, or removed, you should restart the sip services under voice service voip sip mode so that the change in global sip bind comes into effect with associated VRF ID.

```
Device(config)# voice service voip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# call service stop
Device(conf-serv-sip)# no call service stop
Device(conf-serv-sip)# end
```

```
Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/1


Device(config)# dial-peer voice 2222 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config-dial-peer)# voice-class sip bind media source-interface GigabitEthernet0/2
```

Execute the following command to verify the interface association with VRF:

```
Device# show ip vrf brief

  Name                             Default    RD             Interfaces
  Mgmt-intf                        <not set>                    Gi0
  VRF1                              1:1                         Gi0/1
  VRF2                              2:2                         Gi0/2
```

Execute the following command to verify a successful and active calls:

For a single call, you should be able to see two RTP connections as shown in the below example.

```
Device# show voip rtp connections

VoIP RTP Port Usage Information:
Max Ports Available: 23001, Ports Reserved: 101, Ports in Use: 2
                                       Min   Max   Ports     Ports     Ports
Media-Address Range                    Port  Port  Available Reserved  In-use
-------------------------------------------------------------------------------
Global Media Pool                      8000  48198 19999     101       0

-------------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId  dstCallId  LocalRTP RmtRTP LocalIP   RemoteIP   MPSS  VRF
1     1        2        25000   16390  10.0.0.1  10.0.0.2   NO    VRF1
2     2        1        25002   16398  11.0.0.1  11.0.0.2   NO    VRF2

Device# show call active voice brief  -


Perf-AR1006#show call active voice brief
<ID>: <CallID> <start>ms.<index> (<start>) +<connect> pid:<peer_id> <dir> <addr> <state>
  dur hh:mm:ss tx:<packets>/<bytes> rx:<packets>/<bytes> dscp:<packets violation>
media:<packets violation> audio tos:<audio tos value> video tos:<video tos value>
 IP <ip>:<udp> rtt:<time>ms pl:<play>/<gap>ms lost:<lost>/<early>/<late>
  delay:<last>/<min>/<max>ms <codec> <textrelay> <transcoded

 media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>

 long duration call detected:<y/n> long duration call duration :<sec> timestamp:<time>
 LostPacketRate:<%> OutOfOrderRate:<%>
 VRF:<%>
```

```
  MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
   last <buf event time>s dur:<Min>/<Max>s
 FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 Tele <int> (callID) [channel_id] tx:<tot>/<v>/<fax>ms <codec> noise:<l> acom:<l> i/o:<l>/<l>
 dBm
  MODEMRELAY info:<rcvd>/<sent>/<resent> 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>
  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: 0
Multicast call-legs: 0
Total call-legs: 2
11FF : 8565722 511605450ms.1 (*16:21:53.676 IST Tue Aug 4 2015) +30 pid:400001
Answer 777412373 active
 dur 00:00:22 tx:1110/66600 rx:1111/66660 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 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
 VRF: VRF1
11FF : 8565723 511605470ms.1 (*16:21:53.696 IST Tue Aug 4 2015) +0 pid:400000 Originate
777512373 active
 dur 00:00:22 tx:1111/66660 rx:1110/66600 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 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
 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

Device# show sip-ua connections udp brief

Total active connections       : 2
No. of send failures            : 0
No. of remote closures          : 0
No. of conn. failures           : 0
No. of inactive conn. ageouts   : 2

--------------- SIP Transport Layer Listen Sockets ----------------
  Conn-Id                Local-Address
 ============    ============================
   2              [10.0.0.1]:5060:VRF1
   3              [11.0.0.1]:5060:VRF2


Device# show call active voice compact

 <callID>  A/O  FAX T<sec>  Codec       type  Peer Address  IP R<ip>:<udp>   VRF
Total call-legs: 2
   8565722 ANS     T12    g711ulaw     VOIP   P777412373   10.0.0.2:30804   VRF1
```

```
   8565723 ORG     T12    g711ulaw     VOIP   P777512373   11.0.0.2:30804   VRF2
```

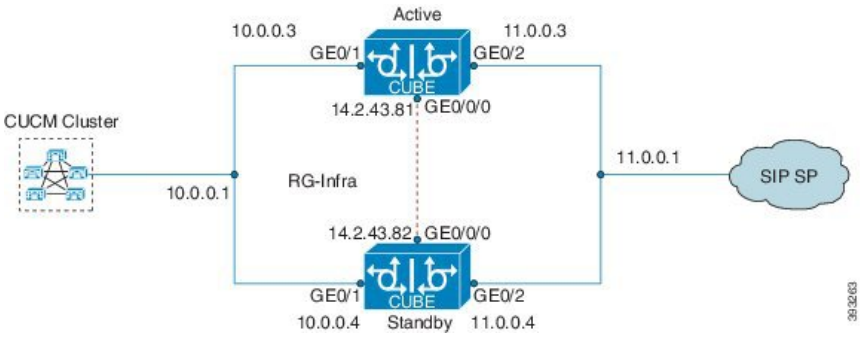
```
Device# show call active video compact
MVRF-CUBE1#show call active video compact
 <callID>  A/O FAX T<sec> Codec type     Peer Address  IP R<ip>:<udp>    VRF
Total call-legs: 2
  10193983 ANS     T30    H264  VOIP-VIDEO  P2005      10.0.0.2:18078    VRF1
  10193985 ORG     T30    H264  VOIP-VIDEO  P3001      11.0.0.2:27042    VRF2
```

Example: Configuring RG Infra High Availability with VRF

Note Below configuration example is applicable for Cisco ASR 1000 Series Aggregated Services Routers (ASR) and Cisco 4000 Series Integrated Services Routers (ISR G3).

Note Do not configure VRF on the interface that is used for RG Infra. Traffic of VRF and RG Infra should be on different interfaces.

Figure 21: 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.

```
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)# bfd interval 50 min_rx 50 multiplier 3
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)# bfd interval 50 min_rx 50 multiplier 3
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

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

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.

```
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)# bfd interval 50 min_rx 50 multiplier 3
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)# bfd interval 50 min_rx 50 multiplier 3
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)# 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
                                        Min   Max   Ports     Ports     Ports
Media-Address Range                     Port  Port  Available Reserved  In-use
-----------------------------------------------------------------------------
Global Media Pool                       8000  48198 19999     101       2
-----------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId     dstCallId   LocalRTP  RmtRTP   LocalIP     RemoteIP    MPSS   VRF
1     5              6       8008     16388   10.0.0.1    10.0.0.2    NO     VRF1

2     6              5       8010     16388   11.0.0.1    11.0.0.2    NO     VRF2
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
                                        Min   Max   Ports     Ports     Ports
Media-Address Range                     Port  Port  Available Reserved  In-use
-----------------------------------------------------------------------------
Global Media Pool                       8000  48198 19999     101       2
-----------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId     dstCallId   LocalRTP   RmtRTP    LocalIP     RemoteIP      MPSS    VRF
1     7              8       8012      16390    10.0.0.1    10.0.0.2       NO     VRF1

2     8              7       8014      16390    11.0.0.1    11.0.0.2       NO     VRF2

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 SRTP: 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
```

```
11F3 : 6 243854170ms.2 (*11:48:43.972 UTC Mon May 25 2015) +6770 pid:3333 Originate 2222
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 SRTP: 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:

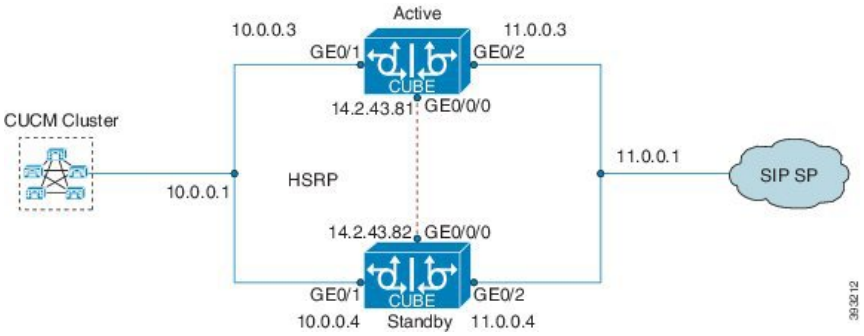
```
Device# show call active voice brief

11F9 : 8 245073830ms.1 (*12:16:18.094 UTC Mon May 25 2015) +26860 pid:3333 Originate 2222
connected
 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 pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:
 off Transcoded: No ICE: Off
 media inactive detected:n media contrl rcvd:n/a timestamp:n/a
 long duration call detected:n long duration call duration:n/a timestamp:n/a
 LostPacketRate:0.00 OutOfOrderRate:0.00

11F9 : 7 245073850ms.1 (*12:16:18.114 UTC Mon May 25 2015) +26840 pid:0 Answer  connected
 dur 00:03:37 tx:6757/405420 rx:6757/405420 dscp:0 media:0 audio tos:0x0 video tos:0x0
 IP 10.0.0.2:16390 SRTP: off rtt:65523ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:
 off Transcoded: No ICE: Off
 media inactive detected:n media 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

Note 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 22: 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
```

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)# 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

Note Control and Media on a dial-peer have to bind with same VRF. Else, while configuring, the CLI parser will display an error.

```
Device(config)# dial-peer voice 1111 voip
Device(config-dial-peer)# voice-class sip bind control source-interface GigabitEthernet0/1
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/1

Device(config)# dial-peer voice 3333 voip
Device(config)# voice-class sip bind control source-interface GigabitEthernet0/2
Device(config)# voice-class sip bind media source-interface GigabitEthernet0/2
```

Verification of redundancy States

On Active Router

```
Device(config)# show redundancy status

my state = 13 -ACTIVE
peer state = 8  -STANDBY HOT
Mode = Duplex
Unit ID = 0

Maintenance Mode = Disabled
Manual Swact = enabled
Communications = Up

client count = 17
client_notification_TMR = 120000 milliseconds
RF debug mask = 0x0
```

On Standby Router

```
Device(config)# show redundancy status

my state = 8 -STANDBY HOT
peer state = 13 ACTIVE
Mode = Duplex
Unit ID = 0

Maintenance Mode = Disabled
Manual Swact = enabled
Communications = Up

client count = 17
client_notification_TMR = 120000 milliseconds
RF debug mask = 0x0
```

Verification of Calls Before and After Switchover

RTP Connections on Active router:

```
Device# show voip rtp connections

VoIP RTP Port Usage Information:
```

```
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 2
                                        Min   Max   Ports     Ports     Ports
Media-Address Range                      Port  Port  Available Reserved  In-use
-----------------------------------------------------------------------------
Global Media Pool                       8000  48198 19999     101       2
-----------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId     dstCallId   LocalRTP  RmtRTP   LocalIP     RemoteIP    MPSS  VRF
1     5             6       8008     16388   10.0.0.1    10.0.0.2    NO    VRF1

2     6             5       8010     16388   11.0.0.1    11.0.0.2    NO    VRF2
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
                                        Min   Max   Ports     Ports     Ports
Media-Address Range                      Port  Port  Available Reserved  In-use
-----------------------------------------------------------------------------
Global Media Pool                       8000  48198 19999     101       2
-----------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId     dstCallId            LocalRTP        RmtRTP      LocalIP         RemoteIP
      MPSS   VRF
1     7             8                8012         16390      10.0.0.1         10.0.0.2
       NO    VRF1
2     8             7                8014         16390      11.0.0.1         11.0.0.2
       NO    VRF2
Found 2 active RTP connections
```

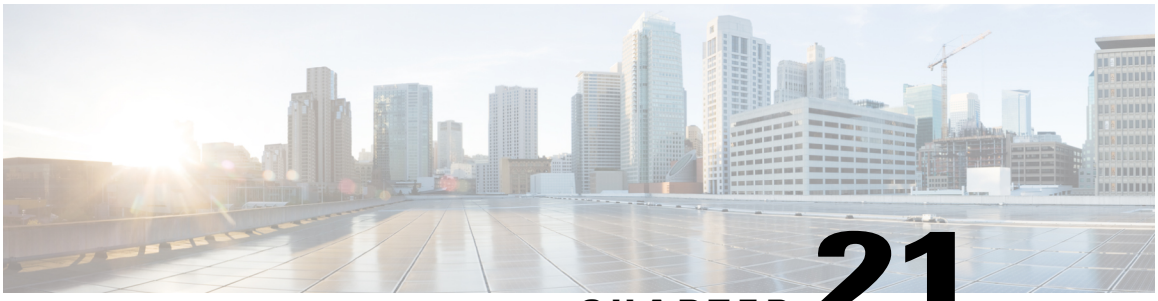
Active calls on Active Router

```
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 SRTP: 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

11F3 : 6 243854170ms.2 (*11:48:43.972 UTC Mon May 25 2015) +6770 pid:3333 Originate 2222
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 SRTP: 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

11F9 : 8 245073830ms.1 (*12:16:18.094 UTC Mon May 25 2015) +26860 pid:3333 Originate 2222
connected
 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 pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:
 off Transcoded: No ICE: Off
 media inactive detected:n media contrl rcvd:n/a timestamp:n/a
 long duration call detected:n long duration call duration:n/a timestamp:n/a
 LostPacketRate:0.00 OutOfOrderRate:0.00

11F9 : 7 245073850ms.1 (*12:16:18.114 UTC Mon May 25 2015) +26840 pid:0 Answer  connected
 dur 00:03:37 tx:6757/405420 rx:6757/405420 dscp:0 media:0 audio tos:0x0 video tos:0x0
 IP 10.0.0.2:16390 SRTP: off rtt:65523ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729r8 TextRelay:
 off Transcoded: No ICE: Off
 media inactive detected:n media 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
```



CHAPTER 21

Configuring Multi-Tenants on SIP Trunks

The 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

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 Name	Releases	Feature Information
Support for Configuring Multi Tenants on SIP Trunks	Cisco IOS 15.6(2)T	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

In the previous releases of Cisco IOS software, CUBE supported only single tenancy, with sip-specific attributes configured either globally or under dial-peer. With the introduction of multi-tenancy support on CUBE, the sip-specific attributes can now 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 32: 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 32: Multi-Tenant Configuration List

Command	Description
aaa	SIP-UA AAA related configuration
anat	Allow alternative network address types IPv4 and IPv6
asserted-id	Configure SIP UA privacy identity settings
associate	Associate a RCB for outgoing calls
asymmetric	Configure global SIP asymmetric payload support

Command	Description
authenticate	Call authentication policy
authentication	Digest Authentication Configuration
bandwidth	Allow SIP SDP bandwidth-related options
bind	SIP bind command
block	Block 18X response to INVITE
call-route	Configure call routing options
conn-reuse	Reuse the sip registration tcp connection for the end-point behind a Firewall
connection-reuse	Use listener port for sending requests over UDP
contact-passing	302 contact to be passed through for CFWD
content	Content carried as part of SIP message
copy-list	Configure list of entities to be sent to peer leg
credentials	User credentials for registration
disable-early-media	Disable early-media cut through
dns -a-override	Skip DNS A/AAAA query when SRV query timesout
dscp -profile	DSCP Profile global config
early-media	Configure method to handle early-media Update Request
early-offer	Configure sending Early-Offer
encap	Configure SDP encapsulation
error-code-override	Configure sip error code
error- passthru	SIP error response pass-thru functionality

exit	Exits from the voice class configuration mode
g729	G729 codec interoperability settings
handle-replaces	Handle INVITE with REPLACES header at SIP spi

header-passing	SIP Headers need to be passed to Applications
help	Description of the interactive help system
history-info	History Info header support
host-registrar	Use sip-ua registrar value in Diversion and Contact header for 3xx messages
interop-handling	Enable interop-handling
localhost	Specify the DNS name for the localhost
map	Mapping options
max-forwards	Change number of max-forwards for SIP Methods
midcall -signaling	Configure method to handle mid-call signaling
nat	SIP nat global config
no	Negate a command or set its defaults
notify	SIP Signaling Notify Configuration
offer	Configure settings for Offers made from the Gateway
options-ping	Send OPTION pings to remote end
outbound-proxy	Configure an Outbound Proxy Server
pass-thru	SIP pass-through global config
permit	Permit hostname for this gateway
preloaded-route	Use pre-loaded route header for outgoing calls, if available
privacy	Configure SIP UA privacy settings
privacy-policy	Set privacy behavior for outgoing SIP messages
random-contact	Use Random Contact for outgoing calls, if available
random-request- uri	Configure options for Request-URI having random value

reason-header	Configure settings for supporting SIP Reason Header

redirection	Enable call redirection (3xx) handling
refer- ood	Configure maximum number of out-of-dialog refer made to the Gateway
referto -passing	Refer-To needs to be passed through for transfer
registrar	Configure SIP registrar VoIP Interface
registration	Enable registration options
rel1xx	Type of reliable provisional response support
remote-party-id	Enable Remote-Party-ID support in SIP User Agent
requri -passing	Request URI needs to be passed through
reset	SIP Reset Options
retry	Change default retries for each SIP Method
send	Configure outgoing message options
session	SIP Voice Protocol session config
sip-profiles	SIP Profiles global config
sip-server	Configure a SIP Server Interface
srtp	Allow SIP related SRTP options
srtp-auth	Allow to set preferred suites
tel-config	Tel format cfg for headers other than req -line in
timers	SIP Signaling Timers Configuration
update- callerid	Enable sending updates for callerid
url	Url configuration for request-line url in outgoing INVITE
video	Video related config for sip
warn-header	SIP Warning-Header global config

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 3	Use the following commands to configure multi-tenants:
	• **voice class tenant <tag>** in the global configuration mode
	Once you configure the **voice class tenant <tag>** command in the global mode, the configuration will move to the **voice class tenant <tag>** submode. You can configure all the sip-specific attributes in this submode.
	• **voice-class sip tenant <tag>** in the dial-peer configuration mode
	
	Example:
	In global configuration mode
	`! Configuring tenant 1`
	`Device(config)# voice class tenant 1`
	`Device (config-class)# ?`
		Use the **voice-class sip tenant <tag>** 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 **voice-class sip tenant <tag>** command under the dial-peers.

	Command or Action	Purpose
	aaa – sip-ua AAA related configuration anat – Allow alternative network address types IPV4 and IPV6 asserted-id – Configure SIP-UA privacy identity settings .... Video – video related function Warn-header – SIP related config for SIP. SIP warning-header global config. Device (config-voi-tenant)# end -------- **! Configuring tenant 2** Device(config)# **voice class tenant 2** Device (config-class)# **?** aaa – sip-ua AAA related configuration anat – Allow alternative network address types IPV4 and IPV6 asserted-id – Configure SIP-UA privacy identity settings .... outbound-proxy - Configure an Outbound Proxy Server pass-thru - SIP pass-through global config .... srtp - Allow SIP related SRTP options Warn-header – SIP related config for SIP. SIP warning-header global config. Device (config-voi-tenant)# **end** **Example:** In dial-peer configuration mode **!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** **Example:** Device(config-dial-peer)# end	Returns to privileged EXEC mode.

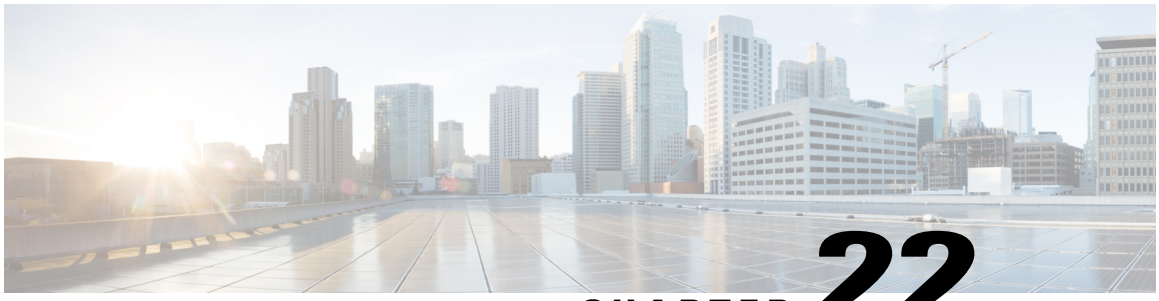
Example: SIP Trunk Registration in Multi-Tenant Configuration

For SIP trunk registration, the **voice class tenant <tag>** command is not associated with any dial-peer configuration. All outgoing registrations are triggered to the Registrars when credentials are configured under **voice class tenant <tag>**.

```
Router# show run | sec tenant

Voice class tenant 1
registrar 1 ipv4:10.64.86.35:9051 expires 3600
credentials username aaaa password 7 06070E204D realm aaaa.com
outbound-proxy ipv4:10.64.86.35:9057
bind control source-interface GigabitEthernet0/0

Voice class tenant 2
registrar 1 ipv4:9.65.75.45:9052 expires 3600
credentials username bbbb password 7 110B1B0715 realm bbbb.com
outbound-proxy ipv4:10.64.86.40:9040
bind control source-interface GigabitEthernet0/1
```

PART IV

Codecs



CHAPTER 22

Codec Support and Restrictions

This chapter provides advanced information about the support of and restrictions for certain codecs on CUBE. For basic information on how to configure codecs, refer to the Introduction to Codecs section.

Feature Information for Codec Support on CUBE

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

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

Table 33: Feature Information for Codec Support on CUBE

Feature Name	Releases	Feature Information
AAC-LD MP4A-LATM Codec Support on Cisco UBE	15.4(1)T Cisco IOS XE Release 3.12S	The AAC-LD MP4A-LATM codec is a wideband audio codec used by video endpoints. MP4A-LATM is an MPEG4 audio coding standard, where LATM is Low-Overhead MPEG-4 Audio Transport Multiplex. The Cisco Unified Border Element (Cisco UBE) supports MP4A-LATM to enable call flows involving endpoints that use this codec, especially for media recording. The following commands were introduced or modified: **codec mp4a-latm**, **codec preference tag mp4a-latm**
ISAC Codec Support on CUBE	15.1(1)T	The ISAC Codec Support on CUBE The following commands were introduced by this feature: **codec isac**, **codec preference tag isac**.

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:

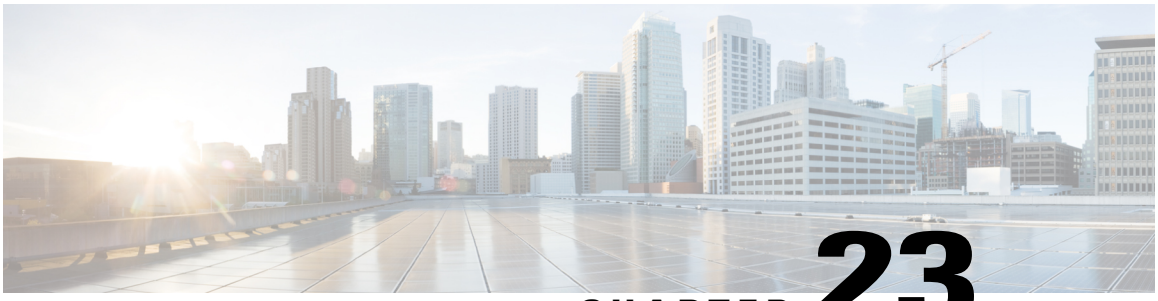
- Accept and send MP4A-LATM codec and corresponding FMTP profiles
- Configure MP4A-LATM under dial-peer or under voice-class codec as preferred codec
- Pass across real-time transport protocol (RTP) media for MP4A-LATM codec without any interworking

- Offer pre-configured FMTP profile for MP4A-LATM for DO-EO (Delayed-Offer to Early-Offer) calls
- Offer more than one FMTP profile (each with different payload type number) as mentioned by the offering endpoint, so that the answering endpoint can choose the best option.
- Offer only one instance of MP4A-LATM if media forking is applicable. The offered instance is the first one received in the offer.
- Calculate bandwidth for MP4A-LATM on the basis of either “b=TIAS” attribute or “bitrate” parameter in the FMTP attribute. If none of them are present in the session description protocol (SDP), the default maximum bandwidth, that is, 128 Kbps will be used for calculation.
- The following Cisco UBE features are supported with the MP4A-LATM codec:
  - Basic call (audio and video) flow-around and flow-through (FA and FT).
  - Voice Class Codec support in Cisco UBE with codec filtering
  - SRTP and SRCTP passthrough for SIP-to-SIP calls
  - Supplementary services
  - RSVP
  - Dynamic payload type interworking for DTMF and codec packets for SIP-to-SIP calls
  - Media Anti-Trombone with SIP signaling control on CUBE
  - Support for SIP UPDATE message per RFC 3311
  - RTP Media Loopback
  - Media forking for IP based calls using Zephyr recording server
  - Cisco UBE Mid-call Re-INVITE consumption
  - Signaling forking (Fastweb multile SIP Early Dialog Support, FA and FT)
  - Maximum bandwidth-based CAC
  - Media Policing
  - Box-to-Box High Availability (B2B HA)
  - Inbox High Availability (Inbox HA)

Restrictions for AAC-LD MP4A-LATM Codec Support on Cisco UBE

Cisco UBE does not support the following:

- Codec transcoding between MP4A-LATM and other codecs
- Dual-tone Multifrequency (DTMF) interworking with MP4A-LATM codec
- Non-SIP-SIP, that is, SIP to other service provider interface (SPI) interworking with MP4A-LATM codec



CHAPTER 23

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

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 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 Name	Releases	Feature Information
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	15.1(2)T	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).**
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	Cisco IOS XE Release 3.8S	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)**.
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.	15.3(2)T	This feature provides high availability support 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 under the Voice Class Codec.

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.
- Mid-call 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 inband DTMF interworking required.

- Reinvite-based supplementary services 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 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 mid-call renegotiation, if the call releases the transcoder, then the call is checkpointed.

Prerequisites for Codec Preference Lists

- Transcoding configuration on the CUBE.
- The digital signal processor (DSP) requirements to support the transcoding feature on the CUBE.

Restrictions for Codecs Preference Lists

For All Calls (SIP-to-SIP, H323-to-H323, SIP-to-H323 calls)

- Video codecs are not supported with preference lists.
- Multiple audio streams are not supported.
- High-density transcoding is not supported when delayed offer to early offer is configured. Only low density transcoding is supported.
- Codec re-packetization feature is not supported when preference lists are configured.

For H323-to-H323 and SIP-to-H323 Calls

The below restrictions do not exist for SIP-to-SIP calls from 15.1(2)T and Cisco IOS XE Release 3.8S onwards.

- You can configure dissimilar preference lists on the incoming and outgoing dial peers.
- Incoming and outgoing dial-peers cannot be configured with the different preference lists.
- Transcoding is not supported when preference lists are used.
- Mid-call codec changes and supplementary services (call-hold / resume, call forward) do not work when a preference list is configured.
- Mid-call insertion or deletion of transcoder is not supported with preference lists.
- Rotary dial peers are not supported when preference lists are used.
- Both incoming and outgoing dial-peers need to be configured with the same codec voice classes.
- The preference of codecs configured in a codec voice classes is not be applied to the outgoing call-leg. Basically codec filtering is applied first and only the filtered codecs will be sent out in the outgoing offer from CUBE.

- T.38 fax, fax-passthru and modem-passthru is not be supported with preference lists.
- SRTP<->RTP is not supported with preference lists.
- When a codec voice class is configured, call establishment is un-predictable when a transcoder is involved in the call. The call succeeds only if the end points choose the first codec in the list of offered codecs.

How to Configure Codec Preference Lists

Configuring Audio Codecs Using a Codec Voice Class and Preference Lists

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

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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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	Purpose
Step 3	**voice class codec** *tag* **Example:** `Device(config)# voice class codec 10`	Enters voice-class configuration mode for the specified codec voice class.
Step 4	Do the following for each audio codec you want to configure in the voice class: • **codec preference** *value codec-type*[**bytes** *payload-size* **fixed-bytes**] • **codec preference** *value* **isac** [**mode {adaptive | independent}** [**bit-rate** *value* **framesize { 30 | 60 } [fixed]**] • **codec preference** *value* **ilbc** [**mode** *frame-size* [**bytes** *payload-size*]] • **codec preference** *value* **mp4-latm** [**profile** *tag*]	Configure a codec within the voice class and specifies a preference for the codec. This becomes part of a preference list
Step 5	**exit** **Example:** `Device(config-class)# exit`	Exits the current mode. • Enter your password if prompted.
Step 6	**dial-peer voice** *number* **voip** **Example:** `Device(config)# dial-peer voice 1 voip`	Enters dial peer configuration mode for the specified VoIP dial peer.
Step 7	**voice-class codec** *tag* **offer-all** **Example:** `Device(config-dial-peer)# voice-class codec 10`	Applies the previously configured voice class and associated codecs to a dial peer. • The **offer-all** keyword allows the device to offer all codecs configured in a codec voice class.
Step 8	**end** **Example:** `Device(config-dial-peer)# end`	Returns to privileged EXEC mode.

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

	Command or Action	Purpose
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 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**
2. **show call active voice brief**
3. **show voip rtp connections**
4. **show sccp connections**
5. **show dspfarm dsp active**

DETAILED STEPS

Step 1 **enable**
Enables privileged EXEC mode.

Step 2 **show call active voice brief**
Displays a truncated version of call information for voice calls in progress.

Example:

```
Device# show call active voice brief
<ID>: <CallID> <start>ms.<index> +<connect> pid:<peer_id> <dir> <addr> <state>
  dur hh:mm:ss tx:<packets>/<bytes> rx:<packets>/<bytes>
 IP <ip>:<udp> rtt:<time>ms 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>s dur:<Min>/<Max>s
 FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 Tele <int> (callID) [channel_id] tx:<tot>/<v>/<fax>ms <codec> noise:<l> acom:<l> i/o:<l>/<l> dBm
  MODEMRELAY info:<rcvd>/<sent>/<resent> 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>
  tx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
 rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 2
Multicast call-legs: 0
Total call-legs: 4
1243 : 11 971490ms.1 +-1 pid:1 Answer 1230000 connecting
 dur 00:00:00 tx:415/66400 rx:17/2561
 IP 192.0.2.1:19304 SRTP: off rtt:0ms 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:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729br8 TextRelay: off
 media inactive detected:n media contrl rcvd:n/a timestamp:n/a
 long duration call detected:n long duration call duration:n/a timestamp:n/a
0    : 13 971560ms.1 +0 pid:0 Originate  connecting
 dur 00:00:08 tx:415/66400 rx:17/2561
 IP 192.0.2.2:2000 SRTP: off rtt:0ms 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    : 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:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms g729br8 TextRelay: off
 media inactive detected:n media contrl rcvd:n/a timestamp:n/a
 long duration call detected:n long duration call duration:n/a timestamp:n/a
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 2
Multicast call-legs: 0
Total call-legs: 4
```

Step 3 **show voip rtp connections**

Displays Real-Time Transport Protocol (RTP) connections.

Example:

```
Device# show voip rtp connections
VoIP RTP active connections :
No. CallId     dstCallId  LocalRTP RmtRTP     LocalIP                                  RemoteIP
1     11         12         16662    19304    192.0.2.1
192.0.2.2
2     12         11         17404    16512    192.0.2.2
192.0.2.3
3     13         14         18422    2000     192.0.2.4
9.44.26.3
4     15         14         16576    2000     192.0.2.6
```

```
192.0.2.5
Found 4 active RTP connections
```

Step 4 **show sccp connections**

Displays information about the connections controlled by the Skinny Client Control Protocol (SCCP) transcoding and conferencing applications.

Example:

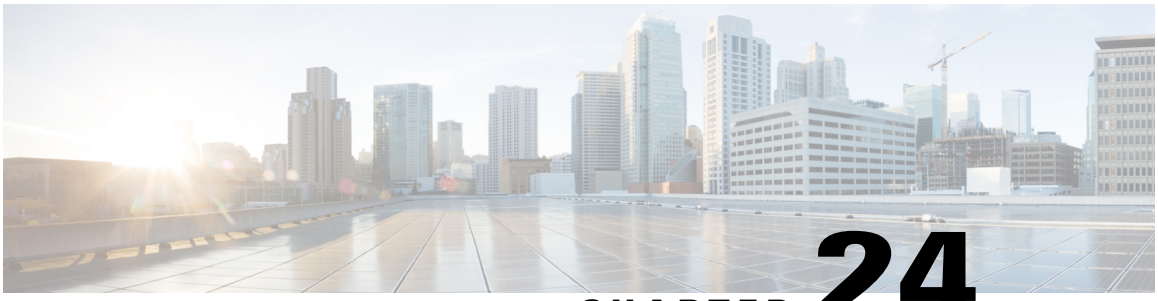
```
Device# show sccp connections
sess_id    conn_id      stype mode     codec   sport rport ripaddr
5          5            xcode sendrecv g729b   16576 2000  192.0.2.3
5          6            xcode sendrecv g711u   18422 2000  192.0.2.4
Total number of active session(s) 1, and connection(s) 2
```

Step 5 **show dspfarm dsp active**

Displays active DSP information about the DSP farm service.

Example:

```
Device# show dspfarm dsp active
SLOT DSP VERSION  STATUS CHNL USE   TYPE    RSC_ID BRIDGE_ID PKTS_TXED PKTS_RXED
0    1   27.0.201 UP     1    USED  xcode   1      0x9         5         8
0    1   27.0.201 UP     1    USED  xcode   1      0x8         2558      17
Total number of DSPFARM DSP channel(s) 1
```



CHAPTER 24

Transcoding

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

SCCP/CME based Transcoding

- Skinny Client Control Protocol (SCCP) protocol is used for controlling Digital Signaling Processor (DSP) resources used for transcoding.
- Transcoding resources (DSP Farm) and CUBE can be on different platforms.
- SCCP client (For example, SCCP, Cisco Call Manager [CCM]) configuration and SCCP server (telephony service) configuration is required for CUBE configuration apart from DSP Farm profile configuration.
- DSPFARM registers with Cisco Unified Communications Manager Express (CUCME) 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
- Secure Real-time Transport Protocol (SRTP)-Real-time Transport Protocol (RTP) using transcoder requires **crypto pki trustpoint** configuration to establish the Transport Layer Security (TLS) connection with SCCP server.

Note *ISRG2 series devices support SCCP-based Transcoding only*.

LTI based Transcoding

- Internal API is used to access Digital Signaling Processor (DSP) resources for transcoding.
- Transcoding resources (DSPFARM) and CUBE need to be on the same platform.
- Only DSPFARM profile configuration is required. Skinny Client Control Protocol (SCCP) configuration and Cisco Unified Communications Manager Express (CUCME) configuration is not required.
- No TCP socket is opened and no registration is used.

- SSPFARM profile is associated to a new application type CUBE.
```
Device(config)# dspfarm profile 1 transcode
Device(config-dspfarm-profile)# associate application CUBE
```
- High density is flow through. All flow through features are supported on transcoding calls.
- **crypto pki trustpoint** configuration is not required for Secure Real-time Transport Protocol (SRTP)-Real-time Transport Protocol (RTP) calls.

Note *ASR 1000 Series devices support LTI-based Transcoding.*

- Configuring LTI-based Transcoding (ASR devices only), page 252
- Configuring SCCP-based Transcoding (ISR-G2 devices only), page 254
- Configuration Examples for Transcoding, page 256

Configuring LTI-based Transcoding (ASR devices only)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice-card** *voice-interface-slot-number*
4. **dspfarm tdm pooling**
5. **dspfarm services dspfarm**
6. **exit**
7. **dspfarm profile** *profile-id* **transcode**
8. **codec** *codec*
9. **maximum sessions** *sessions*
10. **associate application CUBE**
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
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	Purpose
Step 3	**voice-card** *voice-interface-slot-number* **Example:** `Device(config)# voice-card 1`	Configures a voice card and enters voice-card configuration mode.
Step 4	**dspfarm tdm pooling**	Enable voice card for DSP pooling using TDM bus.
Step 5	**dspfarm services dspfarm**	Enable voice-only dspfarm services on the Voice Card.
Step 6	**exit**	Exits the voice-card configuration mode.
Step 7	**dspfarm profile** *profile-id* **transcode** **Example:** `Device(config)# dspfarm profile 1 transcode`	Configures a Transcoding profile and enters DSP profile configuration mode. • The *profile-identifier* and *service type* uniquely identifies a profile. If the service type and profile-identifier pair is not unique, you are prompted to choose a different profile-identifier. • You can specify just the codec type, and the DSP uses the default codec parameter, such as independent mode, 32 kbps bit-rate, and 30 ms framesize.
Step 8	**codec** *codec* **Example:** `Device(config-dspfarm-profile)# codec ilbc`	The codec rate to be attempted for SCCP-controlled connections.
Step 9	**maximum sessions** *sessions*	Configures maximum number of sessions.
Step 10	**associate application CUBE** **Example:** Configures an application to the profile for LTI based transcoding.	
Step 11	**exit**	Exits interface configuration mode.

Configuring SCCP-based Transcoding (ISR-G2 devices only)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice-card** *voice-interface-slot-number*
4. **dspfarm tdm pooling**
5. **dspfarm services 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. **sccp ccm group** *group-id*
20. **associate ccm** *CCM-identifier* **priority** *priority*
21. **associate profile** *profile-identifier* **register** *Device-Name*
22. **exit**
23. **dspfarm profile** *profile-id* **transcode**
24. **codec** *codec*
25. **maximum sessions** *sessions*
26. **associate application sccp**
27. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device> configure terminal`	Enters global configuration mode.
Step 3	**voice-card** *voice-interface-slot-number* **Example:** `Device(config)# voice-card 1`	Configures a voice card and enters voice-card configuration mode.
Step 4	**dspfarm tdm pooling**	Enable voice card for DSP pooling using TDM bus.
Step 5	**dspfarm services dspfarm**	Enable voice-only dspfarm services on the Voice Card.
Step 6	**exit**	Exits the voice-card configuration mode.
Step 7	**telephony-service**	Configures Cisco Unified Communications Manager Express (CUCME) and enters telephony-service configuration mode.
Step 8	**sdspfarm units** *units*	Define maximum number of dspfarm units.
Step 9	**sdspfarm transcode sessions** *units*	Define maximum number of dspfarm transcode session.
Step 10	**sdspfarm tag** *value Device-Name* **Example:** `Device(config-telephony)# sdspfarm tag 1 CUBE-XCODE`	Configures a name for the transcoder.
Step 11	**max-ephones** *max-phones-to-be-supported*	Configures the maximum number of phones that are to be supported.
Step 12	**max-dn** *max-directorynumbers-to-be-supported*	Configures the maximum number of directories to be supported.
Step 13	**ip source-address** *CUBE-internal-ipv4-address* [**port** *port-number*] **Example:** `Device(config-telephony)# ip source-address 10.1.1.1 port 2000`	Defines an IP address and port number for the telephony service.
Step 14	**exit**	Exits the telephony-service configuration mode.
Step 15	**sccp local** *interface-type number*	Configures the local gateway related parameters values.
Step 16	**sccp ccm** *CUBE-internal-ipv4-address* **identifier** *identifier-number* **version** *version-number*	Configures call manager related parameter values.
Step 17	**sccp**	Enable Skinny Client Control Protocol.

	Command or Action	Purpose
Step 18	**sccp ccm group** *group-id* **Example:** `Device(config)#sccp ccm group 1`	Configures Call Manager Group and enters SCCP CCM configuration mode.
Step 19	**sccp ccm group** *group-id* **Example:** `Device(config-sccp-ccm)#sccp ccm group 1`	Configures Call Manager Group and enters SCCP CCM configuration mode.
Step 20	**associate ccm** *CCM-identifier* **priority** *priority* **Example:** `Device(config-sccp-ccm)# associate ccm 1 priority 1`	Configures Call Manager Group and enters SCCP CCM configuration mode.
Step 21	**associate profile** *profile-identifier* **register** *Device-Name* **Example:** `Device(config-sccp-ccm)# associate profile 1 register CUBE-XCODE`	Specifies the device name that needs to register with the CUCME.
Step 22	**exit**	Exits SCCP CCM configuration mode.
Step 23	**dspfarm profile** *profile-id* **transcode** **Example:** `Device(config)# dspfarm profile 1 transcode`	Configures a Transcoding profile and enters DSP profile configuration mode.
Step 24	**codec** *codec* **Example:** `Device(config-dspfarm-profile)# codec ilbc`	The codec rate to be attempted for SCCP-controlled connections.
Step 25	**maximum sessions** *sessions*	Configures maximum number of sessions.
Step 26	**associate application sccp** **Example:** `Device(config-dspfarm-profile)# associate application sccp`	Configures an application to the profile for SCCP-based transcoding.
Step 27	**exit**	Exits the telephony-service configuration mode.

Configuration Examples for Transcoding

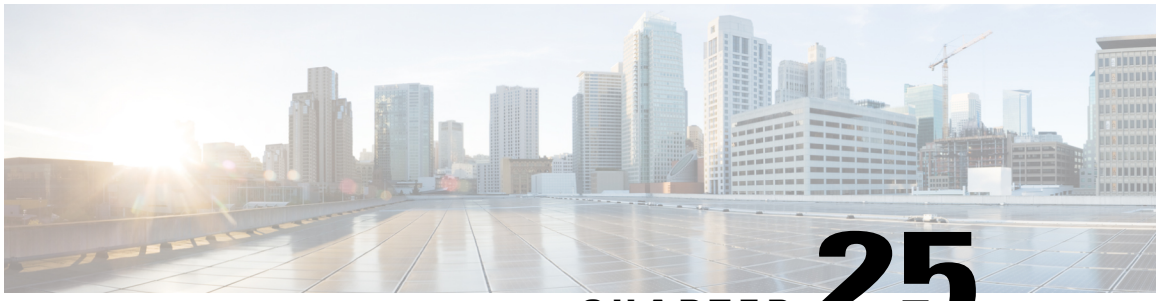
Example: SCCP-based Transcoding

```
! 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
```

Example: 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 g729r8
Device(config-dspfarm-profile)# maximum sessions 10

Device(config-dspfarm-profile)# associate application CUBE

Device(config-dspfarm-profile)# exit

! Starting Service Engine
Device(config)# interface ServiceEngine0/1/0
Device(config-if)# no shutdown
Device(config-if)# exit
```



CHAPTER 25

Transrating

Transrating is a process of configuring a different packetization for a voice codec. For example, transrating G.729 20ms to G.729 30ms.

Voice Packetization

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 13: Codec and Bandwidth Information, on page 50 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 35: Packetization for different Codecs

Supported Codecs	Packetization (ms)
G.711 a-law 64 Kbps	10, 20, 30
G.711 law 64 Kbps	10, 20, 30
G.723 5.3/6/3 Kbps	30, 60
G.729, G.729A, G.729B, G.729AB 8 Kbps	10, 20, 30, 40, 50, 60
G.722—64 Kbps	10, 20, 30

Configuring Transrating for a Codec

SUMMARY STEPS

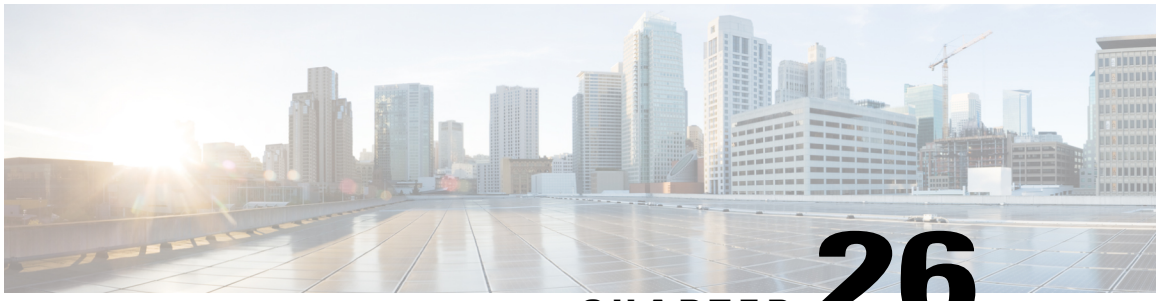
1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **voip**
4. **codec** *codec-name* **bytes** *voice-payload-size* [**fixed-bytes**]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
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** *number* **voip** **Example:** `Device(config)# dial-peer voice 1 voip`	Enters dial peer configuration mode for the specified VoIP dial peer.
Step 4	**codec** *codec-name* **bytes** *voice-payload-size* [**fixed-bytes**] **Example:** `Device(config-dial-peer)# codec g729r8 bytes 30 fixed-byte`	Configures a different packetizations for a voice codec.
Step 5	**end** **Example:** `Device(config-dial-peer)# end`	Exits to privileged EXEC mode.

PART V

Video



CHAPTER 26

Video Suppression

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

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 36: Feature Information for Video Suppression

Feature Name	Releases	Feature Information
Support for Video Suppression	Cisco IOS 15.6(2)T	This feature allows pass-through of only audio and application (for T.38 Fax) media types and drops all other media types in SDP. The following commands are introduced: **audio forced**, **voice-class sip audio forced**

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

	Command or Action	Purpose
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
	`!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
	`! Applying audio forced globally`
	`Device(config)# voice service voip`
	`Device (config-voi-serv)# sip`
		Enables pass-through of only audio and image media types in SDP.

	Command or Action	Purpose
	`Device (config-voi-sip)# audio forced` `Device (config-voi-sip)# end`	
Step 4	**end**	Exits present configuration mode and enters privileged EXEC mode.

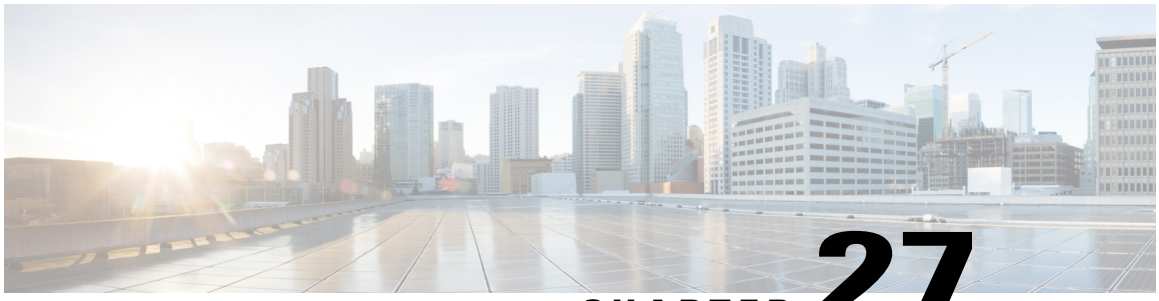
Troubleshooting Tips

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 VI

Media Recording



CHAPTER 27

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

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 37: Feature Information for Network-Based Recording

Feature Name	Releases	Feature Information
Audio-only Stream Forking of Video Call	Cisco IOS 15.4(3)M Cisco IOS XE 3.13S	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**.
Network-Based Recording of Video Calls Using CUBE	Cisco IOS 15.3(3)M Cisco IOS XE 3.10S	The Network-Based Recording of Video Calls using CUBE feature supports forking and recording of video calls.
Network-Based Recording of Audio Calls Using CUBE	Cisco IOS 15.2(1)T Cisco IOS XE 3.8S	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**.

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.
- Media mixing on forked streams is not supported
- Digital Signal Processing (DSP) resources are not supported on forked legs

Restrictions for Video Recording

- If the main call has multiple video streams (m-lines), the video streams other than the first video m-line are not forked.
- Application media streams of the primary call are not forked to the recording server.
- Forking is not supported if the anchor leg or recording server is on IPv6.
- High availability is not supported on forked video calls.

Information About Network-Based Recording Using CUBE

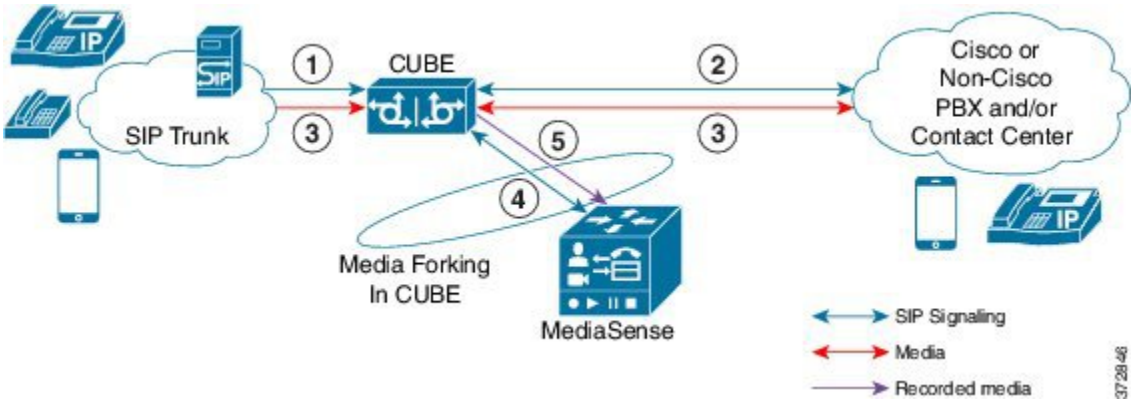
Deployment Scenarios for CUBE-based Recording

CUBE as a recording client has the following functions:

- Acts as a SIP user agent and sets up a recording session (SIP dialog) with the recording server.
- Acts as the source of the recorded media and forwards the recorded media to the recording server.
- Sends information to a server that helps the recording server associate the call with media streams and identifies the participants of the call. This information sent to the recording server is called metadata.

Given below is a typical deployment scenario of a CUBE-based recording solution. The information flow is described below:

Figure 23: Deployment Scenario for CUBE-based Recording Solution



1. Incoming call from SIP trunk.
2. Outbound call to a Contact Centre
3. Media between endpoints flowthrough CUBE
4. CUBE sets up a new SIP session with MediaSense based on policy.
5. CUBE forks RTP media to MediaSense. For an audio call, audio is forked. For a video call, both audio and video are .forked. For an audio-only configuration in a audio-video call, only audio is forked. There will be two or four m-lines to the recording server, based on the type of recording

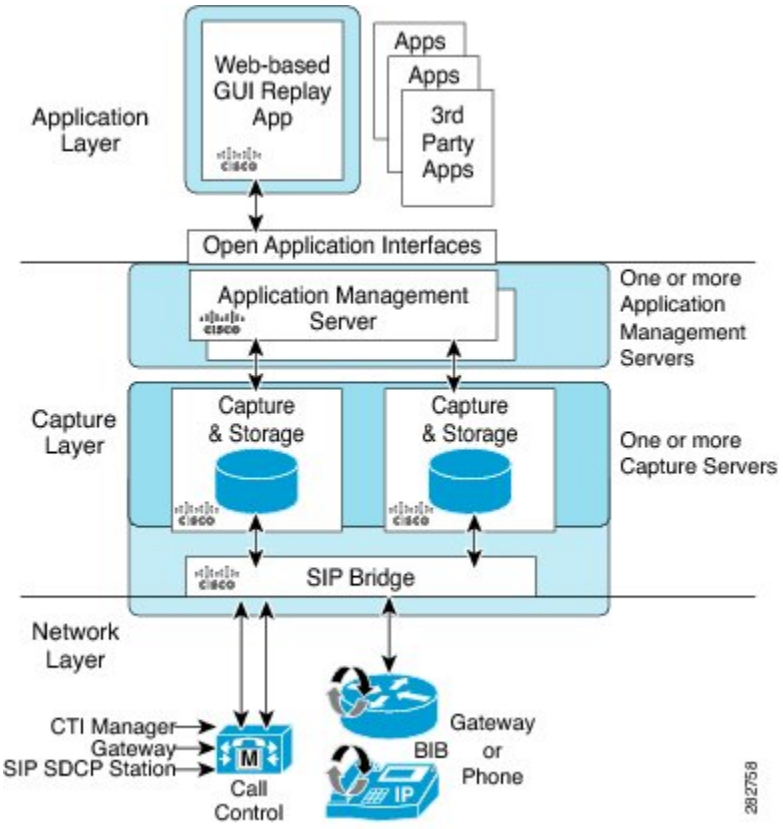
The metadata carried in the SIP session between the recording client and the recording server is to:

- Carry the communication session data that describes the call.
- Send the metadata to the recording server. The recording server uses the metadata to associate communication sessions involving two or more participants with media streams.

The call leg that is created between the recording client and the recording server is known as the recording session.

Open Recording Architecture

The Open Recording Architecture (ORA) comprises of elements, such as application management server and SIP bridge, to support IP-based recording. The ORA IP enables recording by solving topology issues, which accelerates the adoption of Cisco unified communication solutions.



Following are the three layers of the ORA architecture:

Network Layer

The ORA network layer is comprises 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.

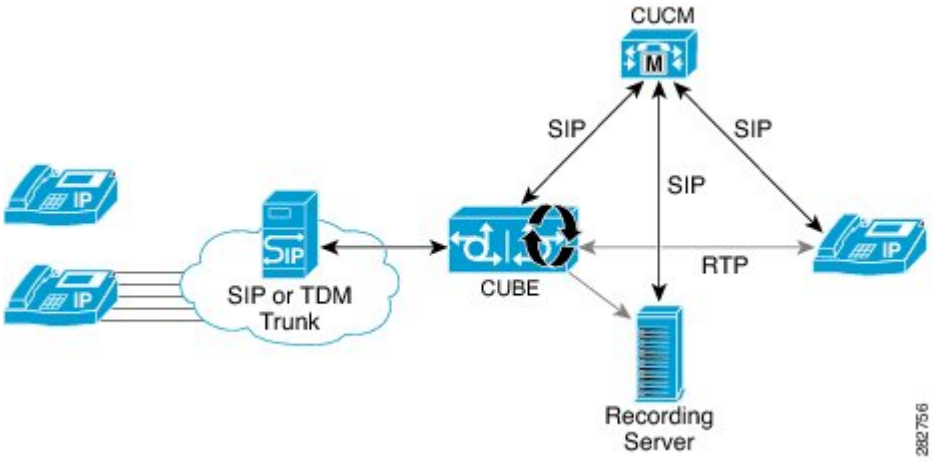
- Custom enterprise integration.
- Enterprise-wide policy management.

Media Forking Topologies

The following topologies support media forking:

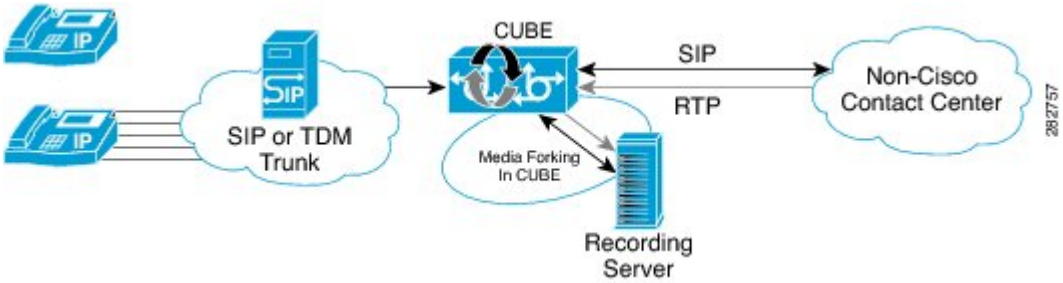
Media Forking with Cisco UCM

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

	Command or Action	Purpose
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	**media profile recorder** *profile-tag* **Example:** `Device(config)# media profile recorder 100`	Configures the media profile recorder and enters media profile configuration mode.
Step 4	**media-type audio** **Example:** `Device(cfg-mediaprofile)# media-type audio`	(Optional) 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.
Step 5	**media-recording** *dial-peer-tag* [*dial-peer-tag2...dial-peer-tag5*] **Example:** `Device(cfg-mediaprofile)# media-recording 8000 8001 8002`	Configures the dial-peers that need to be configured **Note** You can specify a maximum of five dial-peer tags.
Step 6	**exit** **Example:** `Device(cfg-mediaprofile)# exit`	Exits media profile configuration mode.
Step 7	**media class** *tag* **Example:** `Device(config)# media class 100`	Configures a media class and enters media class configuration mode.
Step 8	**recorder profile** *tag* **Example:** `Device(cfg-mediaclass)# recorder profile 100`	Configures the media profile recorder.

	Command or Action	Purpose
Step 9	**exit** **Example:** `Device(cfg-mediaclass)# exit`	Exits media class configuration mode.
Step 10	**dial-peer voice** *dummy-recorder-dial-peer-tag* **voip** **Example:** `Device(config)# dial-peer voice 8000 voip`	Configures a recorder dial peer and enters dial peer voice configuration mode.
Step 11	**media-class** *tag* **Example:** `Device(config-dial-peer)# media-class 100`	Configures media class on a dial peer.
Step 12	**destination-pattern** [+] *string* [**T**] **Example:** `Device(config-dial-peer)# destination-pattern 595959`	Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer.
Step 13	**session protocol sipv2** **Example:** `Device(config-dial-peer)# session protocol sipv2`	Configures the VoIP dial peer to use Session Initiation Protocol (SIP).
Step 14	**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 15	**session transport tcp** **Example:** `Device(config-dial-peer)# session transport tcp`	Configures a VoIP dial peer to use Transmission Control Protocol (TCP).
Step 16	**end** **Example:** `Device(config-dial-peer)# end`	Returns to privileged EXEC mode.

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

	Command or Action	Purpose
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	**media class** *tag* **Example:** `Device(config)# media class 100`	Configures the media class and enters media class configuration mode.
Step 4	**recorder parameter** **Example:** `Device(cfg-mediaclass)# recorder parameter`	Enters media class recorder parameter configuration mode to enable you to configure recorder-specific parameters.

	Command or Action	Purpose
Step 5	**media-type audio** **Example:** `Device(cfg-mediaprofile)# media-type audio`	(Optional) Configures recording of audio only in a call with both audio and video. **Note** If this configuration is not done, both audio and video are recorded.
Step 6	**media-recording** *dial-peer-tag* **Example:** `Device(cfg-mediaclass-recorder)# media-recording 8000, 8001, 8002`	Configures voice-class recording parameters. **Note** You can specify a maximum of five dial-peer tags.
Step 7	**exit** **Example:** `Device(cfg-mediaclass-recorder)# exit`	Exits media class recorder parameter configuration mode.
Step 8	**exit** **Example:** `Device(cfg-mediaclass)# exit`	Exits media class configuration mode.
Step 9	**dial-peer voice** *dummy-recorder-dial-peer-tag* **voip** **Example:** `Device(config)# dial-peer voice 8000 voip`	Configures a recorder dial peer and enters dial peer voice configuration mode.
Step 10	**media-class** *tag* **Example:** `Device(config-dial-peer)# media-class 100`	Configures media class on a dial peer.
Step 11	**destination-pattern** [+] *string* [**T**] **Example:** `Device(config-dial-peer)# destination-pattern 595959`	Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer. Keywords and arguments are as follows:
Step 12	**session protocol sipv2** **Example:** `Device(config-dial-peer)# session protocol sipv2`	Configures the VoIP dial peer to use Session Initiation Protocol (SIP).
Step 13	**session target ipv4:**[*recording-server-destination-address* | *recording-server-dns*]	Specifies a network-specific address for a dial peer. Keyword and argument are as follows:

	Command or Action	Purpose
	Example: `Device(config-dial-peer)# session target ipv4:10.42.29.7`	• **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 ccsip messages** (for audio calls)
11. **debug ccsip messages** (for video calls)
12. **debug ccsip messages** (for audio-only recording in a call with both audio and video)
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)

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

                                                Ports       Ports       Ports
Media-Address Range                             Available   Reserved    In-use

Default Address-Range                           8091        101         8

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

5     6          5          16392    58368  10.104.45.191                           10.104.105.232

6     7          5          16394    53828  10.104.45.191                           10.104.105.232

7     8          5          16396    39318  10.104.45.191                           10.104.105.232

8     9          5          16398    41114  10.104.45.191                           10.104.105.232

Found 8 active RTP connections
```

Step 3 **show voip recmsp session**
Displays active recording Media Service Provider (MSP) session information internal to CUBE.

Example:

```
Device# show voip recmsp session

RECMSP active sessions:
MSP Call-ID             AnchorLeg Call-ID        ForkedLeg Call-ID
143                     141                              145
Found 1 active sessions
```

Step 4 **show voip recmsp session detail call-id** *call-id*
Displays detailed information about the recording MSP Call ID.

Example:

```
Device# show voip recmsp session detail call-id 145
RECMSP active sessions:
Detailed Information
========================
Recording MSP Leg Details:
Call ID: 143
GUID : 7C5946D38ECD

AnchorLeg Details:
Call ID: 141
Forking Stream type: voice-nearend
Participant: 708090

Non-anchor Leg Details:
Call ID: 140
Forking Stream type: voice-farend
Participant: 10000

Forked Leg Details:
Call ID: 145
Near End Stream CallID 145
Stream State ACTIVE
Far End stream CallID 146
Stream State ACTIVE
Found 1 active sessions
Device# show voip recmsp session detail call-id 5

RECMSP active sessions:
Detailed Information
========================
Recording MSP Leg Details:
Call ID: 5
GUID : 1E01B6000000

AnchorLeg Details:
Call ID: 1
Forking Stream type: voice-nearend
Forking Stream type: video-nearend
Participant: 1777

Non-anchor Leg Details:
Call ID: 2
Forking Stream type: voice-farend
Forking Stream type: video-farend
Participant: 1888

Forked Leg Details:
Call ID: 6
Voice Near End Stream CallID 6
Stream State ACTIVE
Voice Far End stream CallID 7
Stream State ACTIVE
Video Near End stream CallID 8
Stream State ACTIVE
Video Far End stream CallID 9
Stream State ACTIVE
Found 1 active sessions
```

Output Field	Description
Stream State	Displays the state of the call. This can be ACTIVE or HOLD.

Output Field	Description
Msp Call-Id	Displays an internal Media service provider call ID and forking related statistics for an active forked call.
Anchor Leg Call-id	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.
Non-Anchor Call-id	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.
Forked Call-id	This forking leg call-id will show near-end and far-end stream call-id details with state of the Stream . Displays an internal foked leg ID. The output displays near-end and far-end details of a stream.

Step 5 **show voip rtp forking**

Displays RTP media-forking connections.

Example:

```
Device# show voip rtp forking
VoIP RTP active forks :
 Fork 1
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 1
     remote ip 10.42.29.7,  remote port 38526,  local port 18648
       codec g711ulaw,  logical ssrc 0x53
       packets sent 29687,  packets received 0
   stream type voice+dtmf-nearend (4): count 0
   stream type voice-farend (5): count 1
     remote ip 10.42.29.7,  remote port 50482,  local port 17780
       codec g711ulaw,  logical ssrc 0x55
       packets sent 29686,  packets received 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count
```

Output Field	Description
remote ip 10.42.29.7, remote port 38526, local port 18648	Recording server IP, recording server port, and local CUBE device port where data for stream 1 was first sent from.
remote ip 10.42.29.7, remote port 50482, local port 17780	Recording server IP, recording server port, and local CUBE device port where data for stream 2 was first sent from.
packets sent 29686	Number of packets sent to the recorder

Output Field	Description
codec g711ulaw	Codec negotiated for the recording leg.

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:3, User Agent Client:2, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID                : 99EA5118-506211E0-80C6E01B-4C27AA62@10.42.30.10
   State of the call       : STATE_ACTIVE (7)
   Substate of the call    : SUBSTATE_NONE (0)
   Calling Number          : 10000
   Called Number           : 708090
   Bit Flags               : 0xC04018 0x10000100 0x80
   CC Call ID              : 141
   Source IP Address (Sig ): 10.42.30.10
   Destn SIP Req Addr:Port : [10.42.30.5]:5060
   Destn SIP Resp Addr:Port: [10.42.30.5]:5060
   Destination Name        : 10.42.30.5
   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           : 141
     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: [10.42.30.10]:19256
     Media Dest IP Addr:Port  : [10.42.30.189]:26184
Options-Ping    ENABLED:NO    ACTIVE:NO
Call 2
SIP Call ID                 : 9A6D8922-506211E0-80CEE01B-4C27AA62@10.42.30.10
   State of the call        : STATE_ACTIVE (7)
   Substate of the call     : SUBSTATE_NONE (0)
   Calling Number           :
   Called Number            : 595959                                    Recoding server number
   Bit Flags                : 0xC04018 0x10800100 0x80
   CC Call ID               : 145
   Source IP Address (Sig ): 10.42.30.10
   Destn SIP Req Addr:Port : [10.42.29.7]:5060
   Destn SIP Resp Addr:Port: [10.42.29.7]:5060
   Destination Name         : 10.42.29.7
   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           : 145
     Stream Type              : voice-nearend (3)
     Stream Media Addr Type   : 1
     Negotiated Codec         : g711ulaw (160 bytes)
     Codec Payload Type       : 0
     Negotiated Dtmf-relay    : inband-voice
     Dtmf-relay Payload Type  : 0
     QoS ID                   : -1
     Local QoS Strength       : BestEffort
     Negotiated QoS Strength  : BestEffort
     Negotiated QoS Direction : None
     Local QoS Status         : None
     Media Source IP Addr:Port: [10.42.30.10]:18648
     Media Dest IP Addr:Port  : [10.42.29.7]:38526
   Media Stream 2
     State of the stream      : STREAM_ACTIVE
     Stream Call ID           : 146
     Stream Type              : voice-farend (5)
     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: [10.42.30.10]:17780
     Media Dest IP Addr:Port  : [10.42.29.7]:50482
Options-Ping    ENABLED:NO    ACTIVE:NO
   Number of SIP User Agent Client(UAC) calls: 2
SIP UAS CALL INFO
Call 1
SIP Call ID                 : 7CF44DF3-506611E0-8ED2B9D4-CA68C314@10.42.30.32
   State of the call        : STATE_ACTIVE (7)
   Substate of the call     : SUBSTATE_NONE (0)
   Calling Number           : 10000
   Called Number            : 708090
   Bit Flags                : 0x8C4401C 0x10000100 0x4
   CC Call ID               : 140
   Source IP Address (Sig ): 10.42.30.10
   Destn SIP Req Addr:Port : [10.42.30.32]:5060
   Destn SIP Resp Addr:Port: [10.42.30.32]:52757
```

```
   Destination Name        : 10.42.30.32
   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           : 140
     Stream Type              : voice+dtmf (0)
     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: [10.42.30.10]:18792
     Media Dest IP Addr:Port  : [10.42.30.32]:18638
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:0/0ms lost: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:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off
Transcoded: No
...
0    : 6 87425990ms.1 (*12:23:54.643 IST Wed Jul 17 2013) +680 pid:1234 Originate 1234 active
 dur 00:00:46 tx:10398/3732871 rx:0/0 dscp:0 media:0 audio tos:0xB8 video tos:0x0
 IP 10.104.105.232:39318 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off
Transcoded: No
...
```

Step 10 **debug ccsip messages** (for audio calls)

```
Sent:
INVITE sip:22222@10.42.29.7:5060 SIP/2.0
Via: SIP/2.0/TCP 10.42.30.10:5060;branch=z9hG4bKB622CF
X-Cisco-Recording-Participant: sip:708090@10.42.30.5;media-index="0"
X-Cisco-Recording-Participant: sip:10000@10.42.30.32;media-index="1"
From: <sip:10.42.30.10>;tag=5096700-1E1A
To: <sip:595959@10.42.29.7>
Date: Fri, 18 Mar 2011 07:01:50 GMT
Call-ID: 6E6CF813-506411E0-80EAE01B-4C27AA62@10.42.30.10
Supported: 100rel,timer,resource-priority,replaces,sdp-anat
Min-SE:  1800
Cisco-Guid: 1334370502-1348997600-2396699092-3395863316
```

```
User-Agent: Cisco-SIPGateway/IOS-15.2(0.0.2)PIA16
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1300431710
Contact: <sip:10.42.30.10:5060;transport=tcp>
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 449
v=0
o=CiscoSystemsSIP-GW-UserAgent 3021 3526 IN IP4 10.42.30.10
s=SIP Call
c=IN IP4 10.42.30.10
t=0 0
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:23232323@10.104.46.201>;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:23232323@10.104.46.201:5060;transport=tcp>
Content-Type: application/sdp
Allow: INVITE, BYE, CANCEL, ACK, NOTIFY, INFO, UPDATE
Server: Cisco-ORA/8.5
v=0
o=CiscoORA 2187 1 IN IP4 10.104.46.201
s=SIP Call
c=IN IP4 10.104.46.201
t=0 0
m=audio 54100 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=recvonly
m=audio 39674 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=recvonly

Sent:
ACK sip:23232323@10.104.46.201:5060;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:23232323@10.104.46.201>;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
```

Output Field	Description
INVITE sip:22222@10.42.29.7:5060 SIP/2.0	22222 is the destination pattern or the number of recording server and is configured under the recorder dial peer.
X-Cisco-Recording-Participant: sip:708090@10.42.30.5;media-index="0"	Cisco proprietary header with originating and terminating participant number and IP address used to communicate to the recording server
Cisco-Guid: 1334370502-1348997600-2396699092-3395863316	GUID is the same for the primary call and forked call .
m=audio 24544 RTP/AVP 0 101 19	First m-line of participant with payload type and codec information .
m=audio 31166 RTP/AVP 0 101 19	Second m- line of another participant with codec info and payload type.
a=sendonly	CUBE is always in send only mode towards Recording server.
a=recvonly	Recording server is in receive mode only.

Step 11 **debug ccsip messages** (for video calls)

```
Sent: INVITE sip:575757@9.45.38.39:7686 SIP/2.0

.
.
Via: SIP/2.0/UDP 9.41.36.41:5060;branch=z9hG4bK2CC2408
X-Cisco-Recording-Participant: sip:1777@10.104.45.207;media-
index="0 2“
X-Cisco-Recording-Participant: sip:1888@10.104.45.207;media-   index="1 3“
.
.
Cisco-Guid: 0884935168-0000065536-0000000401-3475859466
.
.
v=0
.
.
.
m=audio 17232 RTP/AVP 0 19
.
.
a=sendonly
m=audio 17234 RTP/AVP 0 19
.
.
a=sendonly

m=video 17236 RTP/AVP 126
.
.
```

```
.


a=fmtp:126 profile-level-id=42801E;packetization-mode=1
a=sendonly
m=video 17238 RTP/AVP 126
.
.

.
a=fmtp:126 profile-level-id=42801E;packetization-mode=1
a=sendonly
```

Output Field	Description
Sent: INVITE sip:575757@9.45.38.39:7686 SIP/2.0	22222 is the destination pattern or the number of recording server and is configured under the recorder dial peer.
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 proprietary header with originating and terminating participant number and IP address used to communicate to the recording server
Cisco-Guid: 0884935168-0000065536-0000000401-3475859466	GUID is the same for the primary call and forked call .
m=audio 17232 RTP/AVP 0 19	First m-line of participant with payload type and audio codec.
m=audio 17234 RTP/AVP 0 19	Second m-line of another participant with payload type and audio codec.
m=video 17236 RTP/AVP 126	Third m-line of participant with video payload type and codec info .
m=video 17238 RTP/AVP 126	Fourth m-line of another participant with video payload type and codec info .
a=sendonly	CUBE is always in send only mode towards Recording server.

```
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
```

Output Field	Description
m=audio 1592 RTP/AVP 0	First m-line of recording server after it started listening.
m=audio 1594 RTP/AVP 0	Second m-line of recording server after it started listening.
m=video 1596 RTP/AVP 126	Third m-line of recording server after it started listening.
m=video 1598 RTP/AVP 126	Fourth m-line of recording server after it started listening.
a=recvonly	Recording server in receive only mode.

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:4321@9.45.38.39;media-index="0"
X-Cisco-Recording-Participant: sip:1111000010@9.45.38.39;media-index="1"
From: <sip:9.41.36.15>;tag=A2C74-5D9
To: <sip:54321@9.45.38.39>.....
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 337

v=0
o=CiscoSystemsSIP-GW-UserAgent 9849 5909 IN IP4 9.41.36.15
s=SIP Call
c=IN IP4 9.41.36.15
t=0 0
```

```
m=audio 16392 RTP/AVP 0 19
c=IN IP4 9.41.36.15
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
a=ptime:20
a=sendonly
m=audio 16394 RTP/AVP 0 19
c=IN IP4 9.41.36.15
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
a=ptime:20
a=sendonly
```

Response from CUBE has inactive video m-lines.

```
Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.41.36.15:5060;branch=z9hG4bK2216B
…..
v=0
…
m=audio 36600 RTP/AVP 0
c=IN IP4 9.45.38.39
a=rtpmap:0 PCMU/8000
a=ptime:20
a=recvonly
m=audio 36602 RTP/AVP 0
c=IN IP4 9.45.38.39
a=rtpmap:0 PCMU/8000
a=ptime:20
a=recvonly
m=video 0 RTP/AVP 98
c=IN IP4 9.45.38.39
b=TIAS:1500000
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=420015
a=inactive
m=video 0 RTP/AVP 98
c=IN IP4 9.45.38.39
b=TIAS:1500000
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=420015
a=inactive
```

Step 13 Enter one of the following:

- **debug ccsip all**
- **debug voip recmsp all**
- **debug voip ccapi all**
- **debug voip fpi all** (for ASR devices only)

Displays detailed debug messages.

For Audio:
Media forking initialized:

```
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_trigger_media_forking: MF: Recv Ack..
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_trigger_media_forking: MF: Recv Ack & it's
Anchor leg. Start MF.
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_preprocess_event: MF:
initial-call. State = 1 & posting the event E_IPIP_MEDIA_FORKING_CALLSETUP_IND
```

Media forking started:

```
*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/sipSPIUisValidCcb:
*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..
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Function/sipSPIGetMainStream:
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Function/sipSPIGetMainStream:
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_precondition: MF: Can be
 started with current config.
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_BuildMediaRecParticipant:
 MF: Populate rec parti header from this leg.
```

Forking header populated:

```
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_get_recording_participant_header: MF: X-Cisco
 header is RPID..
```

Media forking setup record session is successful:

```
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_get_recording_participant_header: MF: Building
 SIP URL..
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_get_recording_participant_header: MF: Sipuser
 = 98459845
*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:
 direction type =3 3
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_BuildMediaRecStream: MF:
 callid 103 set to nearend..
*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_BuildMediaRecStream: MF:
 First element..
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_BuildMediaRecParticipant:
 MF: First element..
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_BuildMediaRecParticipant:
 MF: Populate rec parti header from peer leg.
*Jun 15 10:37:55.404: //104/3E7E90AE8006/SIP/Info/ccsip_get_recording_participant_header: MF: X-Cisco
 header is RPID..
*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: //-1/xxxxxxxxxxxx/Event/recmsp_api_setup_session: Event: E_REC_SETUP_REQ anchor
 call ID:103, msp call ID:105 infunction recmsp_api_setup_session
*Jun 15 10:37:55.404: //-1/xxxxxxxxxxxx/Inout/recmsp_api_setup_session:  Exit with Success
*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 10:37:55.404: //103/3E7E90AE8006/SIP/Info/sipSPIMFChangeState: MF: Prev state = 1 & New state
 = 2
*Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Info/ccsip_gen_service_process_event: MF: 30 event
handled.
*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_call_setup_request: Set Protocol information
*Jun 15 10:37:55.406: //106/xxxxxxxxxxxx/CCAPI/cc_set_post_tagdata:
*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_ipip_media_forking_read_from_TDContainer:
MF: Data read from TD container..
*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_ipip_media_forking_forked_leg_config: MF:
MSP callid = 105
*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.
*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_iwf_handle_peer_event:
*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_iwf_map_ccapi_event_to_iwf_event: Event
Category: 1, Event Id: 179
*Jun 15 10:37:55.406: //106/000000000000/SIP/Info/ccsip_iwf_process_event:
*Jun 15 10:37:55.406: //106/000000000000/SIP/Function/sipSPIUisValidCcb:
*Jun 15 10:37:55.406: //106/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_add_forking_stream: MF:
Forked stream added..
*Jun 15 10:37:55.406: //106/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_read_from_TDContainer:
MF: Data read from TD container..
*Jun 15 10:37:55.406: //106/3E7E90AE8006/SIP/Function/sipSPIGetFirstStream:
*Jun 15 10:37:55.406: //106/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_Display_TDContainerData:
 ** DISPLAY REC PART ***
```

```
*Jun 15 10:37:55.406: //106/3E7E90AE8006/SIP/Info/ccsip_ipip_media_forking_Display_TDContainerData:
 recorder tag = 5
```

For Video:

Media Forking Initialized:

```
*Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Info/notify/32768/ccsip_trigger_media_forking: MF:
 Recv Ack & it's Anchor leg. Start MF.
*Mar 19 16:40:01.784 IST:
//522/34BF0A000000/SIP/Info/info/32768/ccsip_ipip_media_forking_preprocess_event: MF: initial-call.
 State = 1 & posting the event E_IPIP_MEDIA_FORKING_CALLSETUP_IND
```

Media forking started:

```
 *Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Info/info/36864/ccsip_ipip_media_forking: MF:
Current State = 1, event =31
*Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Info/info/36864/ccsip_ipip_media_forking: MF: State
 & Event combination is cracked..
*Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Function/sipSPIGetMainStream:
 *Mar 19 16:40:01.784 IST: //522/34BF0A000000/SIP/Function/sipSPIGetMainStream:
*Mar 19 16:40:01.787 IST: //522/34BF0A000000/SIP/Info/info/34816/ccsip_ipip_media_forking_precondition:
 MF: Can be started with current config.
*Mar 19 16:40:01.787 IST: //-1/xxxxxxxxxxxx/Event/recmsp_api_create_session: Event:
E_REC_CREATE_SESSION anchor call ID:522, msp call ID:526
*Mar 19 16:40:01.787 IST: //-1/xxxxxxxxxxxx/Inout/recmsp_api_create_session:  Exit with Success
```

Recording participant for anchor leg:

```
//522/34BF0A000000/SIP/Info/verbose/32768/ccsip_ipip_media_forking_BuildMediaRecParticipant: MF:
Populate rec parti header from this leg.
*Mar 19 16:40:01.788 IST:
//522/34BF0A000000/SIP/Info/info/33792/ccsip_get_recording_participant_header: MF: X-Cisco header
is PAI..
```

Adding an audio stream:

```
*Mar 19 16:40:01.788 IST: //522/34BF0A000000/SIP/Function/sipSPIGetFirstStream:
*Mar 19 16:40:01.788 IST:
//522/34BF0A000000/SIP/Info/verbose/32768/ccsip_ipip_media_forking_BuildMediaRecStream: MF: Adding
a Audio stream..
*Mar 19 16:40:01.789 IST: //522/34BF0A000000/SIP/Function/voip_media_dir_to_cc_media_dir:
*Mar 19 16:40:01.789 IST:
//522/34BF0A000000/SIP/Info/info/32768/ccsip_ipip_media_forking_BuildAudioRecStream: MF: direction
type =3 3
*Mar 19 16:40:01.789 IST:
//522/34BF0A000000/SIP/Info/info/32768/ccsip_ipip_media_forking_BuildAudioRecStream: MF: callid 522
 set to nearend..
*Mar 19 16:40:01.789 IST:
//522/34BF0A000000/SIP/Info/info/32768/ccsip_ipip_media_forking_BuildAudioRecStream: MF: This rcstream
 has 522 callid
*Mar 19 16:40:01.789 IST:
//522/34BF0A000000/SIP/Info/verbose/32768/ccsip_ipip_media_forking_BuildAudioRecStream: MF: Setting
 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:
*Mar 19 16:40:01.789 IST:
//522/34BF0A000000/SIP/Info/verbose/32772/ccsip_ipip_media_forking_BuildMediaRecStream: MF: video_codec
 present,Continue with Video Forking..
```

For Video

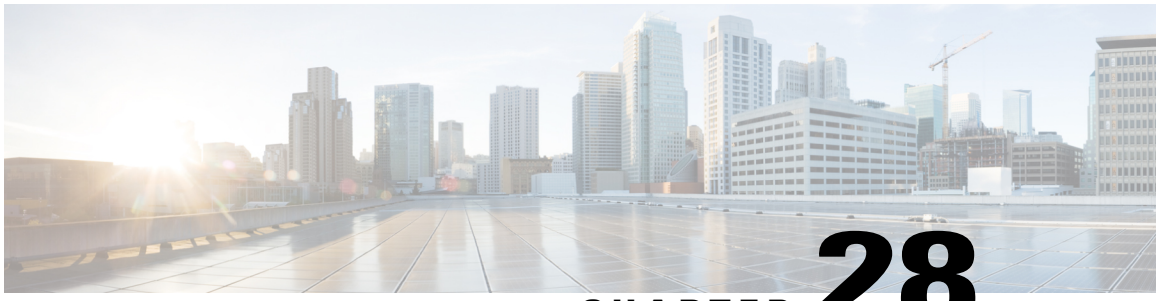
Additional References for Network-Based Recording

Related Documents

Related Topic	Document Title
MediaSense Installation and Administration Guide	Cisco MediaSense Installation and Administration Guide

Standards and RFCs

RFCs	Title
RFC 3984	*RTP Payload Format for H.264 Video*
RFC 5104	*Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)*
RFC 5168	*XML Schema for Media Control*



CHAPTER 28

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

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 Name	Releases	Feature Information
SIPREC (SIP Recording)	Cisco IOS 15.6(1)T Cisco IOS XE 3.17S	The SIPREC Recording feature supports recording of audio and video calls. Only audio and video media lines are forked. The following commands were modified: **recorder parameter** and **recorder profile**.

Prerequisites for SIPREC Recording

- Recorders must be reachable from CUBE
- SIPREC should be configured; else, CUBE will fall back to the existing Network-Based Recording implementation. For more information on Network-Based Recording, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-ntwk-based.html.
- CUBE supports the SIP Recording Metadata model format requirements specified in **draft-ietf-siprec-metadata-17**. Recorders must support metadata format of ver17 at a minimum
- CUBE should be in compliance with the Session Recording Protocols defined in **draft-ietf-siprec-protocol-16**. CUBE supports only the “siprec Option” Tag and the “src feature” tag among the various other extensions defined in the protocols draft; CUBE does not support the SDP extensions

Restrictions for SIPREC Recording

SIPREC-based recording is not supported for the following calls:

- Any media service parameter change via Re-INVITE or UPDATE from recording server is not supported. For example, hold-resume or any codec changes
- IPv6-to-IPv6 call recording
- IPv6-to-IPv4 call recording if the recording server is configured on the IPv6 call leg
- Calls that do not use Session Initiation Protocol (SIP). Must be a SIP-to-SIP call flow
- Flow-around calls
- Session Description Protocol (SDP) pass-through calls
- Real-time Transport Protocol (RTP) loopback calls
- High-density transcoder calls
- Secure Real-time Transport Protocol (SRTP) passthrough calls
- SRTP-RTP calls with forking for SRTP leg (forking is supported for the RTP leg)
- Multicast music on hold (MOH)
- Mid-call renegotiation and supplementary services like Hold/Resume, control pause, and so on are not supported on the recorder call leg

- Recording is not supported if CUBE is running a TCL IVR application
- Media mixing on forked streams is not supported
- Digital Signal Processing (DSP) resources are not supported on forked legs

Restrictions for Video Recording

- If the main call has multiple video streams (m-lines), the video streams other than the first video m-line are not forked
- Application media streams of the primary call are not forked to the recording server
- Forking is not supported if the anchor leg or recording server is on IPv6

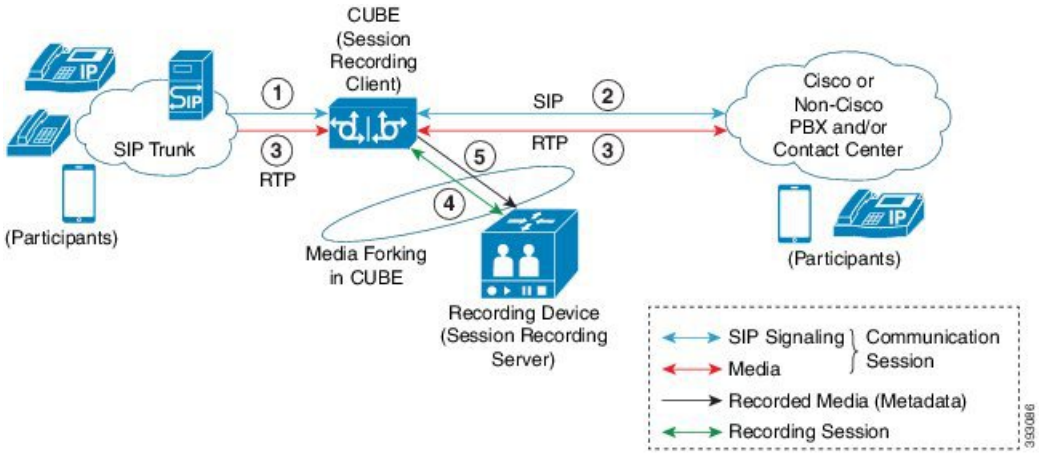
Information About SIPREC Recording Using CUBE

Deployment

- Participants—SIP UAs involved in the Communication Session. The UA can be any SIP element
- Communication Session (CS)—Session established between the endpoints
- Session Recording Client (SRC)—CUBE acts as the session recording client that triggers the recording session
- Session Recording Server (SRS)—A SIP User Agent (UA) which is a specialized media server and that acts as a sink for the recorded media and metadata
- Recording Session (RS) — SIP dialog established between CUBE (recording client) and the recording server
- Recording Metadata—Information on the CS and the associated media stream data sent from CUBE to RS

The following figure illustrates a third party recorder deployment with CUBE.

Figure 24: 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 represent the communication session forked from CUBE to the recording server to indicate forked media. The Session Initiation protocol (SIP) carries call signaling information along with the metadata information. Media streams from CUBE to recording server are unidirectional because only CUBE sends recorded data to recording server; the recording server does not send any media to CUBE.

Metadata has the following functions:

- Carry the communication session data (audio and video calls) that describes the call to the recording server
- Identifies the participants list
- Identifies the session and media association time

If there are any changes in the call sessions, for example, hold-resume, transfer and so on, these sessions are notified to the recording server through metadata.

SIPREC High Availability Support

High availability is supported for SIPREC recording using CUBE. All metadata elements will be checkpointed in a forked call when high-availability is configured. In the event of SSO, all the forked calls and media contexts are preserved on failover.

How to Configure SIPREC-Based Recording

Configuring SIPREC-Based Recording (with Media Profile Recorder)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **media profile recorder** *profile-tag*
4. (Optional) **media-type audio**
5. **media-recording** *dial-peer-tag* [*dial-peer-tag2...dial-peer-tag5*]
6. **exit**
7. **media class** *tag*
8. **recorder profile** *tag* **siprec**
9. **exit**
10. **dial-peer voice** *dummy-recorder-dial-peer-tag* **voip**
11. **media-class** *tag*
12. **destination-pattern** [+] *string* [**T**]
13. **session protocol sipv2**
14. **session target ipv4:**[*recording-server-destination-address* | *recording-server-dns*]
15. **session transport tcp**
16. **end**

DETAILED STEPS

	Command or Action	Purpose
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	**media profile recorder** *profile-tag* **Example:** `Device(config)# media profile recorder 100`	Configures the media profile recorder and enters media profile configuration mode.

	Command or Action	Purpose
Step 4	**media-type audio** **Example:** `Device(cfg-mediaprofile)# media-type audio`	(Optional) 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.
Step 5	**media-recording** *dial-peer-tag* [*dial-peer-tag2...dial-peer-tag5*] **Example:** `Device(cfg-mediaprofile)# media-recording 8000 8001 8002`	Configures the dial-peers that need to be configured **Note** You can specify a maximum of five dial-peer tags.
Step 6	**exit** **Example:** `Device(cfg-mediaprofile)# exit`	Exits media profile configuration mode.
Step 7	**media class** *tag* **Example:** `Device(config)# media class 100`	Configures a media class and enters media class configuration mode.
Step 8	**recorder profile** *tag* **siprec** **Example:** `Device(cfg-mediaclass)# recorder profile 201 siprec`	Configures the media profile SIPREC recorder.
Step 9	**exit** **Example:** `Device(cfg-mediaclass)# exit`	Exits media class configuration mode.
Step 10	**dial-peer voice** *dummy-recorder-dial-peer-tag* **voip** **Example:** `Device(config)# dial-peer voice 8000 voip`	Configures a recorder dial peer and enters dial peer voice configuration mode.
Step 11	**media-class** *tag* **Example:** `Device(config-dial-peer)# media-class 100`	Configures media class on a dial peer.
Step 12	**destination-pattern** [**+**] *string* [**T**] **Example:** `Device(config-dial-peer)# destination-pattern 595959`	Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer.

	Command or Action	Purpose
Step 13	**session protocol sipv2** **Example:** `Device(config-dial-peer)# session protocol sipv2`	Configures the VoIP dial peer to use Session Initiation Protocol (SIP).
Step 14	**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 15	**session transport tcp** **Example:** `Device(config-dial-peer)# session transport tcp`	Configures a VoIP dial peer to use Transmission Control Protocol (TCP).
Step 16	**end** **Example:** `Device(config-dial-peer)# end`	Returns to privileged EXEC mode.

Configuring SIPREC-Based Recording (without Media Profile Recorder)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **media class** *tag*
4. **recorder parametersiprec**
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

	Command or Action	Purpose
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	**media class** *tag* **Example:** `Device(config)# media class 100`	Configures the media class and enters media class configuration mode.
Step 4	**recorder parametersiprec** **Example:** `Device(cfg-mediaclass)# recorder parameter siprec`	Enables SIPREC recording.

	Command or Action	Purpose
Step 5	**media-type audio** **Example:** `Device(cfg-mediaprofile)# media-type audio`	(Optional) Configures recording of audio only in a call with both audio and video. **Note** If this configuration is not done, both audio and video are recorded.
Step 6	**media-recording** *dial-peer-tag* **Example:** `Device(cfg-mediaclass-recorder)# media-recording 8000, 8001, 8002`	Configures voice-class recording parameters. **Note** You can specify a maximum of five dial-peer tags.
Step 7	**exit** **Example:** `Device(cfg-mediaclass-recorder)# exit`	Exits media class recorder parameter configuration mode.
Step 8	**exit** **Example:** `Device(cfg-mediaclass)# exit`	Exits media class configuration mode.
Step 9	**dial-peer voice** *dummy-recorder-dial-peer-tag* **voip** **Example:** `Device(config)# dial-peer voice 8000 voip`	Configures a recorder dial peer and enters dial peer voice configuration mode.
Step 10	**media-class** *tag* **Example:** `Device(config-dial-peer)# media-class 100`	Configures media class on a dial peer.
Step 11	**destination-pattern** [+] *string* [**T**] **Example:** `Device(config-dial-peer)# destination-pattern 595959`	Specifies either the prefix or the full E.164 telephone number (depending on your dial plan) to be used for a dial peer. Keywords and arguments are as follows:
Step 12	**session protocol sipv2** **Example:** `Device(config-dial-peer)# session protocol sipv2`	Configures the VoIP dial peer to use Session Initiation Protocol (SIP).
Step 13	**session target ipv4:**[*recording-server-destination-address* | *recording-server-dns*]	Specifies a network-specific address for a dial peer. Keyword and argument are as follows:

	Command or Action	Purpose
	Example: `Device(config-dial-peer)# session target ipv4:10.42.29.7`	• **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.

Configuration Examples for SIPREC-based Recording

Example: Configuring SIPREC-based Recording with Media Profile Recorder

```
Router> enable
Router# configure terminal
Router(config)# media class 101
Router(cfg-mediaclass)# recorder profile 201 siprec
```

Example: Configuring SIPREC-based Recording without Media Profile Recorder

```
Router> enable
Router# configure terminal
Router(config)# media class 101
Router(cfg-mediaclass)# recorder parameter siprec
Router(cfg-mediaclass-recorder)# media-recording 403
```

Example of Metadata Variations with Different Mid-call Flows

Example: Complete SIP Recording Metadata information sent in INVITE or Re-INVITE

The following example provides all the elements involved in Recording Metadata XML body.

```
--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required
v=0
o=CiscoSystemsSIP-GW-UserAgent 509 7422 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
```

```
t=0 0
m=audio 16552 RTP/AVP 8 101
c=IN IP4 9.42.25.149
a=rtpmap:8 PCMA/8000
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:2
m=video 16556 RTP/AVP 119
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
--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="JaPQeP1CEeSA66sYHx7YVg==">
        <start-time>2015-05-19T09:42:06.911Z</start-time>
    </session>
    <participant participant_id="JaPQeP1CEeSA76sYHx7YVg==">
        <nameID aor="sip:808808@9.0.0.174">
            <name xml:lang="en">808808</name>
        </nameID>
    </participant>
    <participantsessionassoc participant_id="JaPQeP1CEeSA76sYHx7YVg=="
session_id="JaPQeP1CEeSA66sYHx7YVg==">
        <associate-time>2015-05-19T09:42:06.911Z</associate-time>
    </participantsessionassoc>
    <stream stream_id="JaPQeP1CEeSA8KsYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
        <label>1</label>
    </stream>
    <stream stream_id="JaPQeP1CEeSA8asYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
        <label>3</label>
    </stream>
    <participant participant_id="JaPQeP1CEeSA8qsYHx7YVg==">
        <nameID aor="sip:909909@9.0.0.174">
            <name xml:lang="en">909909</name>
        </nameID>
    </participant>
    <participantsessionassoc participant_id="JaPQeP1CEeSA8qsYHx7YVg=="
session_id="JaPQeP1CEeSA66sYHx7YVg==">
        <associate-time>2015-05-19T09:42:06.911Z</associate-time>
    </participantsessionassoc>
    <stream stream_id="JaPQeP1CEeSA86sYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
        <label>2</label>
    </stream>
    <stream stream_id="JaPQeP1CEeSA9KsYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">
        <label>4</label>
    </stream>
    <participantstreamassoc participant_id="JaPQeP1CEeSA76sYHx7YVg==">
        <send>JaPQeP1CEeSA8KsYHx7YVg==</send>
```

```
        <recv>JaPQeP1CEeSA86sYHx7YVg==</recv>
        <send>JaPQeP1CEeSA8asYHx7YVg==</send>
        <recv>JaPQeP1CEeSA9KsYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="JaPQeP1CEeSA8qsYHx7YVg==">
        <send>JaPQeP1CEeSA86sYHx7YVg==</send>
        <recv>JaPQeP1CEeSA8KsYHx7YVg==</recv>
        <send>JaPQeP1CEeSA9KsYHx7YVg==</send>
        <recv>JaPQeP1CEeSA8asYHx7YVg==</recv>
    </participantstreamassoc>
</recording>
—uniqueBoundary—
```

Output Field	Description
urn:ietf:params:xml:ns:recording:1	Defines the namespace URI for the elements—Uniform Resource Namespace (URN).
datamode>complete</datamode	<dataMode> is a recording element that indicates whether the XML document is a complete document or a partial update. If no <dataMode> element is present then the default value is "complete".
session session_id="JaPQeP1CEeSA66sYHx7YVg=="	Session ID which remains constant for the complete call leg.
<participant participant_id="JaPQeP1CEeSA76sYHx7YVg=="> <nameID aor="sip:808808@9.0.0.174">	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**.
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.
participantsessionassoc participant_id="JaPQeP1CEeSA76sYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg==">	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. ParticipantCS association class has the following attributes: • Associate-time—Time when the participant is associated to CS. • Disassociate-time—Time when the participant is disassociated from a CS. Each CS object is represented by one session element. Each session element has a unique 'session_id' attribute which helps to identify unique CS sessions.

Output Field	Description
participantsessionassoc participant_id="JaPQeP1CEeSA8qsYHx7YVg==" session_id="JaPQeP1CEeSA66sYHx7YVg=="	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 session of the second participant.
participantstreamassoc participant_id="JaPQeP1CEeSA76sYHx7YVg=="; participantstreamassoc participant_id="JaPQeP1CEeSA8qsYHx7YVg==";	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. ParticipantStream association class has the following attributes: • Associate-time—Time when the participant starts contributing for a media stream. • Disassociate-time—Time when the participant stops receiving a media stream.

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.

```
--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required

v=0
o=CiscoSystemsSIP-GW-UserAgent 2973 4879 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
t=0 0
m=audio 16464 RTP/AVP 0 101
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 16466 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 16468 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 16470 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

--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">
…
    <stream stream_id="jIBTUf1BEeSAdKsYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>1</label>
    </stream>
    <stream stream_id="jIBTUf1BEeSAdasYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>3</label>
    </stream>
…
    <stream stream_id="jIBTUf1BEeSAd6sYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>2</label>
    </stream>
    <stream stream_id="jIBTUf1BEeSAeKsYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>4</label>
    </stream>
    <participantstreamassoc participant_id="jIBTUf1BEeSAc6sYHx7YVg==">
        <send>jIBTUf1BEeSAdKsYHx7YVg==</send>
        <recv>jIBTUf1BEeSAd6sYHx7YVg==</recv>
        <send>jIBTUf1BEeSAdasYHx7YVg==</send>
        <recv>jIBTUf1BEeSAeKsYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="jIBTUf1BEeSAdqsYHx7YVg==">
        <send>jIBTUf1BEeSAd6sYHx7YVg==</send>
        <recv>jIBTUf1BEeSAdKsYHx7YVg==</recv>
        <send>jIBTUf1BEeSAeKsYHx7YVg==</send>
        <recv>jIBTUf1BEeSAdasYHx7YVg==</recv>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

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:

```
--uniqueBoundary
Content-Type: application/sdp

v=0
o=CiscoSystemsSIP-GW-UserAgent 2973 4880 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
t=0 0
m=audio 16464 RTP/AVP 0 101
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=inactive
a=label:1
m=audio 16466 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 16468 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=inactive
a=label:3
m=video 16470 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

--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">
…
    <stream stream_id="jIBTUf1BEeSAdKsYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>1</label>
    </stream>
    <stream stream_id="jIBTUf1BEeSAdasYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>3</label>
    </stream>
…
    <stream stream_id="jIBTUf1BEeSAd6sYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>2</label>
    </stream>
    <stream stream_id="jIBTUf1BEeSAeKsYHx7YVg==" session_id="jH+2kf1BEeSAb6sYHx7YVg==">
        <label>4</label>
    </stream>
    <participantstreamassoc participant_id="jIBTUf1BEeSAc6sYHx7YVg==">
        <recv>jIBTUf1BEeSAd6sYHx7YVg==</recv>
        <recv>jIBTUf1BEeSAeKsYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="jIBTUf1BEeSAdqsYHx7YVg==">
        <send>jIBTUf1BEeSAd6sYHx7YVg==</send>
        <send>jIBTUf1BEeSAeKsYHx7YVg==</send>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

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.

```
--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

--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">
…
    <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/B4f1BEeSAnKsYHx7YVg==" session_id="uV/B4f1BEeSAk6sYHx7YVg==">
        <label>4</label>
    </stream>
    <participantstreamassoc participant_id="uV/B4f1BEeSAl6sYHx7YVg==">
        <send>uV/B4f1BEeSAmKsYHx7YVg==</send>
        <recv>uV/B4f1BEeSAm6sYHx7YVg==</recv>
        <send>uV/B4f1BEeSAmasYHx7YVg==</send>
        <recv>uV/B4f1BEeSAnKsYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="uV/B4f1BEeSAmqsYHx7YVg==">
        <send>uV/B4f1BEeSAm6sYHx7YVg==</send>
        <recv>uV/B4f1BEeSAmKsYHx7YVg==</recv>
        <send>uV/B4f1BEeSAnKsYHx7YVg==</send>
        <recv>uV/B4f1BEeSAmasYHx7YVg==</recv>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

When the first participant puts the call on hold with inactive SDP attribute, there will be not any active streams in the metadata.

```
--uniqueBoundary
Content-Type: application/sdp

v=0
o=CiscoSystemsSIP-GW-UserAgent 7476 1348 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=inactive
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=inactive
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=inactive
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=inactive
a=label:4

--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">
…
    <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/B4f1BEeSAnKsYHx7YVg==" session_id="uV/B4f1BEeSAk6sYHx7YVg==">
        <label>4</label>
    </stream>
    <participantstreamassoc participant_id="uV/B4f1BEeSAl6sYHx7YVg==">
    </participantstreamassoc>
    <participantstreamassoc participant_id="uV/B4f1BEeSAmqsYHx7YVg==">
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

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.

```
--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required

v=0
o=CiscoSystemsSIP-GW-UserAgent 6360 4788 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
```

```
t=0 0
m=audio 16628 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 16630 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: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">
…
    <stream stream_id="evyS5/1CEeSBOKsYHx7YVg==" session_id="evv2v/1CEeSBM6sYHx7YVg==">
        <label>1</label>
    </stream>
…
    <stream stream_id="evyS5/1CEeSBOqsYHx7YVg==" session_id="evv2v/1CEeSBM6sYHx7YVg==">
        <label>2</label>
    </stream>
    <participantstreamassoc participant_id="evyS5/1CEeSBN6sYHx7YVg==">
        <send>evyS5/1CEeSBOKsYHx7YVg==</send>
        <recv>evyS5/1CEeSBOqsYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="evyS5/1CEeSBOasYHx7YVg==">
        <send>evyS5/1CEeSBOqsYHx7YVg==</send>
        <recv>evyS5/1CEeSBOKsYHx7YVg==</recv>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

After escalation, video streams get added into the participantStream association element in metadata for both the participants. There will be 4 streams in total.

```
--uniqueBoundary
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
a=sendonly
a=label:2
```

```
m=video 16636 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 16638 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

--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">
…
    <stream stream_id="evyS5/1CEeSBOKsYHx7YVg==" session_id="evv2v/1CEeSBM6sYHx7YVg==">
        <label>1</label>
    </stream>
    <stream stream_id="e5Zhtv1CEeSBPKsYHx7YVg==" session_id="evv2v/1CEeSBM6sYHx7YVg==">
        <label>3</label>
    </stream>
…
    <stream stream_id="e5Zhtv1CEeSBPasYHx7YVg==" session_id="evv2v/1CEeSBM6sYHx7YVg==">
        <label>4</label>
    </stream>
    <participantstreamassoc participant_id="evyS5/1CEeSBN6sYHx7YVg==">
        <send>evyS5/1CEeSBOKsYHx7YVg==</send>
        <recv>evyS5/1CEeSBOqsYHx7YVg==</recv>
        <send>e5Zhtv1CEeSBPKsYHx7YVg==</send>
        <recv>e5Zhtv1CEeSBPasYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="evyS5/1CEeSBOasYHx7YVg==">
        <send>evyS5/1CEeSBOqsYHx7YVg==</send>
        <recv>evyS5/1CEeSBOKsYHx7YVg==</recv>
        <send>e5Zhtv1CEeSBPasYHx7YVg==</send>
        <recv>e5Zhtv1CEeSBPKsYHx7YVg==</recv>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

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.

```
--uniqueBoundary
Content-Type: application/sdp
Content-Disposition: session;handling=required

v=0
o=CiscoSystemsSIP-GW-UserAgent 7616 8308 IN IP4 9.42.25.149
s=SIP Call
c=IN IP4 9.42.25.149
t=0 0
m=audio 16648 RTP/AVP 116 101
c=IN IP4 9.42.25.149
a=rtpmap:116 iLBC/8000
a=fmtp:116 mode=20
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
```

```
a=maxptime:20
a=sendonly
a=label:1
m=audio 16650 RTP/AVP 116 101
c=IN IP4 9.42.25.149
a=rtpmap:116 iLBC/8000
a=fmtp:116 mode=20
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=ptime:20
a=maxptime:20
a=sendonly
a=label:2
m=video 16652 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 16654 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

--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">
…
    <stream stream_id="j5OQdP1CEeSBSqsYHx7YVg==" session_id="j5L0TP1CEeSBRasYHx7YVg==">
        <label>1</label>
    </stream>
    <stream stream_id="j5OQdP1CEeSBS6sYHx7YVg==" session_id="j5L0TP1CEeSBRasYHx7YVg==">
        <label>3</label>
    </stream>
…
    <stream stream_id="j5OQdP1CEeSBTasYHx7YVg==" session_id="j5L0TP1CEeSBRasYHx7YVg==">
        <label>2</label>
    </stream>
    <stream stream_id="j5OQdP1CEeSBTqsYHx7YVg==" session_id="j5L0TP1CEeSBRasYHx7YVg==">
        <label>4</label>
    </stream>
    <participantstreamassoc participant_id="j5OQdP1CEeSBSasYHx7YVg==">
        <send>j5OQdP1CEeSBSqsYHx7YVg==</send>
        <recv>j5OQdP1CEeSBTasYHx7YVg==</recv>
        <send>j5OQdP1CEeSBS6sYHx7YVg==</send>
        <recv>j5OQdP1CEeSBTqsYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="j5OQdP1CEeSBTKsYHx7YVg==">
        <send>j5OQdP1CEeSBTasYHx7YVg==</send>
        <recv>j5OQdP1CEeSBSqsYHx7YVg==</recv>
        <send>j5OQdP1CEeSBTqsYHx7YVg==</send>
        <recv>j5OQdP1CEeSBS6sYHx7YVg==</recv>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

After de-escalation, video streams are removed from the metadata and only audio calls will be present in the participantStream association element.

```
--uniqueBoundary
Content-Type: application/sdp

v=0
o=CiscoSystemsSIP-GW-UserAgent 7616 8309 IN IP4 9.42.25.149
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

--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">
...
    <stream stream_id="j5OQdP1CEeSBSqsYHx7YVg==" session_id="j5L0TP1CEeSBRasYHx7YVg==">
        <label>1</label>
    </stream>
...
    <stream stream_id="j5OQdP1CEeSBTasYHx7YVg==" session_id="j5L0TP1CEeSBRasYHx7YVg==">
        <label>2</label>
    </stream>
    <participantstreamassoc participant_id="j5OQdP1CEeSBSasYHx7YVg==">
        <send>j5OQdP1CEeSBSqsYHx7YVg==</send>
        <recv>j5OQdP1CEeSBTasYHx7YVg==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="j5OQdP1CEeSBTKsYHx7YVg==">
        <send>j5OQdP1CEeSBTasYHx7YVg==</send>
        <recv>j5OQdP1CEeSBSqsYHx7YVg==</recv>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

Example of Metadata Variations with Different Transfer Flows

Example: Transfer of Re-INVITE/REFER Consume Scenario

In the case of Re-INVITE or REFER Consume transfer scenarios, CUBE receives re-INVITE with caller-id change. This re-INVITE will have the remote-party-ID details.

After transfer, participant A is disassociated from the call and participant C joins the call. This information is provided in the metadata sent to the recording server. Here, **7774442214** associates and **7774442212** disassociates from the call.

```
INVITE sip:7774442216@10.64.86.102:5060;transport=tcp SIP/2.0
From: <sip:7774442212@10.104.54.52>;tag=498652~97a89a01
To: <sip:7774442216@10.64.86.102>;tag=7C798-1441
...
...
Remote-Party-ID: <sip:7774442214@10.104.54.52>;party=calling;screen=yes;privacy=off
Contact: <sip:7774442214@10.104.54.52:5060;transport=tcp>
...
 <participant participant_id="vm+z2xM6EeWAIN4iOrLrag==">
        <nameID aor="sip:7774442214@10.104.54.52">
        </nameID>
    </participant>
    <participantsessionassoc participant_id="vm+z2xM6EeWAIN4iOrLrag=="
session_id="vACJ+xM6EeWAF94iOrLrag==">
        <associate-time>2015-06-16T08:44:32.869Z</associate-time>
    </participantsessionassoc>
...
 <participant participant_id="vACJ+xM6EeWAGN4iOrLrag==">
        <nameID aor="sip:7774442212@10.104.54.52">
        </nameID>
    </participant>
    <participantsessionassoc participant_id="vACJ+xM6EeWAGN4iOrLrag=="
session_id="vACJ+xM6EeWAGN4iOrLrag==">
        <disassociate-time>2015-06-16T08:44:32.869Z</disassociate-time>
</participantsessionassoc>
...
```

Example of Metadata Variations with Caller-ID UPDATE Flow

Example: Caller-ID UPDATE Request and Response Scenario

In case of Re-INVITE based transfer, any UPDATE request will contain caller-id changes. These changes are forwarded to the remote party and once CUBE receives a 200OK message, the remote-party-ID details are transferred.

The response of UPDATE request contains the associated caller-id changes. The CUBE forwards the response UPDATE information to the remote party with caller-id changes after the UPDATE request. From the metadata, you can see that the participants A and C disassociate from the call and participants B and D joins (associates) the call. Here, **7774442212** and **7774442216** disassociates from the call and **7774442214** and **7774442218** joins the call after the caller-id update.

```
UPDATE sip:7774442216@10.64.86.102:5060;transport=tcp SIP/2.0
From: <sip:7774442212@10.104.54.52>;tag=498652~97a89a01
To: <sip:7774442216@10.64.86.102>;tag=7C798-1441
...
...
Remote-Party-ID: <sip:7774442214@10.104.54.52>;party=calling;screen=yes;privacy=off
Contact: <sip:7774442214@10.104.54.52:5060;transport=tcp>

Response of UPDATE contains caller-id changes
...
SIP/2.0 200 OK
From: <sip:7774442212@10.64.86.102>;tag=7C78C-1E7C
To: <sip:7774442216@10.104.54.52>;tag=498653~97a89a01
...
Remote-Party-ID: <sip:7774442218@10.104.54.52>;party=called;screen=yes;privacy=off
Contact: <sip:7774442218@10.104.54.52:5060>
Content-Length: 0

...
```

```
    <participant participant_id="vm+z2xM6EeWAIN4iOrLrag==">
        <nameID aor="sip:7774442214@10.104.54.52">
        </nameID>
    </participant>
    <participantsessionassoc participant_id="vm+z2xM6EeWAIN4iOrLrag=="
session_id="vACJ+xM6EeWAF94iOrLrag==">
        <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+xM6EeWAF94iOrLrag==">
        <associate-time>2015-06-16T08:44:32.869Z</associate-time>
    </participantsessionassoc>
    <participant participant_id="vACJ+xM6EeWAGN4iOrLrag==">
        <nameID aor="sip:7774442212@10.104.54.52">
        </nameID>
    </participant>
    <participantsessionassoc participant_id="vACJ+xM6EeWAGN4iOrLrag=="
session_id="vACJ+xM6EeWAGN4iOrLrag==">
        <disassociate-time>2015-06-16T08:44:32.869Z</disassociate-time>
    </participantsessionassoc>
    <participant participant_id="vACJ+xM6EeWAGN4iOrLrag==">
        <nameID aor="sip:7774442216@10.104.54.52">
        </nameID>
    </participant>
    <participantsessionassoc participant_id="vACJ+xM6EeWAGN4iOrLrag=="
session_id="vACJ+xM6EeWAGN4iOrLrag==">
        <disassociate-time>2015-06-16T08:44:32.869Z</disassociate-time>
    </participantsessionassoc>
...
```

Example of Metadata Variations with Call Disconnect

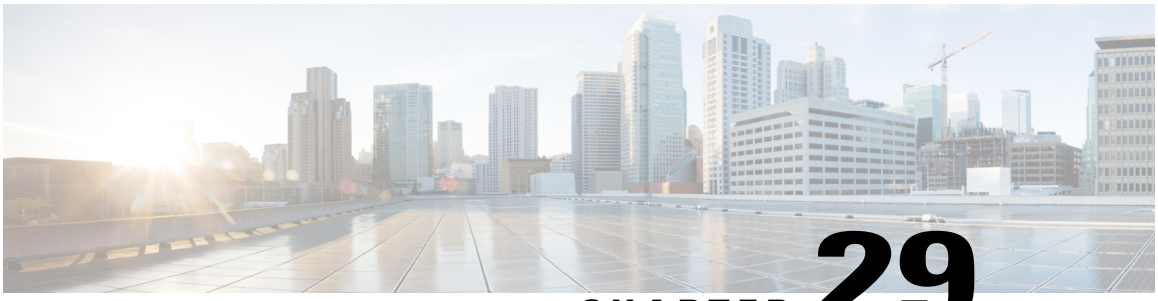
Example: Disconnect while Sending Metadata with BYE

When the original call disconnects without any reason, CUBE initiates a BYE session with the recording server along with the metadata.

In this case, the metadata contains the end time of the session along with the disassociation time of all the active participants from the call.

```
BYE sip:5555555@8.41.17.71:13961;transport=UDP SIP/2.0
...
Reason: Q.850;cause=16
Content-Type: application/rs-metadata+xml
Content-Disposition: recording-session
Content-Length: 984

<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
    <datamode>complete</datamode>
    <session session_id="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 29

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

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 38: Feature Information for Network-Based Recording of Video Calls Using Cisco Unified Border Element

Feature Name	Releases	Feature Information
Network-Based Recording of Video Calls Using Cisco Unified Border Element	15.3(3)M Cisco IOS XE Release 3.10S	The Network-Based Recording of Video Calls Using Cisco Unified Border Element feature supports software-based forking and recording of video calls. The following commands were introduced or modified: **media profile video**, **ref-frame-req rtcp**, **ref-frame-req sip-info**, **video profile**, **h264-packetization-mode**, **monitor-ref-frames**.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**media profile video** *media-profile-tag* **Example:** `Device(config)# media profile video 1`	Configures a video media profile and enters media profile configuration mode.
Step 4	Do one of the following: • **ref-frame-req rtcp retransmit-count** *retransmit-number* • **ref-frame-req sip-info** **Example:** `Device(cfg-mediaprofile)# ref-frame-req rtcp retransmit-count 4`	Enables FIR using the RTCP or SIP INFO method.

	Command or Action	Purpose
	Example: `Device(cfg-mediaprofile)# ref-frame-req sip-info`	
Step 5	**end** **Example:** `Device(cfg-mediaprofile)# end`	Exits media profile configuration mode.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**media profile video** *media-profile-tag* **Example:** `Device(config)# media profile video 1`	Configures a video media profile and enters media profile configuration mode.

	Command or Action	Purpose
Step 4	**h264-packetization-mode** *packetization mode* **Example:** `Device(cfg-mediaprofile)# h264-packetization-mode 2`	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.
Step 5	**end** **Example:** `Device(cfg-mediaprofile)# end`	Exits media profile configuration mode.

Monitoring Reference files or Intra Frames

Perform this task to configure device to perform deep packet inspection (DPI) of RTP packets received from an endpoint and keep track of how many instantaneous decoder refresh (IDR) frames have been received and the timestamp of the IDRs.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **media profile video** *media-profile-tag*
4. **monitor-ref-frames**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**media profile video** *media-profile-tag* **Example:** `Device(config)# media profile video 1`	Configures a video media profile and enters media profile configuration mode.

	Command or Action	Purpose
Step 4	**monitor-ref-frames** **Example:** `Device(cfg-mediaprofile)# monitor-ref-frames`	Monitors reference frames or intra-frames.
Step 5	**end** **Example:** `Device(cfg-mediaprofile)# end`	Exits media profile configuration mode.

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
VideoRtcpIntraFrameRequestCount=1
VideoRtcpIntraFrameRequestCount=1

!CUBE does not generate FIR request on forked leg
VideoRtcpIntraFrameRequestCount=0
```

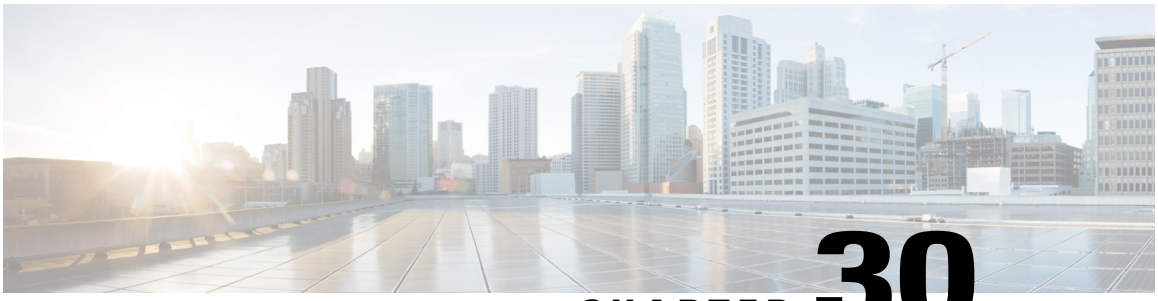
Step 3 **show call active video called-number** *number* **| include VideoSipInfoIntraFrameRequestCount**
Displays the number of SIP INFO FIR requests sent on each leg.

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**
Djsplays the number of IDR frames received on that call leg.



CHAPTER 30

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

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

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

Table 39: Feature Information for Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording

Feature Name	Releases	Feature Information
Third-Party GUID Capture for Correlation Between Calls and SIP-based Recording	Cisco IOS 15.4(3)M Cisco IOS XE 3.13S	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.

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

	Command or Action	Purpose
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	Purpose
Step 3	**voice class sip-copylist** *tag* **Example:** `Device(config)# voice class sip-copylist 100`	Configures a list of entities to be sent to a peer call leg and enters voice class configuration mode.
Step 4	**sip-header** *ThirdParty-GUID-headername* **Example:** `Device(config-class)# sip-header Third-Party-GUID`	Specifies that the third-party GUID header must be copied from the inbound dial-peer leg to the outbound dial-peer call leg.
Step 5	**exit** **Example:** `Device(config-class)# exit`	Exits voice class configuration mode.
Step 6	**dial-peer voice** *inbound-dialpeer-tag* **voip** **Example:** `Device(config)# dial-peer voice 2 voip`	Enters inbound dial-peer configuration mode.
Step 7	**voice class sip-copylist** *tag* **Example:** `Device(config-dial-peer)# voice class sip-copylist 100`	Applies the copy list to the dial peer.
Step 8	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits to global configuration mode.
Step 9	**dial-peer voice** *outbound-dialpeer-tag* **voip** **Example:** `Device(config)# dial-peer voice 3 voip`	Enters outbound dial-peer configuration mode.
Step 10	**voice class sip-copylist** *tag* **Example:** `Device(config-dial-peer)# voice class sip-copylist 100`	Applies the copy list to the dial peer.
Step 11	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits to global configuration mode.

	Command or Action	Purpose
Step 12	**voice class sip-profiles** *profile-id* **Example:** `Device(config)# voice class sip-profiles 10`	Creates a SIP profile and enters voice class configuration mode.
Step 13	**request INVITE peer-header sip** *GUID-header-to-copy* **copy** *header-value-to-match copy-variable* **Example:** `Device(config-class)# request INVITE peer-header sip Third-Party-GUID copy "(.*)" u01`	Copies headers from the INVITE message of the incoming dial peer into a copy variable.
Step 14	**request INVITE sip-header** *header-to-add* **add** *header-value-to-add* **Example:** `Device(config-class)# request INVITE sip-header Unsupported add "Unsupported: Dummy Header"`	Adds a SIP header to a SIP request.
Step 15	**request INVITE sip-header** *GUID-header-to-modify* **modify** *header-value-to-match header-value-to-replace* **Example:** `Device(config-class)# request INVITE sip-header Unsupported modify ".*" "Third-Party-GUID: \u01"`	Modifies the outgoing header using the copy variable defined in the previous step.
Step 16	**exit** **Example:** `Device(config-class)# exit`	Exits to global configuration mode.
Step 17	**dial-peer voice** *recorder-dial-peer-tag* **voip** **Example:** `Device(config)# dial-peer voice 2 voip`	Enters the dial peer configuration mode for the specified outbound recorder dial peer.
Step 18	**voice-class sip profiles** *profile-tag* **Example:** `Device(config-dial-peer)# voice-class sip profiles 30`	Applies the SIP profile to the recording dial peer.
Step 19	**end** **Example:** `Device(config-dial-peer)# end`	Exits to privileged EXEC mode.

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:

```
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
```

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.

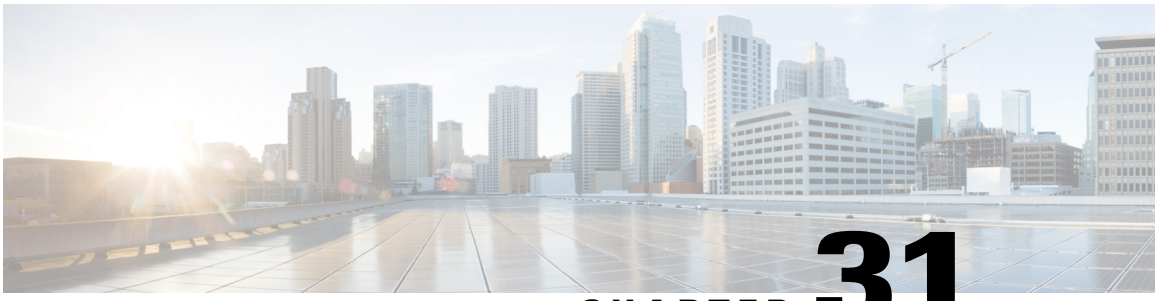
Example:

```
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 31

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

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 Name	Releases	Feature Information
Cisco Unified Communications Gateway Services	Cisco IOS 15.3(3)M Cisco IOS XE 3.10S	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.

Feature Name	Releases	Feature Information
Cisco UC Gateway Services API support for Secure RTP Forking	Cisco IOS 15.4(3)M Cisco IOS XE 3.13S	This feature provides support for Extended Media Forking (XMF) provider to monitor calls and trigger media forking on RTP and SRTP calls.
Support for Cisco UC Services API Media Forking with Survivability TCL	Cisco IOS 15.6(1)T Cisco IOS XE 3.17S	This feature allows media forking for the calls controlled by CVP Survivability TCL script with Cisco Unified Communication Services API.

Restrictions for Unified Communications Gateway Services—Extended Media Forking

- Media renegotiation is not supported.
- Media mixing on forked media streams is not supported.
- recordTone insertion is not supported with SRTP calls.
- mediaForkingReason tag is only to notify midcall stream events; notification for events such as codec change is not supported.
- Only voice media stream is supported.
- Supplementary services are not supported.
- High Availability is not supported.

Information About Cisco Unified Communications Gateway Services

Extended Media Forking (XMF) Provider and XMF Connection

The XMF provider allows applications to monitor calls and trigger media forking on the calls and has the capability to service up to 32 applications. The XMF provider can invoke a call-based or a connection-based media forking using the Unified Communications (UC) API. After the media forking is invoked, it can preserve the media forking initiated by the web application if the WAN connection to the application is lost. The XMF provider also provides the recording tone to the parties involved in the call.

The XMF connection describes the relationship between an XMF call and the endpoint (or trunk) involved in the call. A connection abstraction maintained in the gateway has the following connection states:

- IDLE: This state is the initial state for all new connections. Such connections are not actively part of a telephone call, yet their references to the Call and Address objects are valid. Connections typically do not stay in the IDLE state for long and quickly transition to other states. The application may choose to be notified at this state using the event filters and if done, call/connection at the gateway provider will use the NotifyXmfConnectionData(CREATED) message to notify the application listener that a new connection is created.
- ADDRESS_COLLECT: In this state the initial information package is collected from the originating party and is examined according to the “dialing plan” to determine the end of collection of addressing

information. In this state, the call in the gateway collects digits from the endpoint. No notification is provided.

- CALL_DELIVERY: On the originating side, this state involves selecting of the route as well as sending an indication of the desire to set up a call to the specified called party. On the terminating side, this state involves checking the busy/idle status of the terminating access and also informing the terminating message of an incoming call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (CALL_DELIVERY) message to notify the application listener.
- ALERTING: This state implies that the Address is being notified of an incoming call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (ALERTING) message to notify the application listener.
- CONNECTED: This state implies that a connection and its Address is actively part of a telephone call. In common terms, two parties talking to one another are represented by two connections in the CONNECTED state. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (CONNECTED) message to notify the application listener.
- DISCONNECTED: This state implies it is no longer part of the telephone call. A Connection in this state is interpreted as once previously belonging to this telephone call. The application may choose to be notified at this state using the event filters and if done, the call or connection at the gateway provider will use the NotifyXmfConnectionData (DISCONNECTED) message to notify the application listener.

XMF Call-Based Media Forking

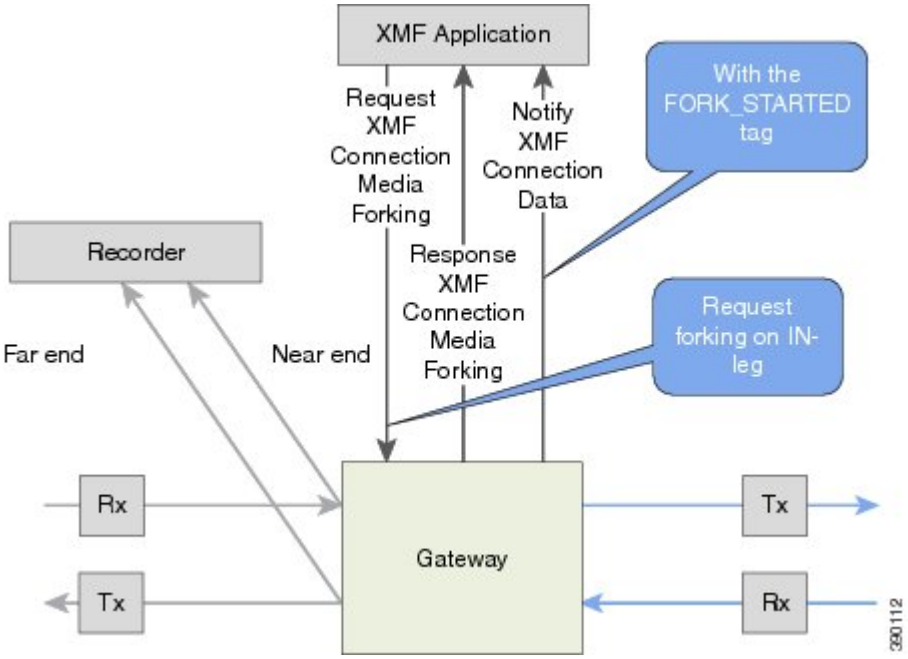
In call-based media forking of the gateway, the stream from the calling party is termed as near-end stream and the stream from the called party is termed as far-end stream. The XMF provider actively handles single media forking request per session. Any new media forking request from the external application will override or stop the current forking instance and would start a new forking instance (to the appropriate target IP address or ports). After the media forking request is accepted, the XMF provider returns a response message and starts to fork media streams of a connection to the target forked streams. A NotifyXmfCallData message will be notified to the application for the updated media forking status, that is, FORK-FAILED, FORK_STARTED, or FORK_DONE.

XMF Connection-Based Media Forking

In connection-based media forking of the gateway, the incoming stream to the connection is termed as near-end stream and the outgoing stream of the connection is termed as far-end stream. The XMF provider actively handles single media forking request per session. Any new media forking request from the external application will override or stop the current forking instance and would start a new forking instance (to the appropriate

target IP address or ports). After the media forking request is accepted, the XMF provider returns a response message and starts to fork media streams of a connection to the target forked streams.

Figure 25: 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.

Cisco UC Gateway Services Media Forking API with Survivability TCL

Cisco Unified Border Element (CUBE) supports Survivability TCL Script to co-exist with Cisco Unified Communication (UC) Services API.

Cisco UC Services API XMF interface supports media forking for all the calls controlled by survivability TCL script including the survivability re-attempted calls. Thus, all the calls controlled by survivability TCL script can be recorded when requested by Cisco UC Services XMF API.

Cisco Unified Communications Manager controlled Gateway recording utilizes XMF to trigger media forking on CUBE or SIP based PSTN gateways in the supported call flows.

Note Media forking is allowed only for survivability TCL script supported by Cisco Unified Customer Voice Portal (CVP). CVP survivability TCL script is not supported in High Availability mode.

The following call scenarios are supported:

- Basic comprehensive call
- Calls with Refer Consume
- Calls with Mid-call failure
- Calls with alternative route with initial call failure

There are no configuration changes required for enabling CVP survivability TCL support with Cisco UC Gateway Services API.

Media Forking for SRTP Calls

- SRTP forking is supported in XCC and XMF application service providers and the supported APIs are RequestCallMediaForking, RequestCallMediaSetAttributes, and RequestConnectionMediaForking.
- SRTP forking is supported for SRTP-to-SRTP, SRTP-to-RTP, and RTP-to-SRTP calls.
  - For SRTP-to-SRTP calls, media forking on either leg would result in SRTP streams being forked.
  - For SRTP fallback calls, after the initial offer, CUBE will fall back to RTP. Media forking either call legs would result in RTP streams being forked.
  - For SRTP-to-RTP interworking calls, a digital signal processor (DSP) is required and involves transcoding. In this case, one leg would be SRTP and the other leg RTP.
- SRTP Crypto keys are notified over the API.
- Supports automatic stopping of media forking when stream changes from SRTP or to SRTP.
  - The optional mediaForkingReason tag in XMF or XCC Notify messages indicates that the forking has been stopped internally.
  - mediaForkingReason tag is only present when the connection changes state, such as mid-call re-INVITE. SRTP stream can change to RTP or SRTP stream can change keys mid-call.
  - mediaForkingReason tag is always accompanied by FORK_DONE.

Crypto Tag

For SRTP forking, the optional Crypto tag in NotifyXmfConnectionData or NotifyXmfCallData message indicates the context of an actively forked SRTP connection.

Note The Crypto tag is only present in the notification message where FORK_STARTED tag is present.

The optional Crypto tag specifies the following:

- The Crypto suite used for encryption and authentication algorithm.
- The base64 encoded mastery key and salt used for encryption.

Crypto suite can be one of the two suites supported in IOS:

- AES_CM_128_HMAC_SHA1_32
- AES_CM_128_HMAC_SHA1_80

Example of SDP Data sent in an SRTP Call

Original SIP SDP Crypto Offer	SIP SDP Crypto Answer
v=0	v=0
o=CiscoSystemsSIP-GW-UserAgent 7826 3751 IN IP4 172.18.193.98	o=CiscoSystemsSIP-GW-UserAgent 7826 3751 IN IP4 172.18.193.98
s=SIP Call	s=SIP Call
c=IN IP4 172.18.193.98	c=IN IP4 172.18.193.98
t= 0 0	t=0 0
m=audio 51372 RTP/SAVP 0	m=audio 49170 RTP/SAVP 0
a=rtpmap:0 PCMU/8000	**a=crypto:1 AES_CM_128_HMAW_SHA1_32**
a=crypto:1 AES_CM_128_HMAC_SHA1_32	**inline:NzB4d1BINUAvLEw6UzF3WSJ+PSdFcGdUJShpX1Zj**
inline:d0RmdmcmVCspEc3QGZiNWpVLFJhQX1cfHAwJSoj	

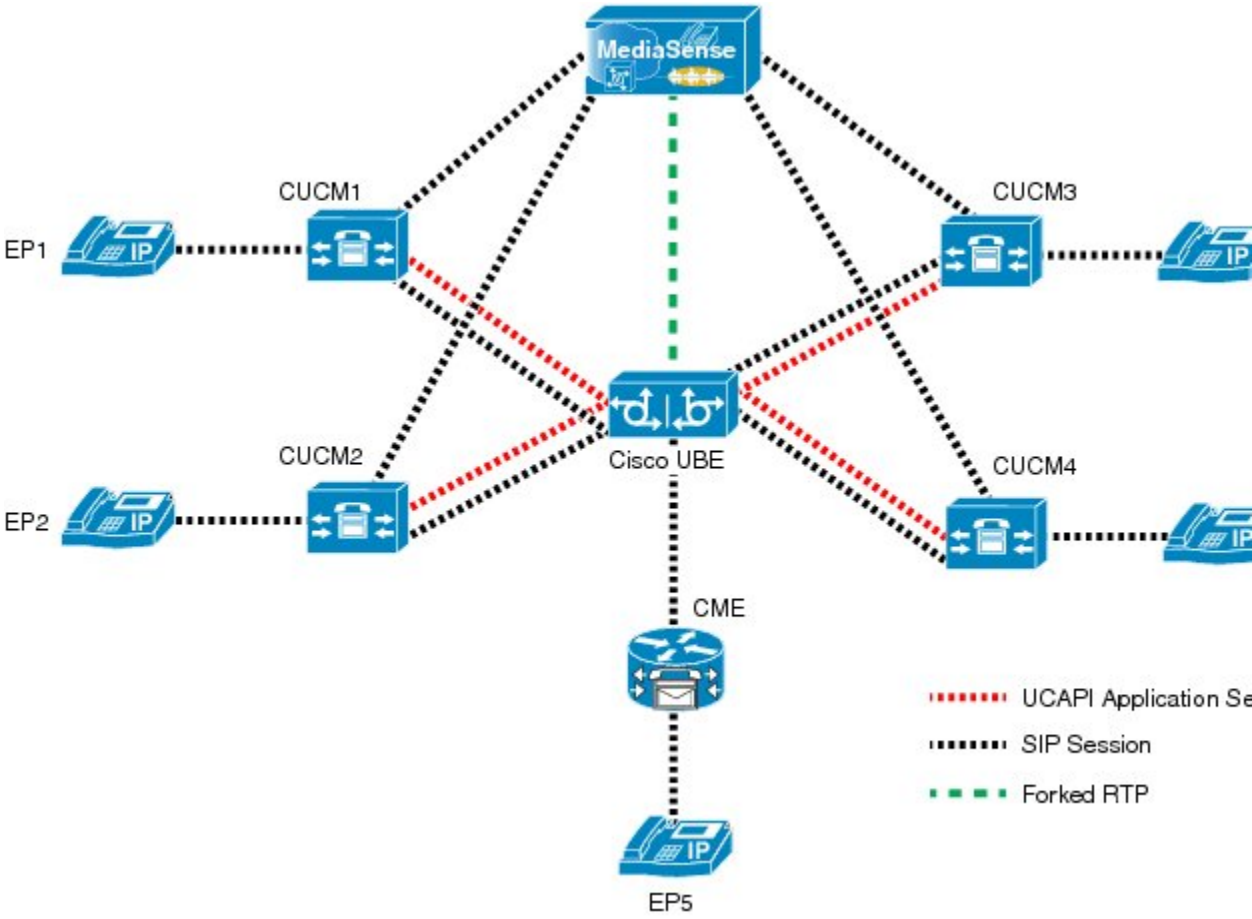
Note The application is notified of the content in Crypto and inline SDP lines.

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.

Figure 26: Multiple XMF Applications and Recording Tone



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.

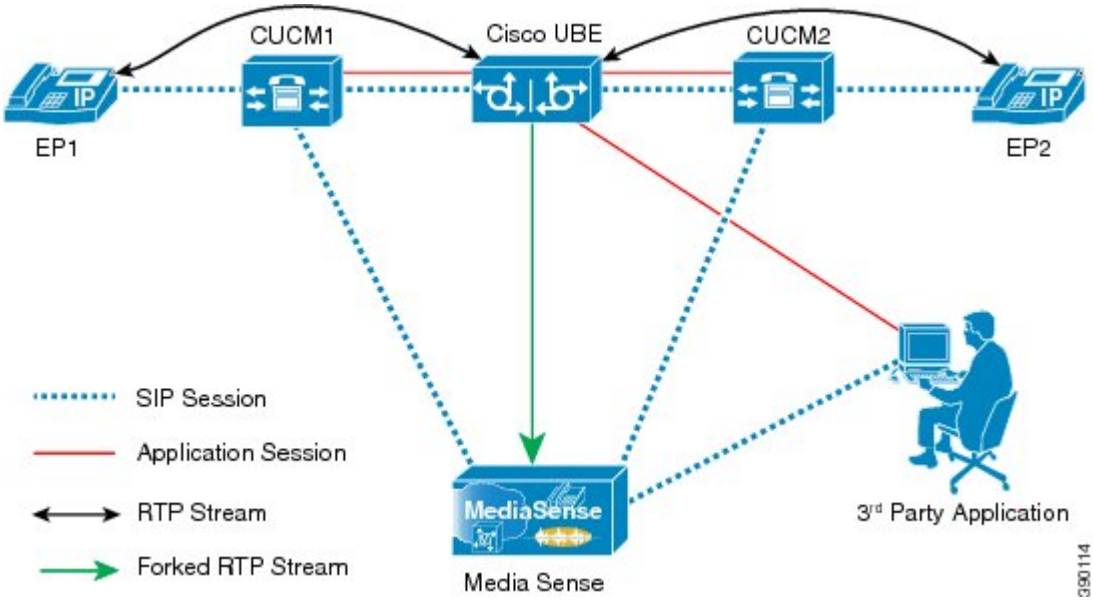
If “recordTone” parameter is enabled only on the farEndAddr, then this tone is played only on the outgoing leg. Likewise, if enabled only on the nearEndAddr, then the tone is played only on the incoming leg. When enabled in both the far and near end, then recording tone is played on both the legs.

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 27: Forking Preservation



The “preserve” parameter value can be set to TRUE or FALSE in any of the 3 forking requests (RequestXmfConnectionMediaForking, RequestXmfCallMediaForking, or RequestXmfCallMediaSetAttributes) from the application to Cisco UBE.

- If the “preserve” parameter received is TRUE, then forking will continue the recording, even if the WAN connection to application is lost or application is unregistered.
- If the “preserve” parameter received is FALSE, then forking will not continue the recording.

- If the “preserve” parameter is not received in the media forking request, then forking will not continue the recording.

How to Configure UC Gateway Services

Configuring Cisco Unified Communication IOS Services on the Device

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http server**
4. **ip http max-connections** *value*
5. **ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value*
6. **http client connection idle timeout** *seconds*
7. **uc wsapi**
8. **message-exchange max-failures** *number*
9. **probing max-failures** *number*
10. **probing interval keepalive** *seconds*
11. **probing interval negative** *seconds*
12. **source-address** *ip-address*
13. **end**

DETAILED STEPS

	Command or Action	Purpose
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	**ip http server** **Example:** `Device(config)# ip http server`	Enables the HTTP server (web server) on the system.

	Command or Action	Purpose
Step 4	**ip http max-connections** *value* **Example:** `Device(config)# ip http max-connection 100`	Sets the maximum number of concurrent connections to the HTTP sever that will be allowed. The default value is 5.
Step 5	**ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value* **Example:** `Device(config)# ip http timeout-policy idle 600 life 86400 requests 86400`	Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are: • **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 connection is established. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the number of requests is reached, it may close the connection sooner. Also, since the server will not close the connection while actively processing a request, the connection may remain open longer than the specified life time if processing is occurring when the life maximum is reached. In this case, the connection will be closed when processing finishes. The default value is 180 seconds (3 minutes). The maximum value is 86400 seconds (24 hours). • **requests—**The maximum limit on the number of requests processed on a persistent connection before it is closed. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the life time is reached, the connection may be closed before the maximum number of requests are processed. The default value is 1. The maximum value is 86400.
Step 6	**http client connection idle timeout** *seconds* **Example:** `Device(config)# http client connection idle timeout 600`	Sets the number of seconds that the client waits in the idle state until it closes the connection.
Step 7	**uc wsapi** **Example:** `Device(config)# uc wsapi`	Enters Cisco Unified Communication IOS Service configuration mode.

	Command or Action	Purpose
Step 8	**message-exchange max-failures** *number* **Example:** `Device(config-uc-wsapi)# message-exchange max-failures 2`	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.
Step 9	**probing max-failures** *number* **Example:** `Device(config-uc-wsapi)# probing max-failures 5`	Configures the maximum number of failed probing messages before the router unregisters the application. Range is 1 to 5. Default is 3.
Step 10	**probing interval keepalive** *seconds* **Example:** `Device(config-uc-wsapi)# probing interval keepalive 255`	Configures the maximum number of failed probing messages before the router unregisters the application. Range is 1 to 5. Default is 3.
Step 11	**probing interval negative** *seconds* **Example:** `Device(config-uc-wsapi)# probing interval negative 10`	Configures the interval between negative probing messages, in seconds.
Step 12	**source-address** *ip-address* **Example:** `Device(config-uc-wsapi)# source-address 192.1.12.14`	Configures the IP address (hostname) as the source IP address for the UC IOS service. **Note** The source IP address is used by the provider in the NotifyProviderStatus messages.
Step 13	**end** **Example:** `Device(config-uc-wsapi)# end`	Returns to privileged EXEC mode.

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

	Command or Action	Purpose
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	**uc wsapi** **Example:** `Device(config)# uc wsapi`	Enters Cisco Unified Communication IOS Service configuration mode.
Step 4	**source-address** *ip address* **Example:** `Device(config)# source-address 172.156.19.38`	Configures the source ip address.
Step 5	**provider xmf** **Example:** `Device(config-uc-wsapi)# provider xmf`	Enters XMF provider configuration mode.

	Command or Action	Purpose
Step 6	**no shutdown** **Example:** `Device(config-uc-wsapi)# no shutdown`	Activates XMF provider.
Step 7	**remote-url** *index url* **Example:** `Device(config-uc-wsapi)# remote-url 1 http://test.com:8090/ucm_xmf`	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.
Step 8	**end** **Example:** `Device(config-uc-wsapi)# end`	Returns to privileged EXEC mode.

Verifying the UC Gateway Services

The **show** commands can be entered in any order.

SUMMARY STEPS

1. **enable**
2. **show wsapi registration all**
3. **show wsapi registration xmf** *remote-url-index*
4. **show call media-forking**

DETAILED STEPS

Step 1 **enable**
Enables privileged EXEC mode.

Example:
```
Device> enable
```

Step 2 **show wsapi registration all**
Displays the details of applications registered. Each registered application is identified by a different ID.

Example:
```
Device# show wsapi registration all
```

```
 Provider XMF
=====================================================
registration index: 11
  id: 2E7C3034:XMF:myapp:26
  appUrl:http://pascal-lnx.cisco.com:8094/xmf
  appName: myapp
  provUrl: http://9.45.46.16:8090/cisco_xmf
  prober state: STEADY
  connEventsFilter:
CREATED|REDIRECTED|ALERTING|CONNECTED|TRANSFERRED|CALL_DELIVERY|DISCONNECTED|HANDOFF_JOIN|HANDOFF_LEAVE

  mediaEventsFilter: DTMF|MEDIA_ACTIVITY|MODE_CHANGE|TONE_DIAL|TONE_OUT_OF_SERVICE|TONE_SECOND_DIAL

registration index: 1
  id: 2E7C304A:XMF:myapp:27
  appUrl:http://pascal-lnx.cisco.com:8092/xmf
  appName: myapp
  provUrl: http://9.45.46.16:8090/cisco_xmf
  prober state: STEADY
  connEventsFilter:
CREATED|REDIRECTED|ALERTING|CONNECTED|TRANSFERRED|CALL_DELIVERY|DISCONNECTED|HANDOFF_JOIN|HANDOFF_LEAVE

  mediaEventsFilter: DTMF|MEDIA_ACTIVITY|MODE_CHANGE|TONE_DIAL|TONE_OUT_OF_SERVICE|TONE_SECOND_DIAL

registration index: 21
  id: 2E7C6423:XMF:myapp:28
  appUrl:http://pascal-lnx.cisco.com:8096/xmf
  appName: myapp
  provUrl: http://9.45.46.16:8090/cisco_xmf
  prober state: STEADY
  connEventsFilter:
CREATED|REDIRECTED|ALERTING|CONNECTED|TRANSFERRED|CALL_DELIVERY|DISCONNECTED|HANDOFF_JOIN|HANDOFF_LEAVE

  mediaEventsFilter: DTMF|MEDIA_ACTIVITY|MODE_CHANGE|TONE_DIAL|TONE_OUT_OF_SERVICE|TONE_SECOND_DIAL

registration index: 31
  id: 2E7C69E8:XMF:myapp:29
  appUrl:http://pascal-lnx.cisco.com:8098/xmf
  appName: myapp
  provUrl: http://9.45.46.16:8090/cisco_xmf
  prober state: STEADY
  connEventsFilter:
CREATED|REDIRECTED|ALERTING|CONNECTED|TRANSFERRED|CALL_DELIVERY|DISCONNECTED|HANDOFF_JOIN|HANDOFF_LEAVE

  mediaEventsFilter: DTMF|MEDIA_ACTIVITY|MODE_CHANGE|TONE_DIAL|TONE_OUT_OF_SERVICE|TONE_SECOND_DIAL
```

Step 3 **show wsapi registration xmf** *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
=====================================================
registration index: 1
  id: 2E7C6423:XMF:myapp:28
  appUrl:http://pascal-lnx.cisco.com:8096/xmf
  appName: myapp
  provUrl: http://9.45.46.16:8090/cisco_xmf
  prober state: STEADY
  connEventsFilter:
CREATED|REDIRECTED|ALERTING|CONNECTED|TRANSFERRED|CALL_DELIVERY|DISCONNECTED|HANDOFF_JOIN|HANDOFF_LEAVE

  mediaEventsFilter: DTMF|MEDIA_ACTIVITY|MODE_CHANGE|TONE_DIAL|TONE_OUT_OF_SERVICE|TONE_SECOND_DIAL
```

Step 4 **show call media-forking**

Displays the forked stream information.

Example:

```
Device# show call media-forking

Warning: Output may be truncated if sessions are added/removed concurrently!

Session    Call     n/f  Destination (port address)
187        BA       near 45864 10.104.105.232
188        BA       far  54922 10.104.105.232
189        B9       near 45864 10.104.105.232
190        B9       far  54922 10.104.105.232

FORK _DONE Notifications

//WSAPI/INFRA/wsapi_send_outbound_message_by_provider_info:
*Dec 21 10:31:21.016 IST: //WSAPI/INFRA/0/9/546CF8:25:tx_contextp 15898C1C tx_id 19 context1 (0 0)
context2 (9 9):
out_url http://gauss-lnx.cisco.com:8081/xmf*Dec 21 10:31:21.020 IST:
wsapi_send_outbound_message_by_provider_info:
<?xml version="1.0" encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"><SOAP:Body>
<NotifyXmfConnectionData xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0"><msgHeader><transactionID>
546CF8:25</transactionID><registrationID>4CA5E4:XMF:myapp:4</registrationID></msgHeader><callData><callID>25</callID><state>
ACTIVE</state></callData><connData><connID>132</connID><state>ALERTING</state></connData><event><mediaForking>
<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):
out_url http://gauss-lnx.cisco.com:8081/xmf*Dec 21 10:31:21.020 IST:
wsapi_send_outbound_message_by_provider_info:
<?xml version="1.0" encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"><SOAP:Body>
<NotifyXmfConnectionData xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0"><msgHeader><transactionID>
546CF8:25</transactionID><registrationID>4CA5E4:XMF:myapp:4</registrationID></msgHeader><callData><callID>25</callID><state>
ACTIVE</state></callData><connData><connID>132</connID><state>ALERTING</state></connData><event><mediaForking>
<mediaForkingState>FORK_FAILED</mediaForkingState></mediaForking></event></NotifyXmfConnectionData></SOAP:Body>
</SOAP:Envelope>
```

Troubleshooting Tips

You can use the following **debug** commands to troubleshoot the UC Gateway Services configurations.

- **debug wsapi infrastructure all**
- **debug wsapi xcc all**
- **debug wsapi xmf all**
- **debug wsapi xmf messages**
- **debug wsapi infrastructure detail**
- **debug voip application**
- **debug voip application media forking**

Configuration Examples for UC Gateway Services

Example: Configuring Cisco Unified Communication IOS Services

The following example shows how to configure the device for Cisco Unified Communication IOS Services and enable the HTTP server:

```
Device> enable
Device# configure terminal
Device(config)# ip http server
Device(config)# ip http max-connection 100
Device(config)# ip http timeout-policy idle 600 life 86400 requests 86400
Device(config)# http client connection idle timeout 600
Device(config)# uc wsapi
Device(config-uc-wsapi)# message-exchange max-failures 2
Device(config-uc-wsapi)# probing max-failures 5
Device(config-uc-wsapi)# probing interval keepalive 255
Device(config-uc-wsapi)# probing interval negative 10
Device(config-uc-wsapi)# source-address 192.1.12.14
Device(config-uc-wsapi)# end
```

Example: Configuring the XMF Provider

The following example shows how to enable the XMF providers. The configuration specifies the address and port that the application uses to communicate with the XMF provider:

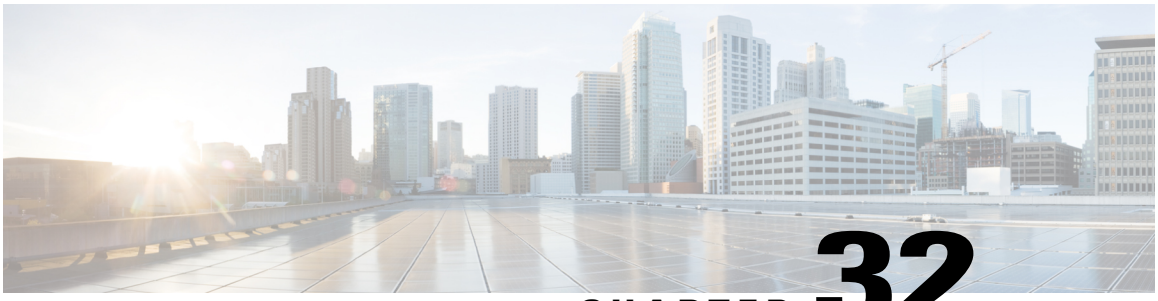
```
Device> enable
Device# configure terminal
Device(config)# uc wsapi
Device(config-uc-wsapi)# provider xmf
Device(config-uc-wsapi)# no shutdown
Device(config-uc-wsapi)# remote-url 1 http://test.com:8090/ucm_xmf
Device(config-uc-wsapi)# end
```

Example: Configuring UC Gateway Services

```
uc wsapi
 message-exchange max-failures 5
 response-timeout 10
 source-address 192.1.12.14
 probing interval negative 20
 probing interval keepalive 250
 !
 provider xmf
  remote-url 1 http://pascal-lnx.cisco.com:8050/ucm_xmf
```

PART VII

SIP Header Manipulation



CHAPTER 32

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

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 Copying with SIP Profiles

Feature Name	Releases	Feature Information
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 another. This is done by copying the content of messages into variables which can then be used to modify other SIP headers. This feature modifies the following commands: **voice class sip-profiles**, **response**, **request**, **voice-class sip copy-list**, **sip-header**

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

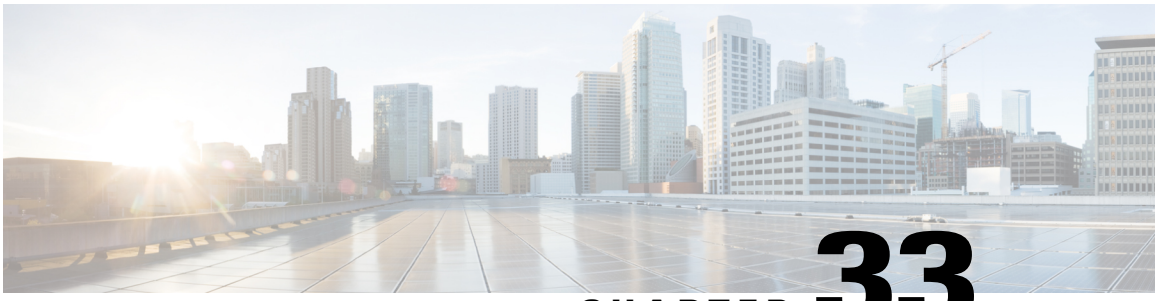
```
Contact: <sip:89016442998@10.86.176.19:5060>;x-cisco-tip
```

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 33

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

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

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

Table 41: Feature Information for Copying with SIP Profiles

Feature Name	Releases	Feature Information
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 another. This is done by copying the content of messages into variables which can then be used to modify other SIP headers. This feature modifies the following commands: **voice class sip-profiles**, **response**, **request**, **voice-class sip copy-list**, **sip-header**

How to Copy SIP Header Fields to Another

Copying From an Incoming Header and Modifying an Outgoing Header

To copy content from an incoming header that a device receives to an outgoing header, configure a SIP copylist for that header and apply it to an incoming dial peer. A SIP profile is configured to copy this incoming header to a user-defined variable and apply it to an outgoing header.

Before You Begin

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class sip-copylist** *tag*
4. Do one of the following:
   - **sip-header** *header-name*
   - **sip-header SIP-Req-URI**
5. **exit**
6. **dial-peer voice** *inbound-dial-peer-tag* **voip**
7. **voice class sip-copylist** *tag*
8. **exit**
9. **voice class sip-profiles** *profile-id*
10. {**request** | **response**} *message* **peer-header sip** *header-to-copy* **copy** *header-value-to-match copy-variable*
11. {**request** | **response**} *message* {**sip-header** | **sdp-header**} *header-to-modify* **modify** *header-value-to-match header-value-to-replace*
12. **exit**
13. **dial-peer voice** *inbound-dial-peer-tag* **voip**
14. **voice class sip-copylist** *tag*
15. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal**	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**voice class sip-copylist** *tag* **Example:** `Device(config)# voice class sip-copylist 100`	Configures a list of entities to be sent to a peer call leg and enters voice class configuration mode.
Step 4	Do one of the following: • **sip-header** *header-name* • **sip-header SIP-Req-URI** **Example:** `Device(config-class)# sip-header To`	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** **Example:** `Device(config)# dial-peer voice 2 voip`	Enters the dial peer configuration mode for the specified inbound dial peer.
Step 7	**voice class sip-copylist** *tag* **Example:** `Device(config-dial-peer)# voice class sip-copylist 100`	Applies the copy list to the dial-peer.
Step 8	**exit**	Exits to global configuration mode.
Step 9	**voice class sip-profiles** *profile-id* **Example:** `Device(config)# voice class sip-profiles 10`	Creates a SIP Profiles and enters voice class configuration mode.
Step 10	{**request** | **response**} *message* **peer-header sip** *header-to-copy* **copy** *header-value-to-match copy-variable* **Example:** `Device(config-class)# request INVITE peer-header sip TO copy "sip:(.*)@" u01`	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.

	Command or Action	Purpose
	Example: `Device(config-class)# request INVITE sip-header SIP-Req-URI modify ".*@(.*)" "INVITE sip:\u01@\1"`	
Step 12	**exit**	Exits to global configuration mode.
Step 13	**dial-peer voice** *inbound-dial-peer-tag* **voip** **Example:** `Device(config)# dial-peer voice 2 voip`	Enters the dial peer configuration mode for the specified inbound dial peer.
Step 14	**voice class sip-copylist** *tag* **Example:** `Device(config-dial-peer)# voice class sip-copylist 100`	Applies the copy list to the dial-peer.
Step 15	**exit**	Exits to global configuration mode.

What to Do Next

Apply the SIP profile to an 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

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal**	Enters global configuration mode.
Step 3	**voice class sip-profiles** *profile-id* **Example:** `Device(config)# voice class sip-profiles 10`	Creates a SIP profile and enters voice class configuration mode.
Step 4	{**request** | **response**} *message* {**sip-header** | **sdp-header**} *header-to-copy* **copy** *header-value-to-match copy-variable* **Example:** `Device(config-class)# request INVITE sip-header TO copy "sip:(.*)@" u01`	Copies the contents of the specified header from an outbound message into a copy variable.
Step 5	{**request** | **response**} *message* {**sip-header** | **sdp-header**} *header-to-modify* **modify** *header-value-to-match header-value-to-replace* **Example:** `Device(config-class)# request INVITE sip-header SIP-Req-URI modify ".*@(.*)" "INVITE sip:\u01@\1"`	Modifies an outgoing SIP or SDP header using the copy variable defined in the previous step.
Step 6	**end** **Example:** `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.

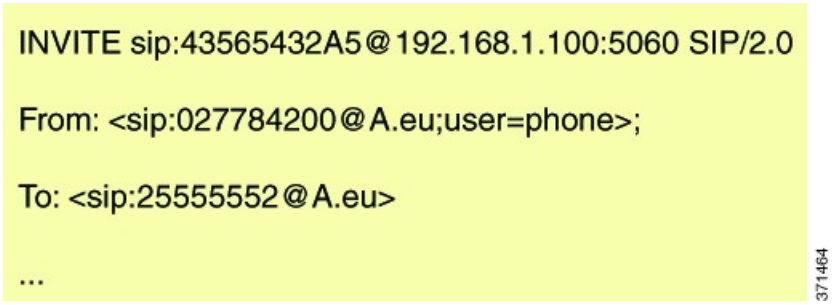
Example: Copying the To Header into the SIP-Req-URI

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.

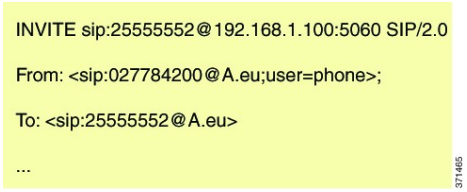
Given below is the original SIP message, where the INVITE has a non-routable value of 43565432A5. The actual phone destination number is 25555552 and is present in the To: SIP header.

Figure 28: Incoming SIP Message



Given below is the SIP message that is required. Note that 43565432A5 has changed to 25555552 in the SIP INVITE.

Figure 29: Modified SIP Message



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

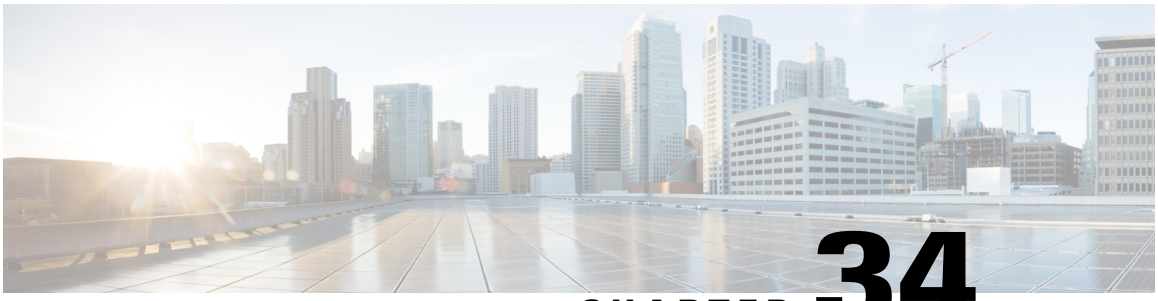
```
Device# voice class sip-profiles 1

!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
```



CHAPTER 34

Manipulating SIP Status-Line Header of SIP Responses

The SIP status line is a SIP response header, and it can be modified like any other SIP headers of a message. it can either be modified with a user-defined value, or the status line from an incoming response can be copied to an outgoing SIP response. The SIP header keyword used for the response status line is **SIP-StatusLine**.

Feature Information for Manipulating SIP Responses

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 Manipulating SIP Responses

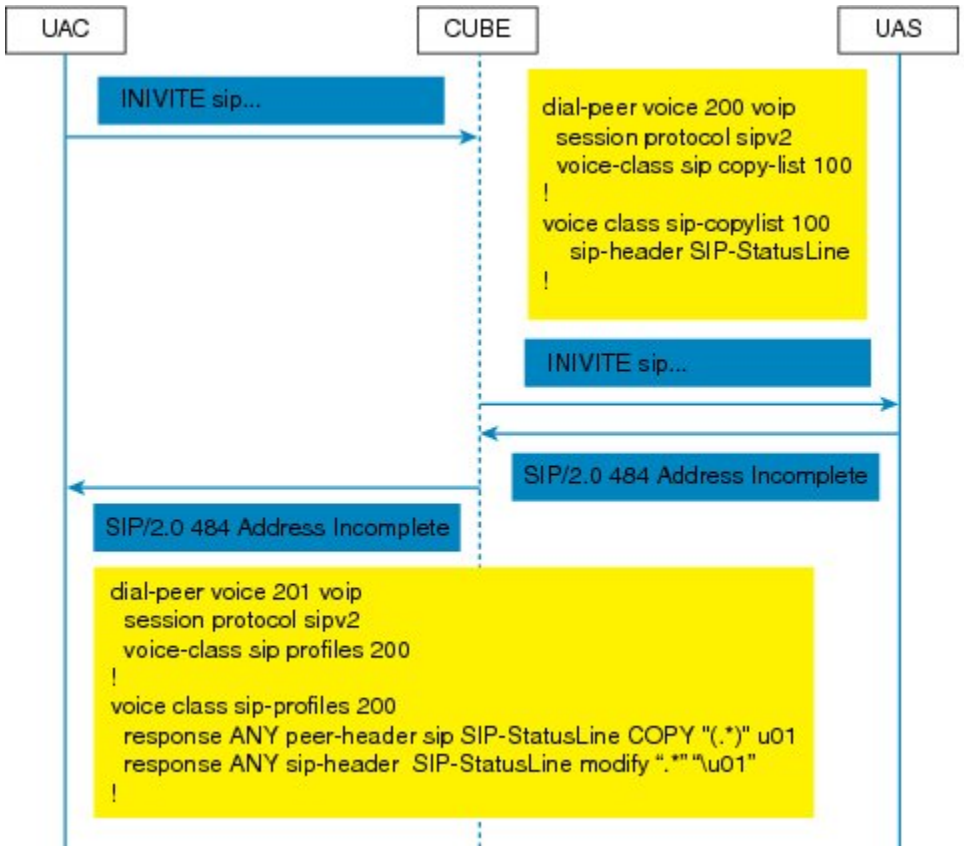
Feature Name	Releases	Feature Information
SIP Profile Enhancements for SIP responses and error codes	15.4(1)T Cisco IOS XE Release 3.12S	This feature extends SIP profiles to allow the following: • Modification of the outgoing SIP response status line. Previously, only modification of outgoing SIP requests and responses was possible. • Copying of the incoming SIP response status-line. The information from the peer-leg status-line can then be copied to user-variables and applied to the outbound response status-line. This option can be used to pass-thru the error-code and error phrase from peer-leg. Previously, only copying of SIP headers were possible. • Before applying a SIP profile to a response from CUBE, the response can be mapped to its corresponding request.
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 another. This is done by copying the content of messages into variables which can then be used to modify other SIP headers. This feature modifies the following commands: **voice class sip-profiles**, **response**, **request**, **voice-class sip copy-list**, **sip-header**

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 30: 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**
5. **exit**
6. **dial-peer voice** *inbound-dial-peer-id* **voip**
7. **voice-class sip copy-list** *list-id*
8. **exit**
9. **voice class sip-profiles** *tag*
10. **response** *response-code* **peer-header sip SIP-StatusLine copy** *match-pattern copy-variable*
11. **response** *response-code* **sip-header SIP-StatusLine modify** *match-pattern copy-variable*
12. **exit**

DETAILED STEPS

	Command or Action	Purpose
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	**voice class sip-copylist** *tag* **Example:** `Device(config)# voice class sip-copylist 1`	Configures a list of entities to be sent to the peer call leg and enters voice class configuration mode.
Step 4	**sip-header SIP-StatusLine** **Example:** `Device(config-class)# sip-header SIP-StatusLine`	Specifies that the Session Initiation Protocol (SIP) status line header must be sent to the peer call leg.
Step 5	**exit** **Example:** `Device(config-class)# exit`	Exits voice class configuration mode and returns to global configuration mode.
Step 6	**dial-peer voice** *inbound-dial-peer-id* **voip** **Example:** `Device(config)# dial-peer voice 99 voip`	Specifies an inbound dial peer and enters dial peer configuration mode.
Step 7	**voice-class sip copy-list** *list-id* **Example:** `Device(config-dial-peer)# voice-class sip copy-list 1`	Associates the SIP copy list with the inbound dial peer.
Step 8	**exit** **Example:** `Device(config-dial-peer)# exit`	Exits dial peer configuration mode and returns to global configuration mode.
Step 9	**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.

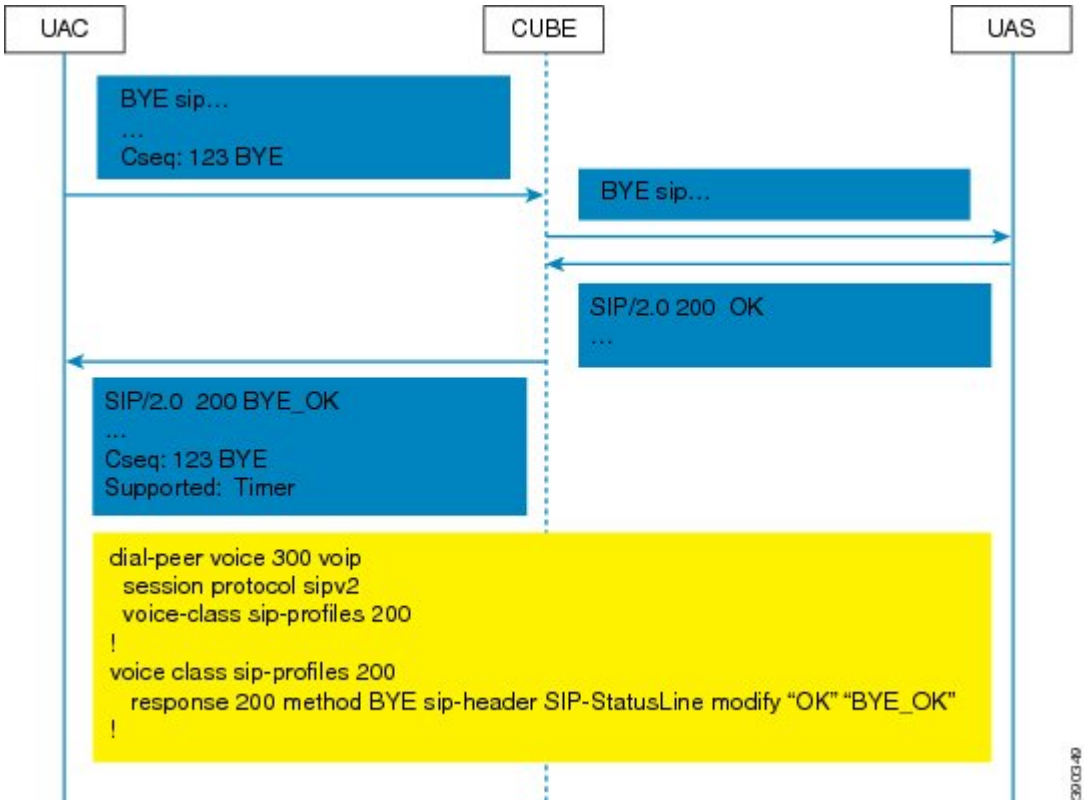
	Command or Action	Purpose
Step 10	**response** *response-code* **peer-header sip SIP-StatusLine copy** *match-pattern copy-variable* **Example:** `Device(config-class)# response ANY peer-header sip SIP-StatusLine copy "(.*)" u01`	Copies responses from the corresponding incoming call leg into a copy variable.
Step 11	**response** *response-code* **sip-header SIP-StatusLine modify** *match-pattern copy-variable* **Example:** `Device(config-class)# response ANY sip-header SIP-StatusLine modify ".*" "\u01"`	Modifies an outgoing response using the copy variable defined in the previous step.
Step 12	**exit** **Example:** `Device(config-class)# exit`	Exits voice class configuration mode and returns to global configuration mode.

What to Do Next

Apply the SIP profile to the outbound dial peer to copy the SIP response to the outbound leg.

Modifying Status-Line Header of Outgoing SIP Response with User Defined Values

Figure 31: Call Flow Configuring a New Status Line for an Outgoing SIP Response Based on an Incoming SIP Request



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class sip-profiles** *tag*
4. **response** *response-code* [**method** *method-type*] **sip-header SIP-StatusLine modify** *match-pattern replacement-pattern*
5. **exit**

DETAILED STEPS

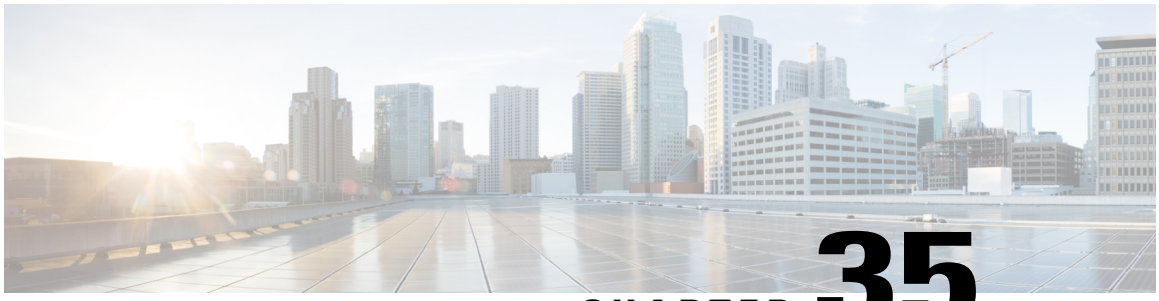
	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice class sip-profiles** *tag* **Example:** `Device(config)# 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 VIII

Payload Type Interoperability



CHAPTER 35

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

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 43: Feature Information for Dynamic Payload Interworking for DTMF and Codec Packets Support

Feature Name	Releases	Feature Information
Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls	15.0(1)XA 15.1(1)T	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**.
Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls	Cisco IOS Release XE 3.1S	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**.
High Availability Checkpointing Support for Asymmetric Payload	15.4(2)T	High availability support for asymmetric payload type interworking was added.
High Availability Checkpointing Support for Asymmetric Payload	Cisco IOS Release XE 3.12S	High availability support for asymmetric payload type interworking was added.

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 = audio <media-port1> RTP/AVP XXX m = video <media-port2> 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.

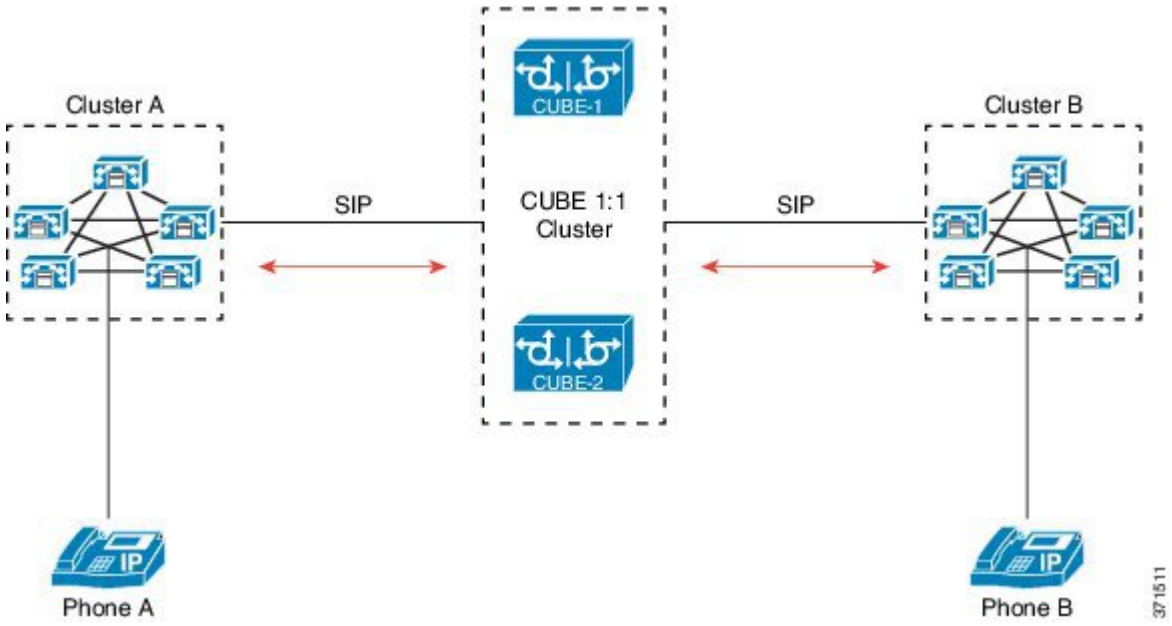
The Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature is enabled by default for a symmetric call. An offer is sent with a payload type based on the dial-peer configuration. The answer is sent with the same payload type as was received in the incoming offer. When 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 32: 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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable` **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters voice service configuration mode.
Step 4	**sip** **Example:** `Device(conf-voi-serv)# sip`	Enters voice service SIP configuration mode.
Step 5	**asymmetric payload** {**dtmf** | **dynamic-codecs** | **full** | **system**} **Example:** `Device(conf-serv-sip)# asymmetric payload full`	Configures global SIP asymmetric payload support. **Note** The **dtmf** and **dynamic-codecs** keywords are internally mapped to the **full** keyword to provide asymmetric payload type support for audio and video codecs, DTMF, and NSEs.
Step 6	**end** **Example:** `Device(conf-serv-sip)# end`	Exits voice service SIP configuration mode and enters privileged EXEC mode.

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

	Command or Action	Purpose
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 77 voip`	Enters dial peer voice configuration mode.
Step 4	**voice-class sip asymmetric payload** {**dtmf** | **dynamic-codecs** | **full** | **system**} **Example:** `Device(config-dial-peer)# voice-class sip asymmetric payload full`	Configures the dynamic SIP asymmetric payload support. **Note** The **dtmf** and **dynamic-codecs** keywords are internally mapped to the **full** keyword to provide asymmetric payload type support for audio and video codecs, DTMF, and NSEs.
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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show call active voice compact** **Example:** `Device# show call active voice compact`	(Optional) Displays a compact version of call information.
Step 3	**show call active voice** **Example:** `Device# show call active voice`	(Optional) Displays call information for voice calls in progress.

Troubleshooting Tips

Use the following commands to debug any errors that you may encounter when you configure the Dynamic Payload Type Interworking for DTMF and Codec Packets for SIP-to-SIP Calls feature:

- **debug ccsip all**
- **debug voip ccapi inout**
- **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 inout**
- **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

```
!
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
!
```

Example: Asymmetric Payload Interworking—Interworking Configuration

```
!
voice service voip
 allow-connections sip to sip
!
dial-peer voice 1 voip
```

```
 session protocol sipv2
 rtp payload-type cisco-codec-fax-ind 110
 rtp payload-type cisco-codec-video-h264 112
 session target ipv4:9.13.8.23
!
```

PART IX

Protocol Interworking

CHAPTER 36

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

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 Delayed-Offer to Early-Offer

Feature Name	Releases	Feature Information
Delayed-Offer to Early-Offer	Cisco IOS 12.4(3) Cisco IOS 12.4(24)T Cisco IOS 15.0(1)M	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**.
Midcall Renegotiation Support for DO-EO Calls	Cisco IOS 15.4(2)T Cisco IOS XE 3.12S	The Midcall renegotiation of codecs feature configures the midcall renegotiation of codecs, if an exchange of parameters that is not end-to-end causes an inefficient media flow. The following commands were modified by this feature: **voice-class sip early-offer forced renegotiatle [always]**, **early-offer forced renegotiate [always]**.

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.

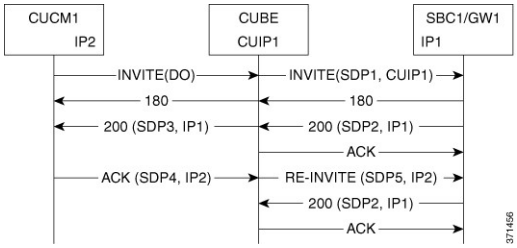
Delayed-Offer to Early-Offer in Media Flow-Around Calls

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 33: Delayed Offer to Early Offer in Media Flow-Around Calls



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

	Command or Action	Purpose
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	Configure conversion of a delayed offer to an early offer: • In dial-peer configuration mode **voice-class sip early-offer forced**	

	Command or Action	Purpose

	• 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 4	**end**	Exits to privileged EXEC mode.

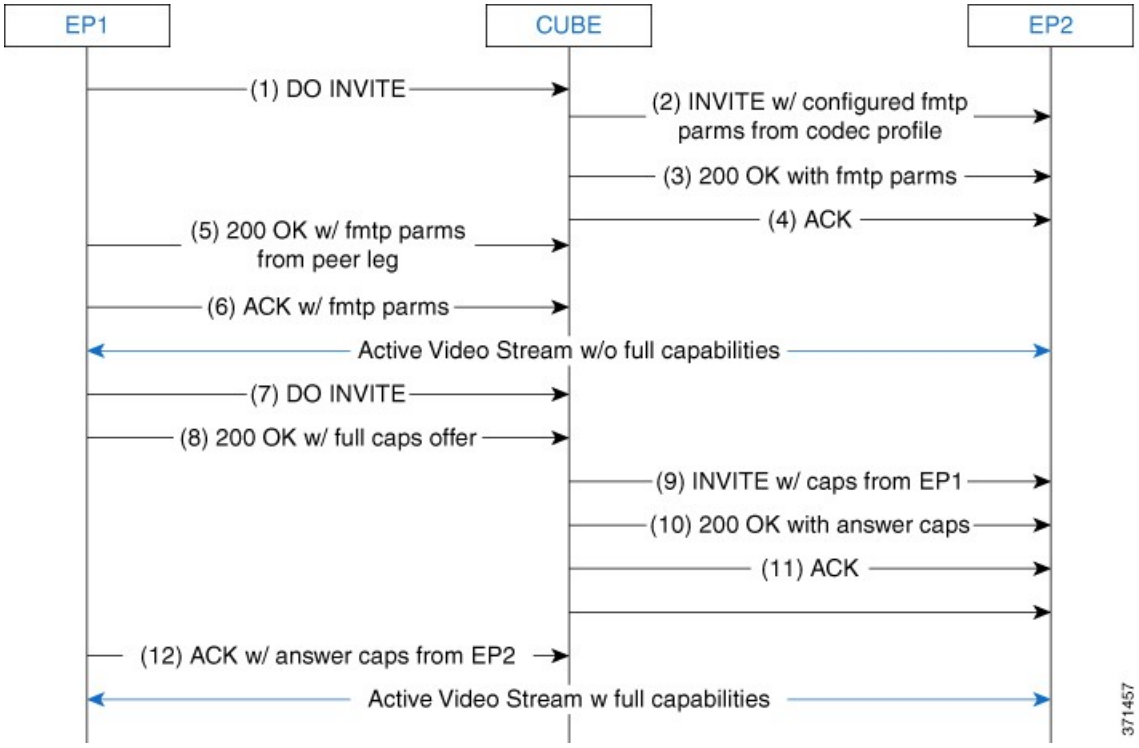
MidCall Renegotiation Support for Delayed-Offer to Early-Offer Calls

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).

Figure 34: MidCall Renegotiation of Capabilities



The **early-offer forced renegotiate** command triggers a delayed-offer RE-INVITE if the negotiated codecs are one of the following:

- aaclld—Audio codec AACLD 90000 bps
- h263—Video codec H263
- h263+—Video codec H263+
- h264—Video codec H264
- mp4a—Wideband audio codec

The **early-offer forced renegotiate always** command always triggers a delayed-offer RE-INVITE. This option can be used to support all other codecs.

Restrictions for MidCall Renegotiation Support for DO-EO Calls

- If **midcall-signaling block** or **midcall-signaling passthru media-change** commands have been configured, the feature does not work because a midcall RE-INVITE is not triggered by CUBE.
- if initial call is transcoded , then midcall re-invite is not triggered by CUBE.

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

	Command or Action	Purpose
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** *id* **voip**	Enters dial-peer configuration mode and configures the selected dial peer.

Step 4	**media transcoder high-density**
	
	Example:
	
	`Device (config) dial-peer voice 10 voip`
	`Device (config-dial-peer) media transcoder high-density`
	`Device (config-dial-peer) end`	

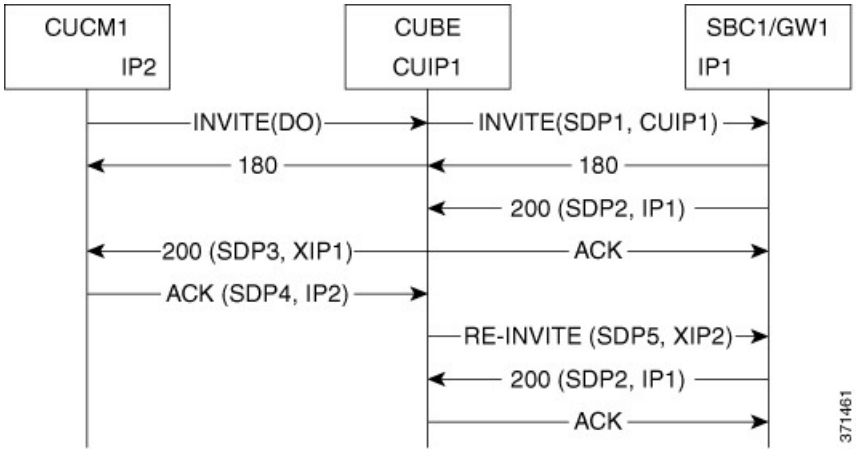
Step 5	**end**	Exits to privileged EXEC mode.

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.

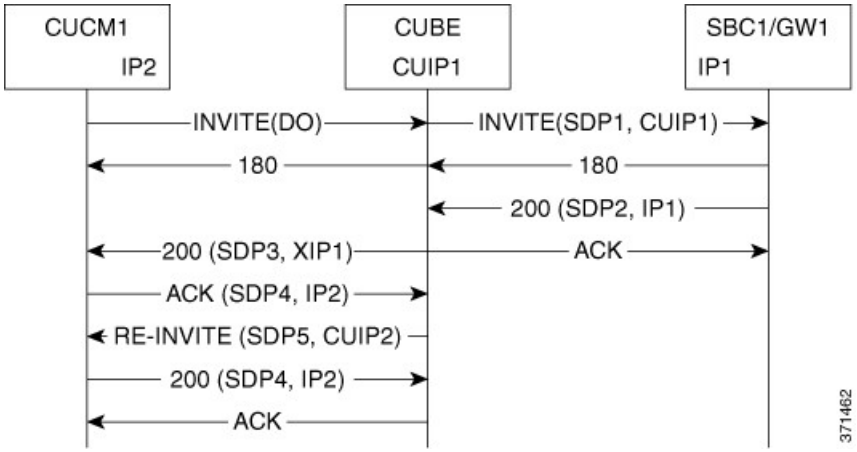
Figure 35: High-Density Transcoding Calls in DO-to-EO



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 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 36: 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

SUMMARY STEPS

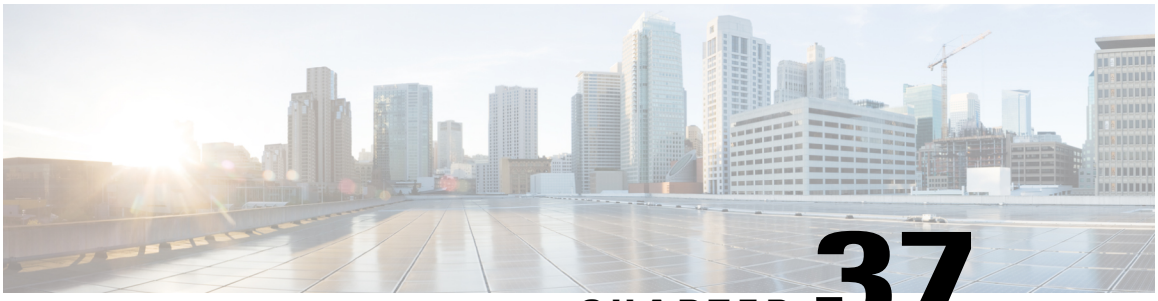
1. **enable**
2. **configure terminal**
3. Configure mid-call renegotiation support for DO-EO calls:
   - **voice-class sip early-offer forced re-negotiate [always]** in dial-peer configuration mode
   - **early-offer forced re-negotiate [always]** in global VoIP configuration mode
4. **end**

DETAILED STEPS

	Command or Action	Purpose
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	Configure mid-call renegotiation support for DO-EO calls:
	• **voice-class sip early-offer forced re-negotiate [always]** in dial-peer configuration mode
	• **early-offer forced re-negotiate [always]** in global VoIP configuration mode
	
	Example:
	In dial-peer configuration mode:
	
	`Device (config) dial-peer voice 10 voip`
	`Device (config-dial-peer) voice-class sip early-offer forced re-negotiate [always]`
	`Device (config-dial-peer) end`
	
	Example:
	In global VoIP SIP mode:
	
	`Device(config)# voice service voip`
	`Device (config-voi-serv) sip`	

	Command or Action	Purpose
	`Device (config-voi-sip) early-offer forced re-negotiate [always]` `Device (config-voi-sip) end`	
Step 4	**end**	Exits to privileged EXEC mode.



CHAPTER 37

H323-to-SIP Inteworking on CUBE

This chapter describes how to configure H.323-to-SIP interworking in CUBE and lists the various features supported in this interworking model.

Prerequisites

- Enable CUBE on the device
- Perform basic H.323 gateway configuration. See Configuring H.323 Gateway (Optional)
- Perform basic H.323 gatekeeper configuration. See Configuring H.323 Gatekeeper (Optional)

Restrictions

- Changing codecs during rotary dial peer selection is not supported.
- Voice class codec is not supported.
- Configure extended capabilities on dial peers for fast start-to-early media scenarios.
- Delayed Offer to Slow-Start is not supported for SRTP-to-SRTP H.323-to-SIP calls.
- During a triggered INVITE scenario the Cisco UBE always generates a delayed offer INVITE.
- Fast-start to delayed-media signal interworking is not supported.
- Fast Start to Early Offer Supplementary Service will not work without extended capabilities configured under dial-peer.

- GSMFR and GSMEFR codecs are not supported.
- Media flow-around is not supported.
- Passing multiple diversion headers or multiple contact header in 302 to the H.323 leg is not supported.
- RSVP for supplementary scenarios is not supported.
- Session refresh is not supported.
- SIP-to-H.323 Supplementary Services based on H.450 is not supported.
- Slow-start to early media signal interworking is not supported.
- Supplementary services are Empty Capability Set (ECS) based supplementary services from the H.323 perspective, not H.450 supplementary services.
- LTI based transcoding is not supported.
- Transcoding for supplementary calls is not supported.
- SCCP based codec transcoding is not support with an exception of Delayed-Offer to Slow-Start with static codec.
- DTMF interworking rtp-nte to inband is supported only with non-high-density transcoding in a delayed-offer to slow-start call.

H323-to-SIP Basic Call Interworking

This feature enables the IP-to-IP gateway to bridge calls between networks that support different VoIP call-signaling protocols (SIP and H.323). The SIP-to-H.323 protocol interworking capabilities of the CUBE support the following:

Feature	Supported Release	Additional Description
Basic voice calls (G.711 and G.729 codecs)	12.4(11)T	
UDP and TCP transport	12.3(11)T	SIP (UDP)<—>H.323 (TCP) SIP (TCP)<—>H.323 (TCP) SIP (UDP)<—>H.323 (UDP) SIP (TCP)<—>H.323 (UDP) Default SIP protocol is UDP. Default H.323 protocol is TCP.

Feature	Supported Release	Additional Description
Interworking between • H.323 Fast-Start and SIP early-media signaling • H.323 Slow-Start and SIP delayed-media signaling	12.3(11)T	• H.323 Fast Start<—>SIP Early Media • H.323 Slow Start <—>SIP Delay Media **Note** No other combinations are supported. For example, H.323 Slow Start <—>SIP Early Media is not supported and results in call failure.
H.323-to-SIP RSVP Support	12.3(11)T	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
DTMF relay interworking:	12.3(11)T 12.4(6)XE	• H.245 alpha/signal<—>SIP Notify • H.245 alpha/signal <—>SIP RFC 2833 • H.245 alpha/signal <—> SIP KPML • G.711 Inband DTMF<—>RFC 2833
Voice call transcoding support	12.3(11)T	• 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

Feature	Supported Release	Additional Description
Calling/called name and number	12.3(11)T	• H.323 IOS FXS/SCCP – IPIPGW – SIP IOS FXS • H.323 IOS FXS/SCCP – IPIPGW – SIP CCME Skinny Phone • H.323 IOS FXS/SCCP – IPIPGW – SIP IP Phone • CCM Phone – IPIPGW – SIP CCME Skinny Phone • CCM Phone – IPIPGW – SIP IP Phone • SIP IOS FXS – IPIPGW – H.323 IOS FXS • SIP IOS FXS – IPIPGW – H.323 CCME Skinny Phone
RADIUS call-accounting records	12.3(11)T	H.323<—>SIP Radius call accounting
TCL IVR 2.0 for SIP, including media playout and digit collection (RFC 2833 DTMF relay)	12.3(11)T 12.4(11)T	12.3(11)T—TCL IVR 2.0 for SIP, including media playout and digit collection (RFC 2833 DTMF relay) 12.4(11)T —TCL IVR support with SIP NOTIFY DTMF
SRTP Passthrough	12.4(15)XY	
Supplementary Services (ECS based).	12.4(11)XJ2	
Codec Transparent	12.4(11)T	
Extended codec support and codec filtering	12.4(11)T	

H323-to-SIP Supplementary Features Interworking

This interworking provides enhanced termination and re-origination of signaling and media between VoIP and Video Networks in conformance with RFC3261.

Feature	Release
Support H.323-to-SIP Supplementary services for CUCM with MTP on the H.323 Trunk.	12.3(11)T
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 45: SIP(OGW)—>IPIPGW—>H323(TGW) calls

SIP at In Leg	H.323 at Out Leg	Comments
183 Session Progress	Progress/Alert PI = 8	Analog phone at TGW
180 Ring	Alert with PI = 0	SCCP phone at TGW

Table 46: H323(OGW)—>IPIPGW—>SIP(TGW) calls

H.323 at In Leg	SIP at Out Leg	Comments
Progress/Alert PI = 8	183 Session Progress	Analog phone at TGW
Alert with PI = 0	180 Ring	SIP/SCCP phone at TGW

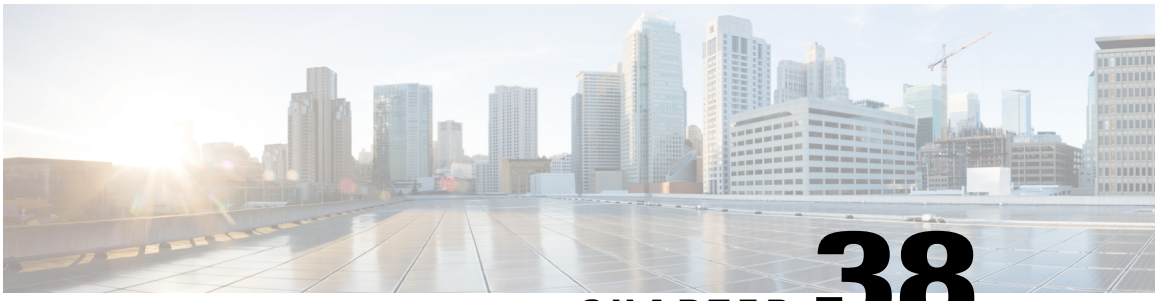
Configuring H323-to-SIP Interworking

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice service voip**
4. **allow-connections h323 to sip**
5. **allow-connections sip to h323**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Router(config)# voice service voip`	Enters Global VoIP configuration mode.
Step 4	**allow-connections h323 to sip** **Example:**	Allows connections from a h323 endpoint to a SIP endpoint.
Step 5	**allow-connections sip to h323** **Example:**	Allows connections from a SIP endpoint to a H.323 endpoint.
Step 6	**end** **Example:** `Router(conf-voi-serv)# end`	Exits to previliged EXEC mode.



CHAPTER 38

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

This chapter describes how to configure and enable features for H.323-to-H.323 connections on CUBE.

Configuring H.323-to-H.323 connections on a CUBE opens all ports by default. If CUBE has a public IP address and a PSTN connection, CUBE becomes vulnerable to malicious attackers who can execute toll fraud across the gateway. To eliminate this threat, you can bind an interface to a private IP address that is inaccessible to untrusted hosts. In addition, you can protect any public or untrusted interface by configuring a firewall or an access control list (ACL) to prevent unwanted traffic from traversing the router.

Feature Information for H.323-to-H.323 Interworking

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 47: Feature Information for H.323-to-H.323 Interworking

Feature Name	Releases	Feature Information
H.323-to-H.323 Connections on a Cisco Unified Border Element	12.3(1)	H.323-to-H.323 Gateway configuration provides a network-to-network demarcation point between independent VoIP and video networks by for billing, security, call-admission control, QoS, and signaling interworking.
Managing H.323 IP Group Call Capacities	12.2(13)T	Creates a maximum capacity for the IP group providing extra control for load and resource balancing.
Overlap Signaling for H.323-to-H.323 Connections on a Cisco Unified Border Element	12.3(11)T	The terminating gateway is responsible for collecting all the called number digits. Overlap signaling is implemented by matching destination patterns on the dial peers.
Rotary Support	12.3(11)T 12.4(6)T	12.3(11)T—H.323-to-H.323 Call Failure Recovery (Rotary) on a Cisco Unified Border Element. Eliminates codec restrictions and enables the Cisco UBE to restart codec negotiation with the originating endpoint based on the codec capabilities of the next dial peer in the rotary group for H.323-to-H.323 interconnections. 12.4(6)T—Secure RTP with IPSEC for Signaling.
Signal Interworking	12.3(11)T	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).

Prerequisites

- Enable CUBE application on a device
- Perform basic H.323 gateway configuration. See Configuring H.323 Gateway
- Perform basic H.323 gatekeeper configuration. See Configuring H.323 Gatekeeper

Restrictions

- Voice class codec is not supported.
- LTI-based transcoding is not supported.
- Supplementary services with transcoding is not supported.
- DTMF Interworking rtp-nte to out of band is not supported when high density transcoder is enabled. Use normal transcoding for rtp-nte to out of band DTMF interworking.
- SCCP based codec transcoding is not supported. An exception to this restriction is slow start to slow start with a static codec.

Slow Start to Fast-Start Interworking

The slow-start to fast-start interworking feature allows two endpoints configured for slow start and fast start respectively to connect with each other through CUBE without dropping the call.

Restrictions for Slow-Start and Fast-Start Interworking

- Slow-start to fast-start interworking is supported only for H.323-to-H.323 calls.
- Transcoding in slow-start to fast-start interworking is not supported.

Enabling Interworking between Slow Start and Fast Start

Configure interworking between slow start and fast start on both inbound and outbound call legs.

Note This task should not be used in situations where fast-start to fast-start or slow-start to slow-start calls are possible.

Before You Begin

Ensure that a codec is configured on incoming and outgoing call legs.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Use one of the following commands to configure interworking between slow start and fast start.
   - **call start interwork** in global VoIP configuration mode
   - **call start interwork** in voice class configuration and applied to inbound and outbound dial peers.
4. **end**

DETAILED STEPS

	Command or Action	Purpose
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	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.
	
	Example:
	In global VoIP configuration mode
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# h323`
	`Device(conf-serv-h323)# call start interwork`
	
	Example:
	In voice class configuration mode
	`!Configuring a Voice class with Fast Start and Slow Start Interworking`
	`Device(config)# voice class h323 10`
	`Device(config-class)# call start interwork`
	
	`!Applying the voice class to a dial peer.`
	`Device(config)# dial-peer voice 20 voip`
	`Device(config-dial-peer)# voice-class h323 10`
		Enables interworking between slow start and fast start.

	Command or Action	Purpose
Step 4	**end**	Exits to privileged EXEC mode.

Call Failure Recovery (Rotary)

Call failure recovery (Rotary) is a feature that provides the flexibility to route a call to a destination with multiple paths based on the policy of a service provider. If one path disconnects the call for any reason (like unreachableDestination, destinationReject, noPermission etc), the call can be routed by choosing another dial peer to the same destination based on configured preference.

Rotary is implemented using the dial peer hunt feature (see Configuring Hunt Groups), and the search for a successful dial peer continues until a **huntstop** command is encountered.

The feature described in this chapter is an enhancement that removes a restriction on codec configuration, that requires for identical codec capabilities configured on all dial peers in a rotary group. This is done by supporting an Empty Capability set (TCS=0) when rotary is configured.

The feature allows the CUBE to restart the codec negotiation process with the originating endpoint based on the codec capabilities of the next dial peer in the rotary group.

Enabling Call Failure Recovery (Rotary) without Identical Codec Configuration

Before You Begin

Configure Call Failure Recovery (Rotary) using dial-peer hunt groups. See Configuring Dial-Peer Hunt Groups.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice service voip**
4. **h323**
5. **emptycapability**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters VoIP voice-service configuration mode.
Step 4	**h323** **Example:** `Device(conf-voi-serv)# h323`	Enters H.323 voice-service configuration mode.
Step 5	**emptycapability** **Example:** `Device(conf-serv-h323)# emptycapability`	Enables call failure recovery (TCS=0) without the need for identical codec configuration.
Step 6	**exit** **Example:** `Router(conf-serv-h323)# exit`	Exits the current mode.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Router(config)# voice service voip`	Enters VoIP voice-service configuration mode.
Step 4	**h323** **Example:** `Router(conf-voi-serv)# h323`	Enters H.323 voice-service configuration mode.
Step 5	ip circuit max-calls *maximum-calls* **Example:** `Router(config-serv-h323)# ip circuit max-calls 1500`	(Required only if reserved calls are to exceed 1000) Sets the maximum number of aggregate H.323 IP circuit carrier call legs. 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.

	Command or Action	Purpose
		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`	(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`	(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(conf-serv-h323)# exit`	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

```
voice service voip
 allow-connections h323 to h323
 h323
  ip circuit max-calls 1000
  ip circuit default only
```

If there is no incoming source carrier ID:

- Capacity only is reported to the gatekeeper using the default circuit (two call legs).
- No source or destination carrier information is reported.

If there is an incoming source carrier ID:

- Two call legs are counted against the default circuit and reported to the GK.
- The source carrier ID is passed through the gateway to the terminating leg.

The following examples show a configuration with more reserved calls than the default value for the **max-calls** argument (1000):

Example: Configuration with Default Calls in Excess of 1000

This example assigns 1100 calls to other carriers, leaving 400 calls available to the default carrier:

```
voice service voip
 allow-connections h323 to h323
 h323
  ip circuit max-calls 1000
  ip circuit carrier-id AA reserved-calls 500
  ip circuit carrier-id bb reserved-calls 500
  ip circuit carrier-id cc reserved-calls 100
```

The following examples show the default carrier configured with an incoming source carrier but no voice source group configured.

Note In this example, 800 call legs are implicitly reserved for the default circuit.

Example: Default Carrier and Incoming Source Carrier with No Voice Source Group

Note A gatekeeper is required with carrier-id routing.

```
voice service voip
 allow-connections h323 to h323
 h323
  ip circuit max-calls 1000
  ip circuit carrier-id AA reserved-calls 200
```

If there is no incoming source carrier ID:

- Capacity only is reported to the GK using the default circuit (two call legs).
- No source or destination carrier information is reported.

If there is an incoming source carrier ID called “AA”:

- One call leg is counted against circuit “AA”.
- One call leg (outbound) is counted against the default circuit.
- The source carrier ID is passed through the gateway to the terminating leg.

If there is an incoming source carrier ID called “BB” (for example) or anything other than “AA”:

- Two call legs are counted against the default circuit.
- The source carrier ID “BB” is passed through the gateway to the terminating leg.

The following examples show the first voice source-group match case:

Example: Voice Source-Group Match Case 1

```
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

```
voice service voip
 allow-connections h323 to h323
 h323
  ip circuit max-calls 1000
  ip circuit carrier-id AA reserved-calls 200
  ip circuit carrier-id BB reserved-calls 200
!
voice source-group 1
 carrier-id source AA
 carrier-id target BB
```

If there is no incoming source carrier ID, the default circuit is used because there is no match in the voice source group.

If there is an incoming source carrier ID called “AA”:

- The voice source-group matches.
- One leg is counted against circuit “AA”.
- One leg is counted against circuit “BB”.
- The source carrier ID is passed through the gateway to the terminating leg.
- The destination carrier ID is “BB”.

The following examples show the third voice source-group match case:

Example: Voice Source-Group Match Case 3

```
voice service voip
 allow-connections h323 to h323
 h323
  ip circuit max-calls 1000
  ip circuit carrier-id AA reserved-calls 200
  ip circuit carrier-id BB reserved-calls 200
!
voice source-group 1
 access-list 1
 carrier-id source BB
```

If the access-list matches, the following apply:

- One leg is counted against circuit “BB”.
- One leg is counted against the default circuit (for the destination circuit).
- The source carrier ID is synthesized to “BB” and used to report to the gatekeeper. It is also used on the outgoing setup.

If a source carrier ID is received on the incoming setup, it is overridden with the synthesized carrier ID

Overlap Signaling

Overlap signaling requires that called digits be sent one-by-one as they are received from the calling device. The first digit is sent in a call setup message and subsequent digits are sent in information messages. This technique is used when a receiving gateway is able to recognize variable-length phone numbers, and requires that the originating gateway signal the end of the call setup process.

Overlap signaling is implemented by matching destination patterns on the dial peers. When H.225 signal overlap is configured on the originating gateway, it sends the SETUP to the terminating gateway once a dial-peer match is found. The originating gateway sends all further digits received from the user to the terminating gateway using INFO messages until it receives a sending complete message from the user. The terminating gateway receives the digits in SETUP and subsequent INFO messages and does a dial-peer match. If a match is found, it sends a SETUP with the collected digits to the PSTN. All subsequent digits are sent to the PSTN using INFO messages to complete the call.

Configuring Overlap Signaling

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice service voip**
4. **h323**
5. **h225 signal overlap**
6. **h225 timeout t302** *seconds*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Router(config)# voice service voip`	Enters VoIP voice-service configuration mode.
Step 4	**h323** **Example:** `Router(conf-voi-serv)# h323`	Enters H.323 voice-service configuration mode.
Step 5	**h225 signal overlap** **Example:** `Router(conf-serv-h323)# h225 signal overlap`	Activates overlap signaling to the destination gateway.
Step 6	**h225 timeout t302** *seconds* **Example:** `Router(conf-serv-h323)# h225 timeout t302 15`	Sets the t302 timer timeout value. The argument is as follows: • *seconds*— Number of seconds for timeouts. Range: 1 to 30.
Step 7	**exit** **Example:** `Router(conf-serv-h323)# exit`	Exits the current mode.

Verifying H.323-to-H.323 Interworking

To verify Cisco Unified Border Element feature configuration and operation, perform the following steps (listed alphabetically) as appropriate.

Note The word “calls” refers to call legs in some commands and output.

SUMMARY STEPS

1. **show call active video**
2. **show call active voice**
3. **show call active fax**
4. **show call history video**
5. **show call history voice**
6. **show call history fax**
7. **show crm**
8. **show dial-peer voice**
9. **show running-config**
10. **show voip rtp connections**

DETAILED STEPS

Step 1 **show call active video**
Use this command to display the active video H.323 call legs.

Step 2 **show call active voice**
Use this command to display call information for voice calls that are in progress.

Step 3 **show call active fax**
Use this command to display the fax transmissions that are in progress.

Step 4 **show call history video**
Use this command to display the history of video H.323 call legs.

Step 5 **show call history voice**
Use this command to display the history of voice call legs.

Step 6 **show call history fax**
Use this command to display the call history table for fax transmissions that are in progress.

Step 7 **show crm**
Use this command to display the carrier ID list or IP circuit utilization.

Step 8 **show dial-peer voice**
Use this command to display information about voice dial peers.

Step 9 **show running-config**
Use this command to verify which H.323-to-H.323, H.323-to-SIP, or SIP-to-SIP connection types are supported.

Step 10 **show voip rtp connections**
Use this command to display active Real-Time Transport Protocol (RTP) connections.

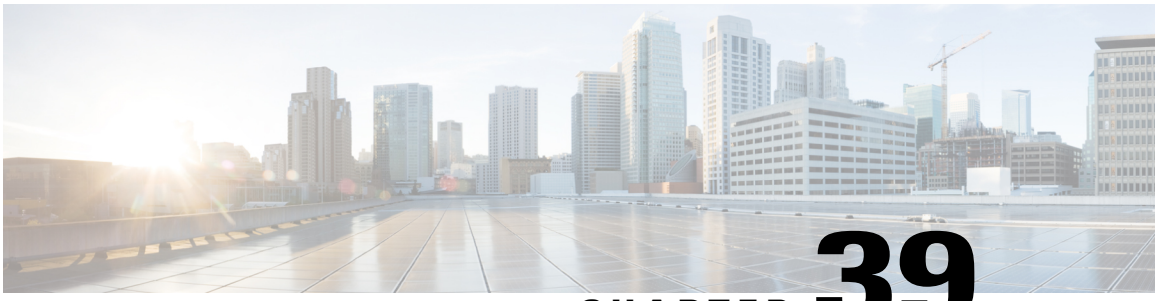
Troubleshooting H.323-to-H.323 Interworking

Caution Under moderate traffic loads, these **debug** commands produce a high volume of output.

- **debug cch323 all**
- **debug h225 asn1**
- **debug h225 events**
- **debug h225 q931**
- **debug h245 asn1**
- **debug h245 events**
- **debug voip ipipgw**
- **debug voip ccapi inout**

PART X

High Availability

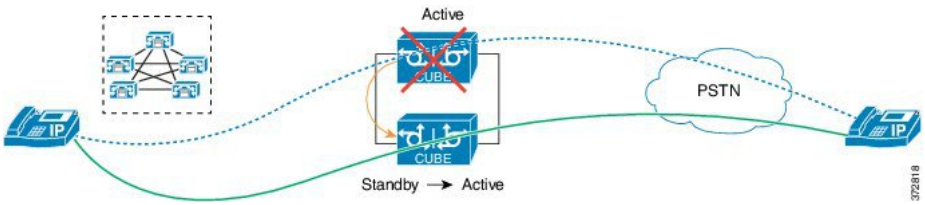


CHAPTER 39

CUBE High Availability Overview

High Availability (HA) is a feature that ensures the availability of resources in a computer system, in case of component failures in the system. The unique hardware and software architecture of the Cisco ASR1000 Series, Cisco ISR G2 and Cisco ISR 4000 Series routers is designed to maximize router uptime during any network event, and thereby provide maximum uptime and resilience within any network scenario.

Figure 37: CUBE High Availability



The diagram above illustrates the CUBE's HA feature that allows active calls to be preserved when a CUBE router experiences an outage.

This chapter provides a brief overview of the different types of high availability (HA) options on the Cisco routers for the CUBE Enterprise edition.

Information About High Availability

Inbox versus Box-to-Box Redundancy

Refer to the next section in this document. For detailed information about inbox and box-to-box redundancies, refer to the chapter titled “Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices” in the *Cisco Unified Border Element Configuration Guide*.

Route Processor Redundancy

Route Processor Redundancy (RPR) allows you to configure a standby RP. When you configure RPR, the standby RP loads the Cisco IOS software on bootup and initializes itself in standby mode. In the event of a fatal error on the active RP, the system switches to the standby RP, which reinitializes itself as the active RP. In this event, the entire system is rebooted, so the switchover with RPR is slower than with other High Availability switchover features such as Nonstop Forwarding/Stateful Switchover (NSF/SSO).

Stateful Switchover

Stateful switchover (SSO) provides media preservation of calls and post-switchover teardown of calls, in case of active RP failure. This means that the CUBE calls would continue to be active even after the active RP card goes down (provided a redundant RP is present). The standby RP would become active and service new CUBE(ENT) calls. The context of the CUBE (ENT) calls that were switched over from the Active card would be present on the new active card. Hold/Resume or any other supplementary services will work after switchover. In SSO, both media and signaling session context are preserved on failover.

Note The terms failover and switchover are used interchangeably.

Nonstop Forwarding

Nonstop forwarding (NSF) helps to suppress routing flaps in devices that are enabled with stateful switchover (SSO), thereby reducing network instability. NSF allows for the forwarding of data packets to continue along known routes while the routing protocol information is being restored after a switchover. Non Stop Forwarding (NSF) works together with SSO and allows the routing protocols to reestablish their routing information by requesting their network neighbors to resend all of the routing information when a switchover occurs.

HA Checkpointing

Checkpointing refers to the facility or architecture to implement stateful switchover (SSO). It provides the mechanisms to help synchronize state data between the active and standby route processors or chassis in a consistent, repeatable, and well-ordered manner.

From Cisco IOS Release 15.6(1)T and Cisco IOS-XE Release 3.17S onwards, checkpointing mechanism is enhanced to provide support for multimedia endpoints with larger SDP up to 6000 bytes. With the new enhancement, CUBE supports preservation of media up to a maximum of 6 m-lines/streams (audio, video, and application).

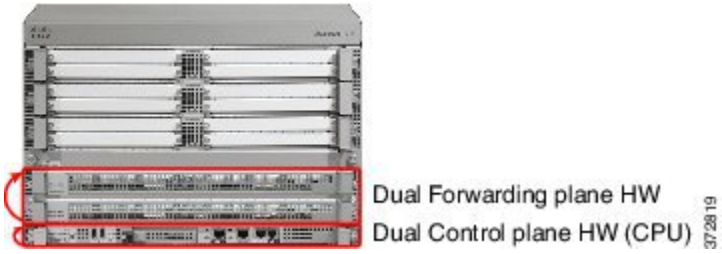
CUBE HA is enhanced to support the preservation of Record-Route and Contact header information. After SSO, all subsequent midcall SIP messages will be routed based on the correct Record-Route and Contact headers.

CUBE High Availability Options

The CUBE HA implementation supports full stateful failover for active SIP-to-SIP calls. This means both media and signaling session information is preserved after switchover. The Cisco Unified Border Element (CUBE) provides three types of high availability (HA) options:

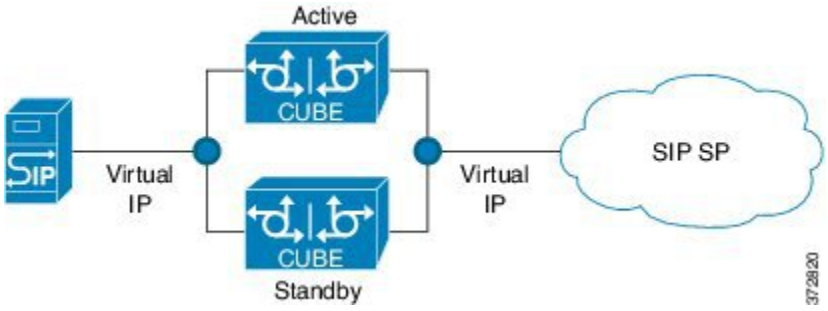
- **Inbox redundancy—**Supported only on ASR devices. Inbox redundancy with Stateful Switchover (SSO) mechanism provides redundancy within the same device. Some models of the ASR offer hardware redundancy within the box and some offer software redundancy.
  - Hardware redundancy—Supports stateful failover from an active Enhanced Services Processor to a standby and from an active route processor to a standby on the same box. Cisco ASR1006 supports this type of failover.
  - Software redundancy—Supports stateful failover from an active IOS process to a standby process, both running on the same route processor. This is different than the platforms running Cisco IOS such as the ISR-G2 devices, where only one process can run on the operating system. Cisco ASR1001/1002/1004 supports this type of failover.

Figure 38: Inbox Redundancy



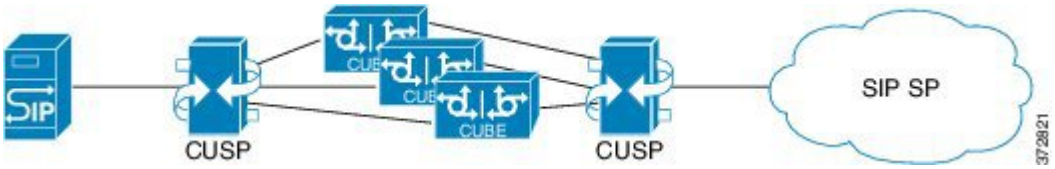
- **L2 box-to-box redundancy (stateful failover of active calls)—**ASR, ISR-G2, and ISR4451 platforms support box-to-box HA. The redundancy protocols used are:
  - ASR—Redundancy Group Infrastructure (RG Infra).
  - ISR-G2—Hot Standby Routing Protocol (HSRP)
  - ISR4451—RG Infrastructure.

Figure 39: L2 Box-to-Box Redundancy



- **Clustering with load balancing—**Clustering is supported on ISR-G2 and ASR devices. Clustering with load balancing provides local and geographical redundancies. Load balancing can be achieved using Cisco Unified SIP Proxy (CUSP) or using a Service Provider (SP) call agent.

Figure 40: Clustering with Load Balancing



Hot Standby Routing Protocol (ISR-G2 Devices)

The Cisco Unified Border Element (CUBE) provides high availability (HA) using box-to-box redundancy configurations when implemented on a Cisco ISR-G2 platform. CUBE box-to-box redundancy on ISR-G2 is based on the Hot Standby Routing Protocol (HSRP) router technology, and HSRP is specific to ISR-G2.

HSRP technology provides high network availability by routing IP traffic from hosts on networks without relying on the availability of any single router. HSRP is used in a group of routers for selecting an Active router and a Standby router. HSRP monitors both the inside and outside interfaces—if any interface goes down, the whole device is considered down, the standby device becomes active and takes over the responsibilities of the active router. Box-to-box high availability is supported using virtual IP addresses for the signaling and media. For more information about HA support using virtual IP addresses, refer to the section titled "Stateful Switchover Between Redundancy Paired Intra- or Inter-box Devices" in the *CUBE Configuration Guide*.

RG Infrastructure (ASR Devices)

On the ASR, box-to-box redundancy option (introduced in the previous section) uses the Redundancy Group (RG) Infrastructure to form an Active/Standby pair of routers. The Active/Standby pair share the same virtual IP address (VIP) and continually exchange status messages. CUBE session information is check-pointed across the Active/Standby pair of routers enabling the Standby router to take over immediately all CUBE call processing responsibilities if the Active router should go out of service. This redundancy option is supported on the ASR 1001/1002-X/1004 platforms, and with Cisco IOS XE 3.11S, it is also supported on ASR 1006 with a single route processor and an Embedded Services Processor (ESP). ASR 1006 supports both box-to-box and inbox redundancy, and you cannot switch between these two modes dynamically.

Box-to-box redundancy requires two identical ASR platforms on the same physical LAN. Redundancy Group (RG) Infra component provides the box-to-box communication infrastructure support between the two ASRs and will negotiate the final stable redundancy state. The RG Infra component provides:

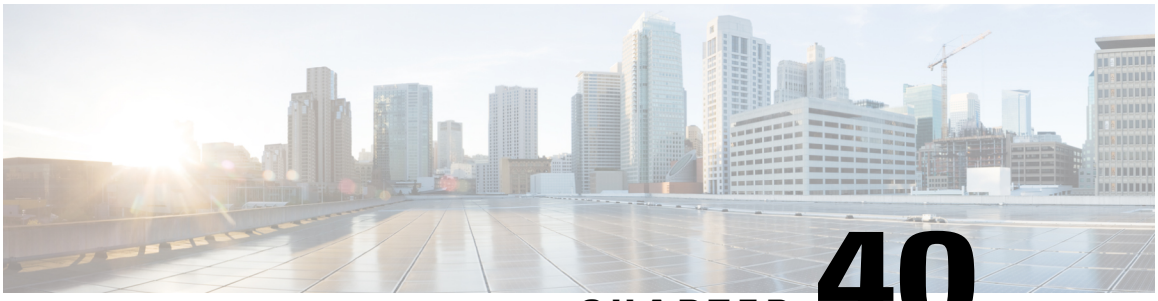
- An HSRP-like protocol that negotiates the final redundancy state for each router by exchanging keepalive and hello messages between the two ASRs in pair (using the control interface).
- A transport mechanism for checkpointing the signaling and media state for each call from the ACTIVE to the STANDBY router (using the data interface).
- Configuration/management of the virtual IP (VIP) interface for the traffic interfaces (multiple traffic interfaces can be configured using the same RG).

Note Licensing implications and configuration details are not covered in this chapter. For information about HA, refer to the High Availability section in the CUBE Advanced guide.

Considerations for Choosing an HA Configuration

When considering HA design, the following VoIP aspects apply:

- Media preservation of active calls
- Calls that are currently being signaled
- Signaling protocol state preservation for active calls (supplementary services will work after switchover)
- Transcoded calls
- Calls using software MTP on the CUBE
- H323-to-SIP and H323-to-H323 calls
- Licensing implications



CHAPTER 40

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

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 48: Feature Information for DSP HA Support on CUBE

Feature Name	Releases	Feature Information
DSP HA Support on CUBE	Cisco IOS 15.5(2)T Cisco IOS XE 3.15S	Provides DSP High availability support for SIP-to-SIP calls on Box-to-Box and Inbox redundancies.

Prerequisites for DSP High Availability

- LTI Transcoding
- DSP HA is supported only on the following routers and its corresponding modules:
  - Cisco ISR G2 series (PVDM3)
  - Cisco ASR 1000 series (SPA-DSP)
  - Cisco ISR 4000 series (PVDM4)
- The same type and capacity DSP modules must be used in the Active and Standby CUBE devices (box-to-box)
- The DSP modules must be installed in the same slot and subslot in the Active and Standby CUBE devices (box-to-box)
- The Active and Standby CUBE devices must have the same DSPFARM configurations (box-to-box)

Features Supported with DSP High Availability

- Transcoding with Supplementary Services
- Voice Class Codec
- G.711 in-band -> RFC2833 (RTP-NTE) DTMF interworking variant
- SRTP-RTP Interworking (ISR-G2 only)
- Fax calls with transcoder invoked for codec mis-match

Restrictions for DSP High Availability

- Media flow-around calls are not supported.
- SDP passthrough calls are not supported.
- Audio Transrating is not supported.
- Call Progress Analysis is not supported.

- Dolby Noise Reduction (NR) and Acoustic Shock Protection (ASP) are not supported.
- Unified Communications Manager initiated transcoding and HW MTP are not supported.

Troubleshooting DSP HA Support on CUBE

You can use the following debug commands to troubleshoot DSP HA:

- debug voip dsmp all
- debug voip dsm all
- debug ccsip message
- debug voip ipipgw
- debug voip ipipgw high-availability
- debug voip high-availability all
- debug media resource provisioning all
- debug dsp-resource-manager flex dspfarm
- debug dsp-resource-manager flex function
- debug dsp-resource-manager flex error

How to Configure High Availability

The Cisco Unified Border Element (CUBE) provides high availability (HA) through box-to-box redundancy configurations when implemented on a Cisco Integrated Services Router Generation 2 router (ISR G2) platform. CUBE box-to-box redundancy is achieved using the router-based Hot Standby Routing Protocol (HSRP) router technology.

To configure DSP HA, follow the general configuration procedure for HA at Cisco Unified Border Element High Availability (HA) Using HSRP Configuration Example.

Configuration Examples for DSP HA

Active Configuration

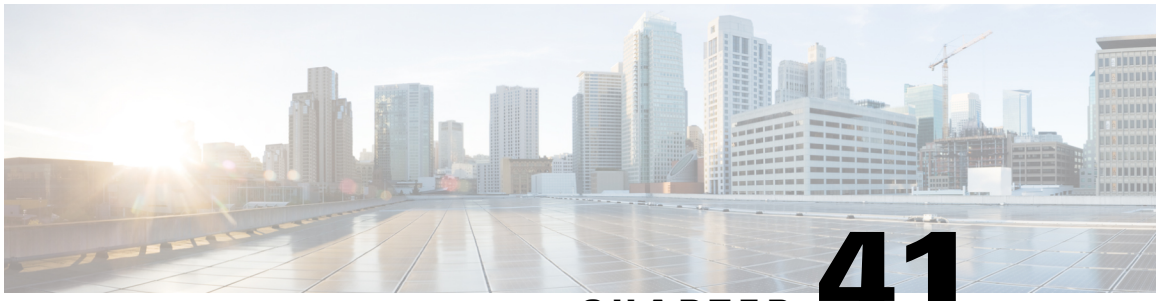
```
----------------
voice-card 0
 dsp services dspfarm

dspfarm profile 2 transcode universal
 codec g711ulaw
 codec g711alaw
 codec g729ar8
 codec g729abr8
 maximum sessions 100
 associate application CUBE
```

Standby Configuration

```
-----------------
voice-card 0
 dsp services dspfarm

dspfarm profile 2 transcode universal
 codec g711ulaw
 codec g711alaw
 codec g729ar8
 codec g729abr8
 maximum sessions 100
 associate application CUBE
```

The following 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
         3005     3007
0            13    39.0.0          UP         1      USED  xcode     1           16559
         3004     3005

Total number of DSPFARM DSP channel(s) 1

On Standby:

Mang-Standby#show dspfarm dsp active
SLOT   DSP VERSION  STATUS CHNL USE   TYPE    RSC_ID BRIDGE_ID PKTS_TXED PKTS_RXED

0           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
```



CHAPTER 41

StatefulSwitchoverBetweenRedundancyPaired 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

Feature Name	Releases	Feature Information
Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices	Cisco IOS XE Release 3.2S	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 Name	Releases	Feature Information
Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices	Cisco IOS Release 15.2(3)T	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.
Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices (Call Escalation and De-escalation with Stateful Switchover)	Cisco IOS XE Release 3.8S 15.3(1)T	Provides support for call escalation and de-escalation with stateful switchover.
Stateful Switchover Between Redundancy Paired Intra or Inter-box Devices (Media Forking with High Availability)	Cisco IOS XE Release 3.8S 15.3(1)T	Provides support for media forking with high availability mechanism.
High Availability Protected Mode and Box-to-Box HA Support	Cisco IOS XE Release 3.11S	Provides support for enabling the PROTECTED mode on a Voice HA-enabled ASR.
OPTIONS PING Support under HA Configuration	15.4(3)M Cisco IOS XE Release 3.13S	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. For configuration examples, see the Examples section about configuring interfaces (ISR and ASR) and configuring SIP binding.

Feature Name	Releases	Feature Information
Support for REFER and BYE/Also after Software Switch-Over	Cisco IOS 15.5(2)T Cisco IOS XE Release 3.15S	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.

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.
- Secure Real-Time Transport Protocol (SRTP)-Real-Time Transport Protocol (RTP) internetworking between one or multiple CUBEs is not supported.
- SRTP passthrough is 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*.

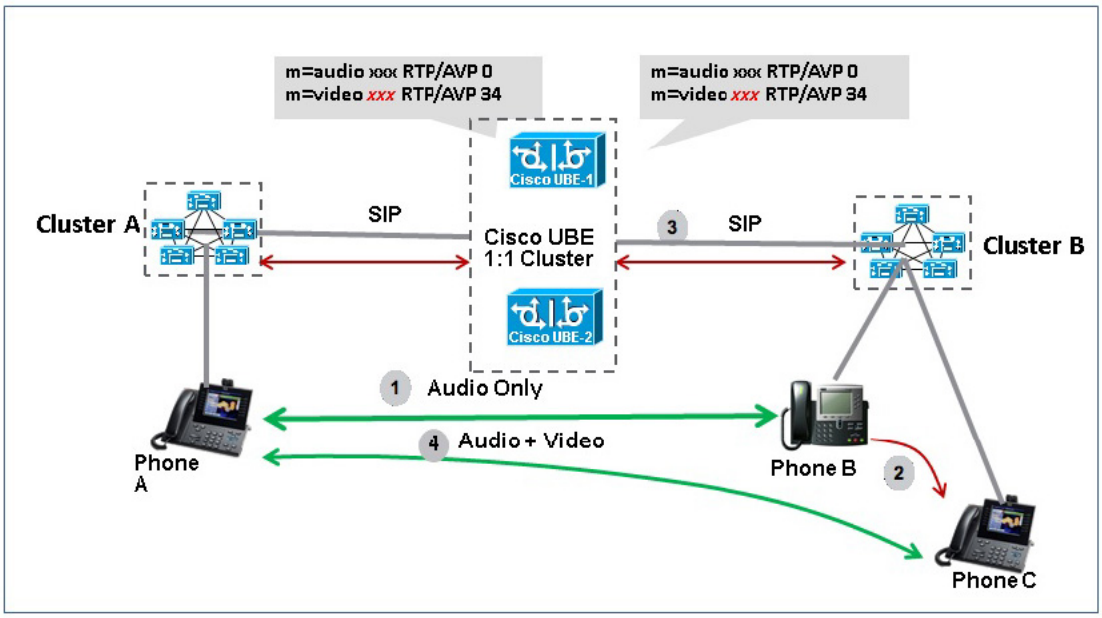
Call Escalation with Stateful Switchover

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.

Figure 41: Call Escalation



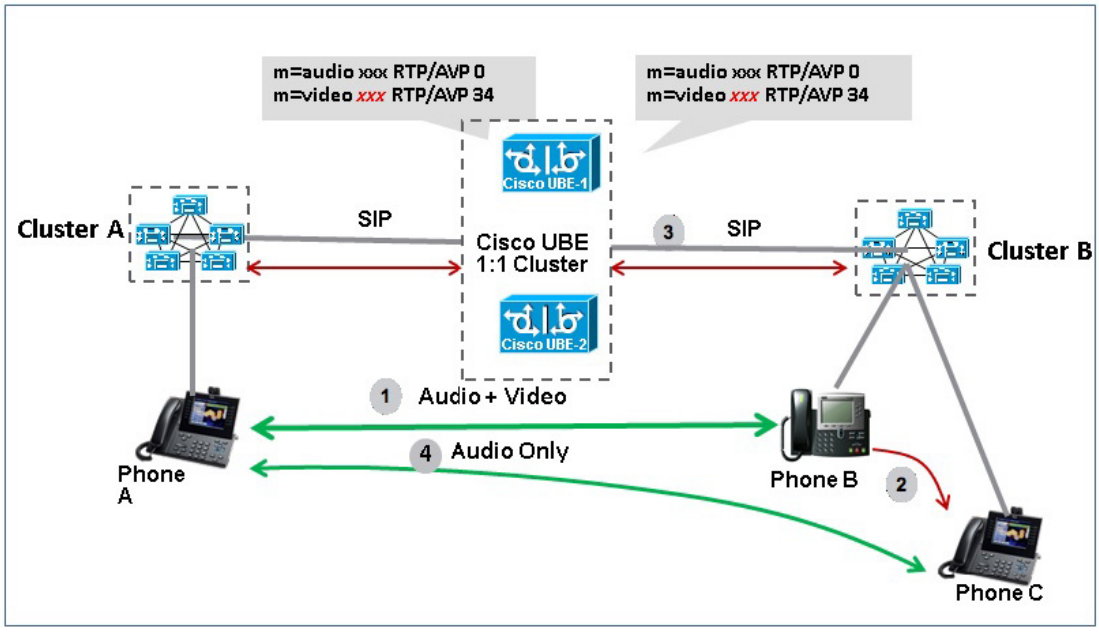
Call De-escalation with Stateful Switchover

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.

Note 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 42: Call De-escalation



Media Forking with High Availability

Media forking with high availability is supported on ISR G2 and ASR platforms. When a primary call is connected and a forked call-leg is established on an active Cisco UBE device, both the primary and the forked call-leg will be checkpointed in the standby Cisco UBE device. If the active device goes down, the standby device ensures that the forking call is active and is able to exchange further transactions with the recording server with preserved calls such as hold/resume, transfer, conference, and so on. A recording server is a Session Initiation Protocol (SIP) user agent that archives media for extended durations, providing search and retrieval of the archived media. The recording server is a storage place of the recorded session metadata.

The active and standby devices must have the same configurations for checkpointing to happen correctly. The recorder can be configured both ways with a media profile and directly on a media class. The media profile can be associated under media class, and the media class can be applied to the incoming or outgoing dial-peer to start recording.

For more information, see the “Network-based Recording Using Cisco UBE” module in the *Cisco Unified Border Element Protocol-Independent Features and Setup Configuration Guide*.

High Availability Protected Mode and Box-to-Box Redundancy for ASR

To configure box-to-box high availability (HA) support for ASRs, use the **mode rpr** command (rpr is route processor redundancy) in **redundancy** configuration mode.

Note

- 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.

Note For configuration examples, see the Examples section about configuring interfaces (ISR and ASR) and configuring SIP binding.

Monitoring Call Escalation and De-escalation with Stateful Switchover

Perform this task to monitor calls before and after escalation or de-escalation and before and after stateful switchover on active and standby Cisco UBE devices. The **show** commands can be entered in any order.

SUMMARY STEPS

1. **enable**
2. **show call active voice compact**
3. **show call active video compact**
4. **show call active voice stats**
5. **show call active video stats**

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode.

Example:

```
Device> enable
```

Step 2 **show call active voice compact**

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

Example:

```
Device# show call active voice compact

<callID>  A/O FAX T<sec> Codec       type        Peer Address       IP R<ip>:<udp>
Total call-legs: 2
       512 ANS     T1     g711ulaw    VOIP        Psipp       9.45.38.39:6016
       513 ORG     T1     g711ulaw    VOIP        P123    10.104.46.222:6000
```

Step 3 **show call active video compact**

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

Example:

```
Device# show call active video compact

<callID>  A/O FAX T<sec> Codec       type        Peer Address       IP R<ip>:<udp>
Total call-legs: 2
       512 ANS     T19    H263        VOIP-VIDEO  Psipp       9.45.38.39:1699
       513 ORG     T19    H263        VOIP-VIDEO  P123    10.104.46.222:1697
```

Step 4 **show call active voice stats**

Displays information about digital signal processing (DSP) voice quality metrics.

Example:

```
Device# show call active voice stats

dur 00:00:16 tx:2238/85044 rx:1618/61484 dscp:0 media:0 audio tos:0xB8 video tos:0x0
IP 9.45.25.33:58300 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:0xB8 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:27352/1039376 rx:36487/1386506 dscp:0 media:0 audio tos:0xB8 video tos:0x88
IP 9.45.25.33:1697 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off Transcoded:
 No
dur 00:00:00 tx:36487/1386506 rx:27352/1039376 dscp:0 media:0 audio tos:0xB8 video tos:0x88
IP 9.45.25.33:1699 SRTP: off rtt:0ms pl:0/0ms lost:0/0/0 delay:0/0/0ms H264 TextRelay: off Transcoded:
 No
```

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
3     4442       4441       16650    36840  10.104.46.251           10.104.46.201
4     4443       4441       16652    54754  10.104.46.251           10.104.46.201
Found 4 active RTP connections
```

Step 4 **show voip recmsp session**

Displays active recording Media Service Provider (MSP) session information. In the output shown, the fork leg details and the number of forking calls are displayed. Both the active and standby devices will have the same call information.

Example:

```
Device# show voip recmsp session

RECMSP active sessions:
MSP Call-ID              AnchorLeg Call-ID        ForkedLeg Call-ID
4441                     4440                     4442
Found 1 active sessions
```

Step 5 **show voip rtp forking**

Displays the RTP media-forking connections. In the output shown, on the active device, packets will be sent.

Example:

```
Device# show voip rtp forking

VoIP RTP active forks :
 Fork 1
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 1
     remote ip 10.104.46.201,  remote port 36840,  local port 16650
       codec g711ulaw,  logical ssrc 0x53
       packets sent 30788,  packets received 0
   stream type voice+dtmf-nearend (4): count 0
   stream type voice-farend (5): count 1
     remote ip 10.104.46.201,  remote port 54754,  local port 16652
       codec g711ulaw,  logical ssrc 0x55
       packets sent 30663,  packets received 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count 0
   stream type application (8): count 0
```

Step 6 **show voip rtp forking**

Displays the RTP media-forking connections. In the output shown, on the standby device, packets will not be sent. After the switchover happens, packets will be sent from the new active device.

Example:

```
Device# show voip rtp forking

VoIP RTP active forks :
 Fork 1
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 1
     remote ip 10.104.46.201,  remote port 36840,  local port 16650
       codec g711ulaw,  logical ssrc 0x53
       packets sent 0,  packets received 0
   stream type voice+dtmf-nearend (4): count 0
   stream type voice-farend (5): count 1
     remote ip 10.104.46.201,  remote port 54754,  local port 16652
       codec g711ulaw,  logical ssrc 0x55
       packets sent 0,  packets received 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count 0
   stream type application (8): count 0
```

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-availablity rf-client** (active device)
3. **show voice high-availablity rf-client** (standby device)

DETAILED STEPS

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

Step 2 **show voice high-availablity rf-client** (active device)

Example:

```
Device# show voice high-availablity 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] :
        [(ACTIVE (13) / PROTECTED (7)]
```

Step 3 **show voice high-availablity rf-client** (standby device)

Example:

```
Device# show voice high-availablity 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 calls handoffs
- Total numbers of unsuccessful call handoffs

The following sample output displays the call handoff statistics:

```
2951-CUBE#show sip-ua handoff stats
Total Calls Handed Off       = 1
Successful Call Hand offs    = 1
Un-Successful Call Hand offs = 0
2951-CUBE#
```

Troubleshooting Tips

Use the following commands to troubleshoot call escalation and de-escalation with stateful switchover:

- **debug voip ccapi all**
- **debug voip ccapi service**
- **debug voice high-availability all**
- **debug voip rtp error**
- **debug voip rtp inout**
- **debug voip rtp high-availability**

- **debug voip rtp function**
- **debug ccsip all**

Use the following commands to troubleshoot media forking support on high availability:

- **debug ccsip all**
- **debug voip high-availability all**
- **debug voip ccapi inout**
- **debug voip recmsp all**

Use the following commands to troubleshoot PROTECTED mode on high availability:

- **debug voice high-availability rf**
- **debug voice high-availability inout**
- **debug redundancy progression**
- **debug redundancy application group faults all**
- **debug redundancy application group protocol all**
- **debug voip ccapi inout**
- **debug cch323 session**
- **debug cch323 function**
- **debug cch323 error**
- **debug 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 ccapi all**
- **debug voice high-availability all**

Example: Configuring the Interfaces for ISR-G2 Devices

ISR-G2 (HSRP-based)

```
interface GigabitEthernet0/0/0
 ip address 10.10.25.14 255.255.255.0
 duplex auto
 keepalive
 speed auto
 standby delay minimum 30 reload 60
 standby version 2
 standby 0 ip 10.10.25.1
  standby 0 preempt
 standby 0 priority 50
```

```
  standby 0 track 2 decrement 10
  standby 0 name SB
```

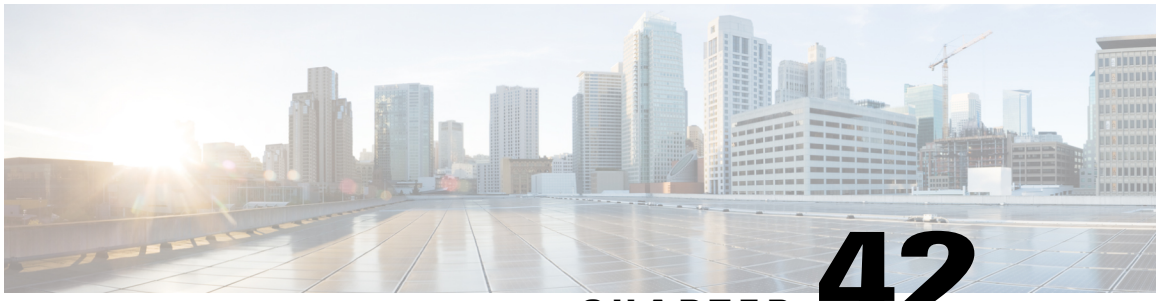
Example: Configuring the Interfaces for ASR Devices

ASR (RG Infra-based)

```
interface GigabitEthernet0/0/0
 ip address 10.13.25.190 255.255.0.0
 negotiation auto
 redundancy rii 1
redundancy group 1 ip 10.13.25.123 exclusive
```

Example: Configuring SIP Binding

```
dial-peer voice inbound-dial-peer-tag voip
  session protocol sipv2
  incoming uri from mydesturi
  voice-class sip call-route url
  voice-class sip bind control source-interface GigabitEthernet 0/0/0
!
voice class uri mydesturi
    host abc.com
```



CHAPTER 42

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

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 49: Feature Information for CVP Survivability TCL support with High Availability

Feature Name	Releases	Feature Information
CVP Survivability TCL support with High Availability	Cisco IOS 15.6(2)T	This feature enables CUBE support call survivability features in CUBE high availability mode for all active calls handled by CVP.

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.

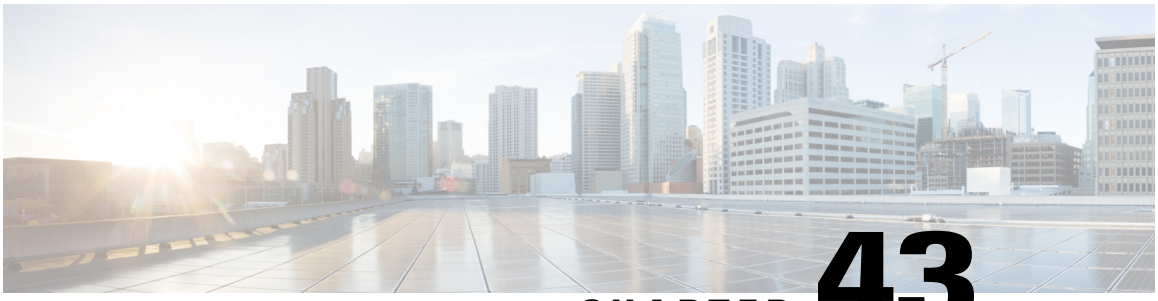
For more information on CVP Call Survivability TCL, refer to http://www.cisco.com/c/dam/en/us/td/docs/voice_ip_comm/cust_contact/contact_center/customer_voice_portal/cvp9_0/configuration/guide/cvp-configuration-and-administration-guide.pdf

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 XI

ICE-Lite Support on CUBE



CHAPTER 43

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

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 ICE-Lite Support on CUBE

Feature Name	Releases	Feature Information
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

	Command or Action	Purpose
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	**voice class stun-usage** *tag* **Example:** `Device(config)# voice class stun-usage 5`	Sets STUN usage global parameters, and enters voice class configuration mode.

	Command or Action	Purpose
Step 4	**stun usage ice lite** **Example:** `Device(config-class)# stun usage ice lite`	Configures ICE in ICE-Lite mode. • You can use the **stun usage ice full** command to configure ICE in ICE-Full mode.
Step 5	**end** **Example:** `Device(config-class)# end`	Returns to privileged EXEC mode.

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:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 20
                                        Min   Max   Ports     Ports     Ports
Media-Address Range                     Port  Port  Available Reserved  In-use
-----------------------------------------------------------------------------
Global Media Pool                       8000  48198 19999     101       20
-----------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId dstCallLocalRTP RmtRTP   LocalIP            RemoteIP          MPSS
1   25     30     8000     2326     10.104.45.107      72.163.212.137    NO
2   26     31     8002     2328     10.104.45.107      72.163.212.137    NO
3   27     32     8036     2454     10.104.45.107      72.163.212.137    NO
4   28     33     8004     2330     10.104.45.107      72.163.212.137    NO
5   29     34     8038     2332     10.104.45.107      72.163.212.137    NO
6   30     25     8006     8016     9.45.46.16         9.45.46.16        NO
7   31     26     8008     8028     9.45.46.16         9.45.46.16        NO
8   32     27     8010     8030     9.45.46.16         9.45.46.16        NO
9   33     28     8012     8032     9.45.46.16         9.45.46.16        NO
10  34     29     8014     8034     9.45.46.16         9.45.46.16        NO
11  35     36     8016     8006     9.45.46.16         9.45.46.16        NO
12  36     35     8018     2326     10.104.45.107      72.163.212.163    NO
13  37     41     8020     2328     10.104.45.107      72.163.212.163    NO
14  38     42     8022     2454     10.104.45.107      72.163.212.163    NO
15  39     43     8024     2330     10.104.45.107      72.163.212.163    NO
16  40     44     8026     2332     10.104.45.107      72.163.212.163    NO
17  41     37     8028     8008     9.45.46.16         9.45.46.16        NO
18  42     38     8030     8010     9.45.46.16         9.45.46.16        NO
19  43     39     8032     8012     9.45.46.16         9.45.46.16        NO
20  44     40     8034     8014     9.45.46.16         9.45.46.16        NO
Found 20 active RTP connections
```

Step 3 **show voip ice instance call-id** *call-id-1*

The following sample output displays the active ICE sessions on the ICE-full and the ICE-lite legs where there are ICE negotiations.

Example:

```
Device# show voip ice instance call-id 25

Interactive Connectivity Check(ICE) Instance details:
Call-ID is 25
Instance is 0x7FC617FC0508
Overall ICE-State is COMPLETED
LocalAgent's mode is ICE-CONTROLLED
RemoteAgent's mode is ICE-CONTROLLING
m-line:1
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107   port 8000  type host         RemoteIP 72.163.212.137  port 2326  type host

m-line:2
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107   port 8002  type host         RemoteIP 72.163.212.137  port 2328  type host
LocalIP 10.104.45.107   port 8003  type host         RemoteIP 72.163.212.137  port 2329  type host

m-line:3
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107   port 8036  type host         RemoteIP 72.163.212.137  port 2454  type host

m-line:4
---------
ICE-State: ACTIVE
NominatedPairs:
```

```
LocalIP 10.104.45.107   port 8004  type host         RemoteIP 72.163.212.137  port 2330  type host
LocalIP 10.104.45.107   port 8005  type host         RemoteIP 72.163.212.137  port 2331  type host

m-line:5
---------
ICE-State: ACTIVE
NominatedPairs:
LocalIP 10.104.45.107   port 8038  type host         RemoteIP 72.163.212.137  port 2332  type host

Total Rx STUN Bind Req 22
Total Tx STUN Bind Succ Resp 22
Total Tx STUN Bind failure resp 0
```

Step 4 **show voip ice instance call-id** *call-id-2*

The following sample output displays the idle ICE sessions on the ICE-lite and the ICE-lite legs where there are no ICE negotiations.

Example:

```
Device# show voip ice instance call-id 30

Interactive Connectivity Check(ICE) Instance details:
Call-ID is 30
Instance is 0x7FC617FC03F8
Overall ICE-State is RUNNING
LocalAgent's mode is ICE-CONTROLLED
RemoteAgent's mode is ICE-CONTROLLING
m-line:1
---------
ICE-State: IDLE
No candidate has been nominated

m-line:2
---------
ICE-State: IDLE
No candidate has been nominated

m-line:3
---------
ICE-State: IDLE
No candidate has been nominated

m-line:4
---------
ICE-State: IDLE
No candidate has been nominated

m-line:5
---------
ICE-State: IDLE
No candidate has been nominated

Total Rx STUN Bind Req 0
Total Tx STUN Bind Succ Resp 0
Total Tx STUN Bind failure resp 0
```

Step 5 **show voip ice summary**

The following sample output displays a summary of active ICE sessions.

Example:

```
Device# show voip ice summary

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:

```
Device# show voip ice global-stats

Interactive Connectivity Establishment(ICE) global stats:
Total Rx Stun BindingRequests        : 43
Total Tx Stun BindingSuccessResponses: 43
Total Tx Stun BindingErrorResponses  : 0
```

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
                                        Min   Max   Ports     Ports     Ports
Media-Address Range                     Port  Port  Available Reserved  In-use
-----------------------------------------------------------------------------
Global Media Pool                       8000  48198 19999     101       2
-----------------------------------------------------------------------------
VoIP RTP active connections :
No. CallId dstCallLocalRTP RmtRTP   LocalIP             RemoteIP          MPSS
1   57     58     8040     7078     10.104.45.107       173.39.64.79      NO
2   58     57     8042     2336     10.104.45.107       72.163.212.163    NO
Found 2 active RTP connections
```

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                : 01CD61B24C077331EDC27A5B
Mapped Address                : Not Set/Present
User Name                     : Not Set/Present
Error code not present
Alternate Server              : Not Set/Present
Realm                         : Not Set/Present
nonce                         : Not Set/Present
Xormapped Address             : Not Set/Present
Server                        : Not Set/Present
ICE Priority                  : Not Set/Present
ICE Controlled                : Not Set/Present
ICE Controlling               : Not Set/Present
Cisco-flowdata                :
cisco-flowdata is not present
Message  Integrity            : Not Set/Present
Finger Print                  : Not Set/Present
###STUN Message structure End###

004013: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/cisco_stun_process_event: Exit
004014: *Aug  8 14:25:30.876 IST: //57/91300134802E/STUN/Inout/cisco_stun_process_event: Entry with
 EventType:7
004015: *Aug  8 14:25:30.876 IST: //57/91300134802E/STUN/Inout/cisco_stun_process_send_msg_event:
Entry
004016: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSendMsg: Entry
004017: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunGetMsgClass: Entry
004018: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunGetMsgClass: en_StunResp
004019: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunSendMsg: dMsgClass:3
004020: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeMsg: Entry
004021: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunCalculateSize:      Length of
ERROR-CODE = 20
004022: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunCalculateSize:      Length of
MESSAGE-INTEGRITY = 24
004023: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsg: STUN Message Length
= 64
004024: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeHdr: Entry
004025: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeHdr: Exit
004026: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeAttr: Entry
004027: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeAttr: Exit
004028: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsgIntegrity:  Original
STUN Message Length = 44
```

```
004029: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsgIntegrity:  Adjusted
STUN Message Length = 44
004030: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsgIntegrity: Successfully
 Encoded MI attribute. Exit
004031: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSetMsgIntegrityToStunMessage:
Entry
004032: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSetMsgIntegrityToStunMessage:
Exit with success
004033: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsg: Total length:64
004034: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeMsg: Exit
004035: *Aug  8 14:25:30.876 IST: //57/91300134802E/STUN/Inout/stunSendMsgToNetwork: Entry
004036: *Aug  8 14:25:30.876 IST: //57/91300134802E/STUN/Detail/stunSendMsgToNetwork: Message sending
 from, 10.104.45.107:8040, to 173.39.64.79:7078
004037: *Aug  8 14:25:30.876 IST: //57/91300134802E/STUN/Detail/stunSendMsgToNetwork: Stun Message:

0111002C2112A44201CD61B24C077331EDC27A5B0009000F00000400426164205265717565737400000800014D0E2E828944BF3D07CC5C06D026D8909B85EF3E9
004038: *Aug  8 14:25:30.876 IST: //57/91300134802E/STUN/Inout/stunSendMsgToNetwork: Exit
004039: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunSendMsg:
** Sent Stun Packet to Network **
###STUN Message structure start###
Message Type                  : STUN_MSG_TYPE_BINDING_ERR_RESP
Magic Cookie                  : 2112A442
Transaction ID                : 01CD61B24C077331EDC27A5B
Mapped Address                : Not Set/Present
User Name                     : Not Set/Present
Error Code                     : Number = 400 ,Reason = Bad Request
Alternate Server              : Not Set/Present
Realm                         : Not Set/Present
nonce                         : Not Set/Present
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###

004041: *Aug  8 14:25:30.876 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunSendMsg: Sent Bind Response,
Free 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:

```
004130: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunGetMsgClass: Entry
004131: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunGetMsgClass: en_StunReq
004132: *Aug  8 14:25:30.912 IST: %CISCO_STUN-4-ICE_ROLE_CONFLICT: Ice Role Conflcit detected in the
 received STUN message on UDP IP address 10.104.45.107 port 8042
004133: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSetErrorCodeToStunMessage: Entry
004134: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunSetErrorCodeToStunMessage:
reason:Role Conflcit, code:487
004135: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSetErrorCodeToStunMessage: Exit
with success
004136: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stun_process_send_bind_response: Exit
004137: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stun_post_bind_request_ind_to_app:
Post Message to Application
004138: *Aug  8 14:25:30.912 IST: //-1/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
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                   : 1862270975
ICE Controlled                 : 11920035603547232620
ICE Controlling                : Not Set/Present
Cisco-flowdata                 :
cisco-flowdata is not present
Message  Integrity             : 0AF4B8C2378CB90AB0B0A3806507D766BF5CD1DD
004139: *Aug  8 14:25:30.912 IST: Finger Print                  : 4235512547
###STUN Message structure End###

004140: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/cisco_stun_process_event: Exit
004141: *Aug  8 14:25:30.912 IST: //58/91300134802E/STUN/Inout/cisco_stun_process_event: Entry with
 EventType:7
004142: *Aug  8 14:25:30.912 IST: //58/91300134802E/STUN/Inout/cisco_stun_process_send_msg_event:
Entry
004143: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSendMsg: Entry
004144: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunGetMsgClass: Entry
004145: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunGetMsgClass: en_StunResp
004146: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunSendMsg: dMsgClass:3
004147: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeMsg: Entry
004148: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunCalculateSize:      Length of
ERROR-CODE = 24
004149: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunCalculateSize:      Length of
MESSAGE-INTEGRITY = 24
004150: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsg: STUN Message Length
= 68
004151: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeHdr: Entry
004152: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeHdr: Exit
004153: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeAttr: Entry
004154: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeAttr: Exit
004155: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsgIntegrity:  Original
STUN Message Length = 48
004156: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsgIntegrity:  Adjusted
STUN Message Length = 48
004157: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsgIntegrity: Successfully
 Encoded MI attribute. Exit
004158: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSetMsgIntegrityToStunMessage:
Entry
004159: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunSetMsgIntegrityToStunMessage:
Exit with success
004160: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunEncodeMsg: Total length:68
004161: *Aug  8 14:25:30.912 IST: //-1/xxxxxxxxxxxx/STUN/Inout/stunEncodeMsg: Exit
004162: *Aug  8 14:25:30.912 IST: //58/91300134802E/STUN/Inout/stunSendMsgToNetwork: Entry
004163: *Aug  8 14:25:30.912 IST: //58/91300134802E/STUN/Detail/stunSendMsgToNetwork: Message sending
 from, 10.104.45.107:8042, to 72.163.212.163:2336
004164: *Aug  8 14:25:30.912 IST: //58/91300134802E/STUN/Detail/stunSendMsgToNetwork: Stun Message:

0111003021 12A442F1CF84958CE76D15C83059D900090011000004 57526F6C6520436F6E666C6369740000000008001413402FC99C6029653902630573977347 6578806E
004165: *Aug  8 14:25:30.913 IST: //58/91300134802E/STUN/Inout/stunSendMsgToNetwork: Exit
004166: *Aug  8 14:25:30.913 IST: //-1/xxxxxxxxxxxx/STUN/Detail/stunSendMsg:
** Sent Stun Packet to Network **
###STUN Message structure start###
Message Type                   : STUN_MSG_TYPE_BINDING_ERR_RESP
Magic Cookie                   : 2112A442
Transaction ID                 : F1CF84958CE76D15C83059D9
Mapped Address                 : Not Set/Present
User Name                      : Not Set/Present
Error Code                      : Number = 487 ,Reason = Role Conflcit
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             : 13402FC99C6029653902630573977347 6578806E
```

```
004167: *Aug  8 14:25:30.913 IST: Finger Print                  : Not Set/Present
###STUN Message structure End###

004168: *Aug  8 14:25:30.913 IST: //-1/xxxxxxxxxxx/STUN/Detail/stunSendMsg: Sent Bind Response,
Free the transaction
004169: *Aug  8 14:25:30.913 IST: //58/91300134802E/STUN/Detail/cisco_stun_process_send_msg_event:
STUN message Sent
```

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

Standard/RFC	Title
RFC 5389	*Session Traversal Utilities for NAT (STUN)*
RFC 5245	*Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*

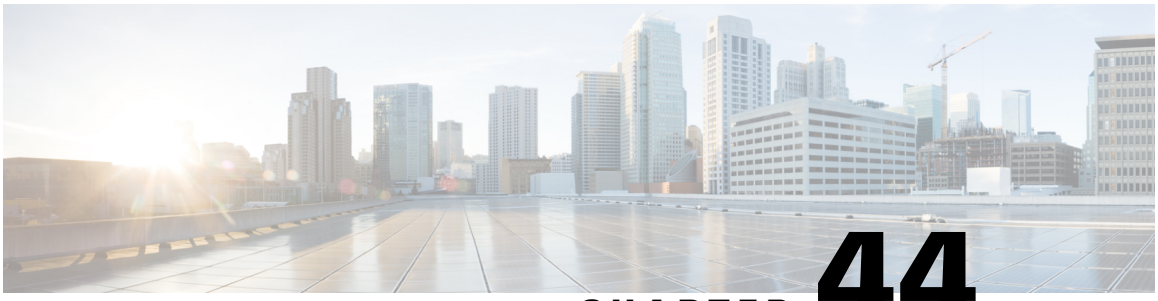
Standard/RFC	Title
RFC 5766	*Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*
RFC 5768	*Indicating Support for Interactive Connectivity Establishment (ICE) in the Session Initiation Protocol (SIP)*
RFC 3840	*Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)*
RFC 7584	*Session Traversal Utilities for NAT (STUN) Message Handling for SIP Back-to-Back User Agents (B2BUAs)*

Technical Assistance

Description	Link
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.	http://www.cisco.com/support

PART XII

SIP Protocol Handling



CHAPTER 44

Mid-call Signaling Consumption

The Cisco Unified Border Element BE Mid-call Signaling support aims to reduce the interoperability issues that arise due to consuming mid-call RE-INVITES/UPDATES.

Mid-call Re-INVITEs/UPDATEs can be consumed in the following ways:

- Mid-call Signaling Passthrough - Media Change
- Mid-call Signaling Block
- Mid-call Signaling Codec Preservation

Note This feature should be used as a last resort only when there is no other option in CUBE. This is because configuring this feature can break video-related features. For Delay-offer Re-INVITE, the configured codec will be passed as an offer in 200 message to change the codec, the transcoder is added in the answer.

Feature Information for Mid-call Signaling

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

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

Table 51: Feature Information for Mid-call Signaling

Feature Name	Releases	Feature Information
Mid-call Re-INVITE Consumption	Cisco IOS 15.2(1)T Cisco IOS XE 3.6S	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**.
Mid-call Codec Preservation	Cisco IOS 15.3(2)T Cisco IOS XE 3.9S	The Mid-call Codec Preservation feature helps to disables codec negotiation in the middle of a call and preserves the codec negotiated before the call. The following commands were introduced or modified: **midcall-signaling preserve-codec** , **voice-class sip midcall-signaling preserve-codec** .
Mid-call Re-INVITE Consumption Enhancements	Cisco IOS 15.5(3)M Cisco IOS XE 3.16S	Mid-call signaling Re-INVITE consumption is enhanced to support: • Re-INVITE based call transfer • Call transfer with REFER Consume • Normalization of call hold in a call set-up

Prerequisites

- Enable CUBE application on a device
- Cisco IOS Release 15.2(1)T or later, or Cisco IOS-XE Release 15.2(2)S or later must be installed.
- **supplementary-service media-renegotiate** must be configured in global voice service voip mode.

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.
- Video transcoding is not supported.
- Secure Real-time Protocol - Real-time Protocol (SRTP-RTP) supplementary services are not supported.
- Multicast Music On Hold (MMOH) is not supported.
- When the **midcall-signaling passthru media-change** command is configured and high-density transcoder is enabled, there might be some impact on Digital Signal Processing (DSP) resources as the transcoder might be used for all the calls.
- Session timer is handled leg by leg whenever this feature is configured.
- More than two m-lines in the SDP is not supported.
- Alternative Network Address Types (ANAT) is not supported.
- Video calls and Application streams are not supported when mid-call signaling block is configured.

Behavior of Mid-call Re-INVITE Consumption

- If mid-call signaling block is enabled on either of call-legs, video parameters and application streams are not negotiated, and are rejected in the answer.
- When flow around and offer-all is configured, CUBE performs codec renegotiation even if mid-call signaling block is configured globally.
- Below behavior is for refer consume scenario:
    - REFER consume is supported for blind, alert and consult call transfers.
    - Existing codecs or DTMF is used for local bridging of new call legs. No Re-INVITE or UPDATE is sent for media re-negotiation after REFER.
    - Call gets dropped when DSP is required but not available.

- A call can be escalated to video only if transferee and transfer-to dial-peers do not have mid-call signaling block configured.
- Video calls are de-escalated if mid-call signaling block configuration on transfer-to dial-peer.
- For Re-INVITE based call-transfer involving Cisco Unified Communications Manager, all Re-INVITE are locally answered and transcoder is invoked if negotiated codecs are different than the codecs before call-transfer.

- The following table provides the details of the behavior when the initial call is establish without 'sendrecv' parameter, that means, the initial call is established with 'sendonly', 'recvonly' or 'inactive'.

Scenario	Behavior
If an Offer is received with 'sendonly' and mid-call block is configured on any or both call legs	Offer is sent with 'sendrecv'.
If an Answer is received with 'sendonly' and the peer leg supports mid-call signaling	Answer is sent with 'sendonly'. Resume transaction is end-to-end.
If an Answer is received with 'sendonly' and the peer leg does not supports mid-call signaling	Answer is sent with 'sendrecv'. Resume transaction is consumed.
If Offer as well as Answer is received with 'sendonly' and Offering leg does not support mid-call signaling	Answer is sent with 'recvonly'. Resume from Offering leg is end-to-end. Resume from answering leg is consumed.
If Offer as well as Answer is received with 'sendonly' and Answering leg does not support mid-call signaling	Answer is sent with 'inactive'. Resume from Offering leg is consumed. Resume from answering leg is end-to-end.
If Offer as well as Answer is received with 'sendonly' and both legs do not support mid-call signaling	Answer is sent with ' recvonly'. Resume transaction is consumed.

Configuring Passthrough of Mid-call Signalling

Perform this task to configure passthrough of mid-call signaling (as Re-invites) only when bidirectional media is added.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Configure passthrough of mid-call signaling changes only when bidirectional media is added.
   - In Global VoIP SIP configuration mode

     midcall-signaling passthru media-change
   - In dial-peer configuration mode

     voice-class sip mid-call signaling passthru media-change
4. **end**

DETAILED STEPS

	Command or Action	Purpose
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	Configure passthrough of mid-call signaling changes only when bidirectional media is added.
	• In Global VoIP SIP configuration mode
	midcall-signaling passthru media-change
	• In dial-peer configuration mode
	voice-class sip mid-call signaling passthru media-change
	
	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 mid-call signaling passthru media-change`
		Re-Invites are passed through only when bidirectional media is added.

	Command or Action	Purpose
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 mid-call 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(conf-voi-serv)# no ip address trusted authenticate
Device(conf-voi-serv)# allow-connections sip to sip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# midcall-signaling passthru media-change
```

Mid-call Signaling Block

The Block method blocks all mid-call media-related signaling to the specific SIP trunk. The command **midcall-signaling block** needs to be configured to enable this behavior. Video escalation and T.38 call flow are rejected when the **midcall-signaling block** command is configured. This command should be configured only when basic call is the focus and mid-call can be consumed.

Restrictions for Mid-Call Signaling Block

- SIP-H.323 calls are not supported.
- TDM Gateways are not supported.
- Session Description Protocol (SDP) -passthrough is not supported
- Video calls and Application streams are not supported.
- When media flow-around is configured, Mid-call INVITE is rejected with 488 error message.
- Re-invites are not consumed if media anti-tromboning is configured.
- SRTP-RTP supplementary services are not supported.

- Multicast Music On Hold (MMOH) is not supported.
- When the **midcall-signaling passthru media-change** command is configured and high-density 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.

Blocking Mid-Call Signaling

Perform this task to block mid-call signaling:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Configure blocking of mid-call signaling changes:
   - In Global VoIP SIP configuration mode

     midcall-signaling block
   - In dial-peer configuration mode

     voice-class sip mid-call signaling block
4. **end**

DETAILED STEPS

	Command or Action	Purpose
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	Configure blocking of mid-call signaling changes: • In Global VoIP SIP configuration mode **midcall-signaling block** • In dial-peer configuration mode **voice-class sip mid-call signaling block**	Mid-call signaling is always blocked.

	Command or Action	Purpose

	Example:
	In Global VoIP SIP configuration mode:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# sip`
	`Device(conf-serv-sip)# midcall-signaling block`
	
	Example:
	In Dial-peer configuration mode:
	`Device(config)# dial-peer voice 2 voip`
	`Device(config-dial-peer)# voice-class sip mid-call signaling block`	

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

Example Blocking SIP Messages at Dial Peer Level

```
dial-peer voice 107 voip
 destination-pattern 74000
 session protocol sipv2
 session target ipv4:9.45.36.9
 incoming called-number 84000
 voice-class codec 1 offer-all
!
dial-peer voice 110 voip
 destination-pattern 84000
 session protocol sipv2
 session target ipv4:9.45.35.2
 incoming called-number 74000
 voice-class codec 1 offer-all
 voice-class sip mid-call signaling block
!
```

Example: Blocking SIP Messages at the Global Level

The following example shows how to block SIP messages at the global Level

```
Device(config)#voice service voip
Device(config-voi-serv)#no ip address trusted authenticate
Device(config-voi-serv)#allow-connections sip to sip
Device(config-voi-serv)#sip
Device(config-serv-sip)#midcall-signaling block
```

Mid Call Codec Preservation

Mid call codec preservation defines whether a codec can be negotiated after a call has been initiated. You can enable or disable codec negotiation in the middle of a call.

Configuring Mid Call Codec Preservation

This tasks 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

	Command or Action	Purpose
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 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
	
	Example:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# sip`
	`Device(conf-serv-sip)# midcall-signaling preserve-codec`
		Disables codec negotiation in the middle of a call and preserves the codec negotiated before the call.

	Command or Action	Purpose

	Example:
	`Device(config)# dial-peer voice 10 voip`
	`Device(conf-dial-peer)# voice-class sip midcall-signaling preserve-codec`	

Step 4	**end** **Example:** `Device(conf-serv-sip)# end`	Exits to privileged EXEC mode.

Example: Configuring Mid Call Codec Preservation at the Dial Peer Level

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

Example: Configuring Mid Call Codec Preservation at the Global Level

```
Device(config)# voice service voip
Device(conf-voi-serv)# no ip address trusted authenticate
Device(conf-voi-serv)# allow-connections sip to sip
Device(conf-voi-serv)# sip
Device(conf-serv-sip)# midcall-signaling preserve-codec
```

CHAPTER 45

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

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 52: Feature Information for Mid-call Signaling

Feature Name	Releases	Feature Information
Early Dialog UPDATE Block	Cisco IOS 15.5(3)M Cisco IOS XE 3.16S	This feature allows CUBE to consume the UPDATE requests with SDP received during an early dialog. The following command is introduced: **early-media update block**.

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

	Command or Action	Purpose
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 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
	
	Example:
	In dial-peer configuration mode
	`!Applying Early Dialog UPDATE block to one dial peer only`
	`Device (config)# dial-peer voice 10 voip`
	`Device (config-dial-peer)# Voice-class sip early-media update block`
	`Device (config-dial-peer)# end`
	
	Example:
	In global VoIP SIP configuration mode
	`! Applying Early Dialog UPDATE block globally`
	`Device(config)# voice service voip`
	`Device (config-voi-serv)# sip`
	`Device (config-voi-sip)# early media update block`
	`Device (config-voi-sip)# end`	

	Command or Action	Purpose
Step 4	**end**	Exits VoIP SIP configuration mode and enters privileged EXEC mode.

Configuring Early Dialog UPDATE Block Renegotiate

Configuring Early Dialog UPDATE Block Renegotiate enables CUBE to renegotiate the call if UPDATE request with SDP contains changes caller-ID, transcoder insertion or deletion, or video escalation or deletion. CUBE renegotiates by sending a DO invite after the call is established.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Enter one of the following commands:
   - In the dial-peer configuration mode

     voice-class sip early-media update block re-negotiate
   - In the global VoIP configuration mode

     early media update block re-negotiate
4. **end**

DETAILED STEPS

	Command or Action	Purpose
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:**	Renegotiates the call if the UPDATE request contains changes in caller ID, transcoder addition or deletion, or video escalation or de-escalation.

	Command or Action	Purpose

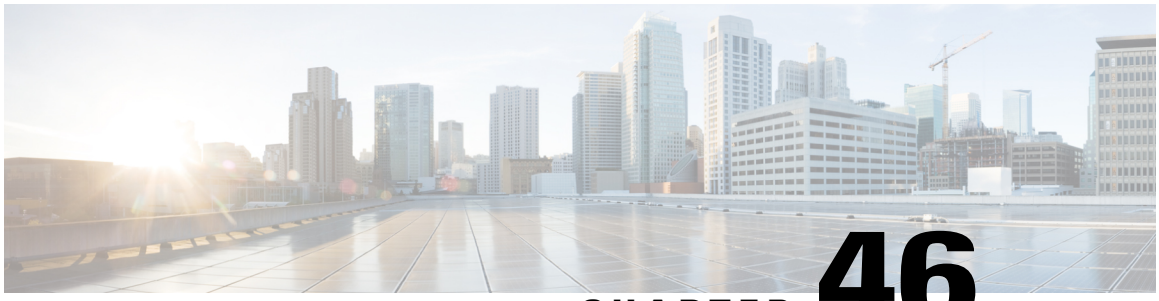
	In dial-peer configuration mode
	
	`!Applying Early Dialog UPDATE block re-negotiate to one dial peer only`
	`Device (config)# dial-peer voice 10 voip`
	`Device (config-dial-peer)# voice-class sip early-media update block re-negotiate`
	`Device (config-dial-peer)# end`
	
	Example:
	In global VoIP SIP 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 all**
- **debug voip ccapi inout**
- **show voip rtp connections**



CHAPTER 46

Support for Pass-Through of Unsupported Content Types in SIP INFO Messages

This feature allows the CUBE to pass-through all unsupported content types in a SIP INFO message.

Feature Information for Support for Pass-Through of Unsupported Content Types in SIP INFO Messages

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 Name	Releases	Feature Information
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.

Prerequisites

You must enable the **pass-thru content unsupp** command to pass-through all unsupported content types in a SIP INFO message. There is no additional configuration task required for this feature.

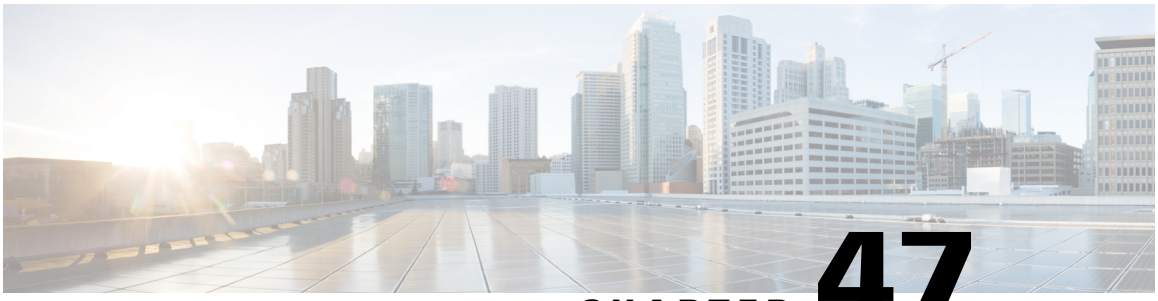
Information About Pass-Through of Unsupported Content Types in SIP INFO Messages

The Support for Pass-Through of Unsupported Content Types in SIP INFO Messages feature allows the CUBE to pass-through all unsupported content types in a SIP INFO message.

Upon receipt of a SIP INFO message with unsupported content type, CUBE triggers a SIP INFO message on the outgoing peer call leg. The response received for this SIP INFO message is triggered on the incoming peer call leg and information flows end-to-end. Prior to releases 15.5(3)M and 3.16S, on reception of SIP INFO message with unsupported content type, CUBE will respond with the "415 Unsupported Media Type" error response.

Supported content types include the following:

- application/sdp
- application/qsig
- application/media-control+xml
- application/x-q931
- application/gtd
- application/simple-message-summary
- application/kpml-response+xml
- application/dtmf-relay
- application/broadsoft
- message/sipfrag
- audio/telephone-event
- multpart/mixed

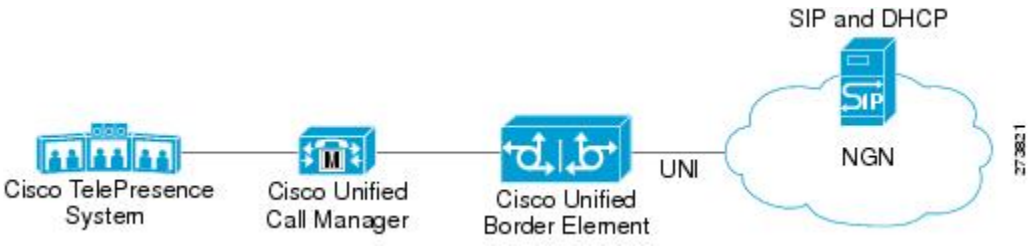


CHAPTER 47

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 43: 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 us 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.

Note If the NGN sets the From: value to anonymous, the PAID is the only value that identifies the caller.

The table below describes the types of INVITE message header translations supported by the Cisco Unified Border Element. It also includes information on the configuration commands to use to configure P-header translations.

The table below shows the P-header translation configuration settings only. In addition to configuring these settings, you must configure other system settings (such as the session protocol).

Table 53: P-header Configuration Settings

Incoming Header	Outgoing Header	Configuration Notes
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.

Incoming Header	Outgoing Header	Configuration Notes
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.
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**

Incoming Header	Outgoing Header	Configuration Notes
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 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**
RPID	PAID	To enable the translation to PAID privacy headers in the outgoing header at a global level, use the **asserted-id pai** command in voice service VoIP SIP configuration mode. For example: Router(conf-serv-sip)# **asserted-id pai** To enable the translation to PAID privacy headers in the outgoing header on a specific dial peer, use the **voice-class sip asserted-id pai** command in dial peer voice configuration mode. For example: Router(config-dial-peer)# **voice-class sip asserted-id pai**

Incoming Header	Outgoing Header	Configuration Notes
RPID	From:	By default, the translation to RPID headers is enabled and the system translates PPID headers in incoming messages to RPID headers in the outgoing messages. To disable the default behavior and enable the translation from PPID 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**

The CUBE can be configured to transparently pass the PAID and PPID 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

Note The priority of P-headers are in the following order: PAID, PPID, and RPID.

Table 54: PAID and PPID header configuration settings for mid-call requests and responses

Incoming Header	Outgoing Header	Configuration Notes
PAID	PPID	To enable the translation to PPID 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 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**

Incoming Header	Outgoing Header	Configuration Notes
RPID	PPID	To enable the translation to PPID 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 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**
PPID	PPID	To enable the translation to PPID headers in the outgoing header at a global level, use the **asserted-id ppi** 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 **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	PAID	To enable the translation to PAID headers in the outgoing header at a global level, use the **asserted-id pai** 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 **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**

Incoming Header	Outgoing Header	Configuration Notes
RPID	PAID	To enable the translation to PAID headers in the outgoing header at a global level, use the **asserted-id pai** 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 **voice-class sip asserted-id pai** command in dial peer voice configuration mode.
PPID	PAID	To enable the translation to PAID headers in the outgoing header at a global level, use the **asserted-id pai** 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 **voice-class sip asserted-id pai** command in dial peer voice configuration mode.
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**. **Note** PAID and PPID headers are not configured in this case.
RPID	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**. **Note** PAID and PPID headers are not configured in this case.

Incoming Header	Outgoing Header	Configuration Notes
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**
FROM	FROM	No configuration required except for the **remote-party-id** header.
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**
PAID	PAID	Enables PPID headers on the incoming dial-peer and PAID headers on the outgoing dial-peer.
RPID	PAID	Enables PPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.
PPID	PAID	Enables PPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.
PAID	PAID	Enables RPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.
RPID	PAID	Enables RPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.
PPID	PAID	Enables RPID headers on incoming dial-peer and PAID headers on outgoing dial-peer.
PAID	PPID	Enables PAID headers on incoming dial-peer and PPID headers on outgoing dial-peer.

Incoming Header	Outgoing Header	Configuration Notes
RPID	PPID	Enables PAID headers on incoming dial-peer and PPID headers on outgoing dial-peer.
PPID	PPID	Enables PAID headers on incoming dial-peer and PPID on outgoing dial-peer.
PAID	PPID	Enables RPID headers on incoming dial-peer and PPID headers on outgoing dial-peer.
RPID	PPID	Enables RPID headers on incoming dial-peer and PPID headers on outgoing dial-peer.
PPID	PPID	Enables RPID headers on incoming dial-peer and PPID headers on outgoing dial-peer.
PAID	RPID	Enables PPID headers on incoming dial-peer and RPID headers on outgoing dial-peer. **Note** PAID headers will be given priority and RPID headers will be created using the PAID header information.
RPID	RPID	Enables PPID headers on incoming dial-peer and RPID headers on outgoing dial-peer.
PPID	RPID	Enables PPID headers on incoming dial-peer and RPID headers on outgoing dial-peer. **Note** PPID headers will be given priority and RPID headers will be created using the PPID header information.

Incoming Header	Outgoing Header	Configuration Notes
PAID	RPID	Enables PAID headers on incoming dial-peer and RPID headers on outgoing dial-peer. **Note** PAID headers will be given priority and RPID headers will be created using the PAID header information.
RPID	RPID	Enables PAID headers on incoming dial-peer and RPID headers on outgoing dial-peer.
PPID	RPID	Enables PAID headers on incoming dial-peer and RPID headers on outgoing dial-peer. **Note** PPID headers will be given priority and RPID headers will be created using the PPID header information.

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.

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, page 490
- Prerequisites for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element, page 493
- Restrictions for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element, page 493
- Configuring P-Header and Random-Contact Support on the Cisco Unified Border Element, page 493

Feature Information for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element

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

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

Table 55: Feature Information for PAID and PPID Headers on Cisco Unified Border Element (CUBE)

Feature Name	Releases	Feature Information
PAID and PPID Headers in mid-call re-INVITE and UPDATE request and responses on Cisco Unified Border Element	Cisco IOS 15.5(3)M Cisco IOS XE 3.16S	This feature enables CUBE platforms to support: • P-Preferred Identity (PPID) and P-Asserted Identity (PAID) in mid-call re-INVITE messages and responses from end-to-end. • P-Preferred Identity (PPID) and P-Asserted Identity (PAID) in mid-call UPDATE messages and responses from end-to-end. • Configuration and/or pass through of PAID and PPID header values.

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

Table 56: Feature Information for PAID, PPID, Privacy, PCPID, and PAURI Headers on CUBE

Feature Name	Releases	Feature Information
PAID, PPID, Privacy, PCPID, and PAURI Headers on the Cisco Unified Border Element	12.4(22)YB 15.0(1)M Cisco IOS XE Release 3.1S	This feature enables CUBE platforms to support: • P-Preferred Identity (PPID), P-Asserted Identity (PAID), Privacy, P-Called Party Identity (PCPID), in INVITE messages • Translation of PAID headers to PPID headers and vice versa • Translation of From: or RPID headers to PAID or PPID headers and vice versa • Configuration and/or pass through of privacy header values • PCPID header to route INVITE messages • Multiple PAURI headers in the response messages (200 OK) it receives to REGISTER messages • P-Preferred Identity and P-Asserted Identity Headers The following commands were introduced: **call-route p-called-party-id**, **privacy-policy**, **random-contact**, **random-request-uri validate**, **voice-class sip call-route p-called-party-id**, **voice-class sip privacy-policy**, **voice-class sip random-contact**, and **voice-class sip random-request-uri validate**.

Prerequisites for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element

Cisco Unified Border Element

- Cisco IOS Release 12.4(22)YB or a later release must be installed and running on your Cisco Unified Border Element.

Cisco Unified Border Element (Enterprise)

- Cisco IOS XE Release 3.1S or a later release must be installed and running on your Cisco ASR 1000 Series Router.

Restrictions for Support for PAID PPID Privacy PCPID and PAURI Headers on the Cisco Unified Border Element

- To enable random-contact support, you must configure the Cisco Unified Border Element to support SIP registration with random-contact information. In addition, you must configure random-contact support in VoIP voice-service configuration mode or on the dial peer.
- If random-contact support is configured for SIP registration only, the system generates the random-contact information, includes it in the SIP REGISTER message, but does not include it in the SIP INVITE message.
- If random-contact support is configured in VoIP voice-service configuration mode or on the dial peer only, no random contact is sent in either the SIP REGISTER or INVITE message.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Router(config)# voice service voip`	Enters VoIP voice-service configuration mode.
Step 4	**sip** **Example:** `Router(conf-voi-serv)# sip`	Enters voice service VoIP SIP configuration mode.
Step 5	**asserted-id** *header-type* **Example:** `Router(conf-serv-sip)# asserted-id ppi`	Specifies the type of privacy header in the outgoing SIP requests and response messages.
Step 6	**exit** **Example:** `Router(conf-serv-sip)# exit`	Exits the current mode.

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

	Command or Action	Purpose
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** *tag* **voip** **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	**voice-class sip asserted-id** *header-type* **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	**exit** **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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Router(config)# voice service voip`	Enters VoIP voice-service configuration mode.
Step 4	**sip** **Example:** `Router(conf-voi-serv)# sip`	Enters voice service VoIP SIP configuration mode.
Step 5	**call-route p-called-party-id** **Example:** `Router(conf-serv-sip)# call-route p-called-party-id`	Enables the routing of calls based on the PCPID header.

	Command or Action	Purpose
Step 6	**random-request-uri validate** **Example:** `Router(conf-serv-sip)# random-request-uri validate`	Enables the validation of the random string in the Request URI of the incoming INVITE message.
Step 7	**exit** **Example:** `Router(conf-serv-sip)# exit`	Exits the current mode.

Configuring P-Called-Party-Id Support on an Individual Dial Peer

To configure P-Called-Party-Id support on an individual dial peer, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *tag* **voip**
4. **voice-class sip call-route p-called-party-id**
5. **voice-class sip random-request-uri validate**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
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.

	Command or Action	Purpose
Step 3	**dial-peer voice** *tag* **voip** **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	**voice-class sip call-route p-called-party-id** **Example:** `Router(config-dial-peer)# voice-class sip call-route p-called-party-id`	Enables the routing of calls based on the PCPID header on this dial peer.
Step 5	**voice-class sip random-request-uri validate** **Example:** `Router(config-dial-peer)# 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.
Step 6	**exit** **Example:** `Router(config-dial-peer)# exit`	Exits the current mode.

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

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice service voip** **Example:** `Router(config)# voice service voip`	Enters VoIP voice-service configuration mode.
Step 4	**sip** **Example:** `Router(conf-voi-serv)# sip`	Enters voice service VoIP SIP configuration mode.
Step 5	**privacy** *privacy-option* **Example:** `Router(conf-serv-sip)# privacy id`	Enables the privacy settings for the header.
Step 6	**privacy-policy** *privacy-policy-option* **Example:** `Router(conf-serv-sip)# privacy-policy passthru`	Specifies the privacy policy to use when passing the privacy header from one SIP leg to the next.
Step 7	**exit** **Example:** `Router(conf-serv-sip)# exit`	Exits the current mode.

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

	Command or Action	Purpose
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** *tag* **voip** **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	**voice-class sip privacy** *privacy-option* **Example:** `Router(config-dial-peer)# voice-class sip privacy id`	Enables the privacy settings for the header on this dial peer.
Step 5	**voice-class sip privacy-policy** *privacy-policy-option* **Example:** `Router(config-dial-peer)# voice-class sip privacy-policy passthru`	Specifies the privacy policy to use when passing the privacy header from one SIP leg to the next, on this dial peer.
Step 6	**exit** **Example:** `Router(config-dial-peer)# exit`	Exits the current mode.

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*
5. **registrar ipv4:** *destination-address* **random-contact expires** *expiry*
6. **exit**
7. **voice service voip**
8. **sip**
9. **random-contact**
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
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.
Step 4	**credentials username** *username* **password** *password* **realm** *domain-name* **Example:** `Router(config-sip-ua)# credentials username 123456 password cisco realm cisco`	Sends a SIP registration message from the Cisco Unified Border Element.
Step 5	**registrar ipv4:** *destination-address* **random-contact expires** *expiry*	Enables the SIP gateways to register E.164 numbers on behalf of analog telephone voice ports (FXS), IP phone virtual voice

	Command or Action	Purpose
	Example: `Router(config-sip-ua)# registrar ipv4:10.1.2.2 random-contact expires 200`	ports (EFXS), and Skinny Client Control Protocol (SCCP) phones with an external SIP proxy or SIP registrar. • The **random-contact** keyword configures the Cisco Unified Border Element to send the random string from the REGISTER message to the registrar.
Step 6	**exit** **Example:** `Router(config-sip-ua)# exit`	Exits the current mode.
Step 7	**voice service voip** **Example:** `Router(config)# voice service voip`	Enters VoIP voice-service configuration mode.
Step 8	**sip** **Example:** `Router(conf-voi-serv)# sip`	Enters voice service VoIP SIP configuration mode.
Step 9	**random-contact** **Example:** `Router(conf-serv-sip)# random-contact`	Enables random-contact support on a Cisco Unified Border Element.
Step 10	**exit** **Example:** `Router(conf-serv-sip)# exit`	Exits the current mode.

Configuring Random-Contact Support for an Individual Dial Peer

To configure random-contact support for an individual dial peer, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sip-ua**
4. **credentials username** *username* **password** *password* **realm** *domain-name*
5. **registrar ipv4:** *destination-address* **random-contact expires** *expiry*
6. **exit**
7. **dial-peer voice** *tag* **voip**
8. **voice-class sip random-contact**
9. **exit**

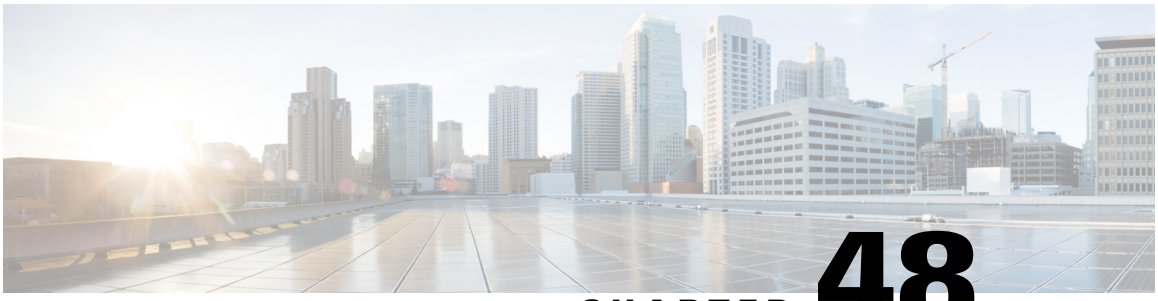
DETAILED STEPS

	Command or Action	Purpose
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.
Step 4	**credentials username** *username* **password** *password* **realm** *domain-name* **Example:** `Router(config-sip-ua)# credentials username 123456 password cisco realm cisco`	Sends a SIP registration message from the Cisco Unified Border Element.
Step 5	**registrar ipv4:** *destination-address* **random-contact expires** *expiry* **Example:** `Router(config-sip-ua)# registrar ipv4:10.1.2.2 random-contact expires 200`	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. • The **random-contact** keyword configures the Cisco Unified Border Element to send the random string from the REGISTER message to the registrar.

	Command or Action	Purpose
Step 6	**exit** **Example:** `Router(config-sip-ua)# exit`	Exits the current mode.
Step 7	**dial-peer voice** *tag* **voip** **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 8	**voice-class sip random-contact** **Example:** `Router(config-dial-peer)# voice-class sip random-contact`	Enables random-contact support on this dial peer.
Step 9	**exit** **Example:** `Router(config-dial-peer)# exit`	Exits the current mode.

PART XIII

SIP Supplementary Services



CHAPTER 48

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

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 57: Feature Information for Dynamic REFER Handling

Feature Name	Releases	Feature Information
REFER Consume (Enhancements)	IOS 15.5(1)T IOS XE 3.14.0 S	REFER Consume (Enhancements) provides additional configurations to conditionally forward the REFER message. The following commands were introduced: **refer consume**.
Dynamic REFER Handling	IOS 15.2(1)T IOS XE Release 3.7S	The Dynamic REFER handling feature provides configurations to pass across or consume the REFER message The following commands were introduced: **referto-passing**, **voice-class sip referto-passing**.

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.

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.

supplementary service refer	Results
yes	REFER is passed through from the in leg to the out leg
no	INVITE is sent towards refer-to dial-peer

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:
   - **referto-passing** in Global VoIP SIP configuration mode.
   - **voice-class sip referto-passing [system]** in dial-peer configuration mode.
5. **end**

DETAILED STEPS

	Command or Action	Purpose
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	Configure REFER passthrough:
	• **supplementary-service sip refer** in global VoIP configuration mode.
	• **supplementary-service sip refer** in dial-peer configuration mode.
	
	Example:
	In Global VoIP configuration mode:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# supplementary-service sip refer`
	
	Example:
		Configures REFER passthrough. A REFER is sent towards the inbound dial peer

	Command or Action	Purpose

	In dial-peer configuration mode:
	`Device(config)# dial-peer voice 22 voip`
	`Device(config-dial-peer)# supplementary-service sip refer`	

Step 4	Configure unmodified Refer-to:
	• **referto-passing** in Global VoIP SIP configuration mode.
	• **voice-class sip referto-passing [system]** in dial-peer configuration mode.
	Example:
	In Global VoIP configuration mode:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# sip`
	`Device(conf-serv-sip)# referto-passing`
	Example:
	In dial-peer configuration mode:
	`Device(config)# dial-peer voice 22 voip`
	`Device(config-dial-peer)# voice-class sip referto-passing`
		(Optional)
		Ensures that the refer-to header is unmodified and not changed to the IP address during passthrough

Step 5	**end**	Exits to privileged EXEC mode.

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 58: Configurations for REFER Consumption

supplementary service refer	refer consume	Results
yes	no	REFER is sent towards inbound dial-peer
yes	yes	INVITE is sent towards refer-to dial-peer
no	no	INVITE is sent towards refer-to dial-peer
no	yes	INVITE is sent towards refer-to dial-peer

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. **end**

DETAILED STEPS

	Command or Action	Purpose
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:
	• **no supplementary-service sip refer** in global VoIP configuration mode.
	• **no supplementary-service sip refer** in dial-peer configuration mode.
	
	Example:
	In global VoIP configuration mode:
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# no supplementary-service sip refer`
	
	Example:
	In dial-peer configuration mode:
	`Device(config)# dial-peer voice 22 voip`
	`Device(config-dial-peer)# no supplementary-service sip refer`
		Configures REFER consumption. An INVITE is sent towards the Refer-to dial peer.

	Command or Action	Purpose

Step 4	**refer consume** in global VoIP configuration mode.
	
	Example:
	In dial-peer configuration mode:
	
	`Device(config)# dial-peer voice 22 voip`
	`Device(config-dial-peer)# refer consume`
		Configures REFER consumption.

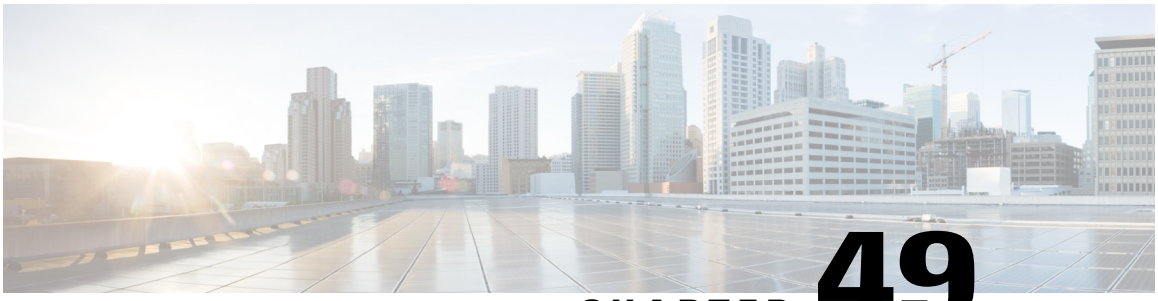
Step 5	**supplementary-service media-renegotiate** in global VoIP configuration mode.
	
	Example:
	In global VoIP configuration mode:
	
	`Device(config)# voice service voip`
	`Device(conf-voi-serv)# supplementary-service media-renegotiate`
		(Optional)
		Enables end-to-end media renegotiation during the call transfer in REFER consumption mode.

Step 6	**end**	Exits to privileged EXEC mode.

Troubleshooting Tips

Use any of the following debug commands:

- **debug ccsip all**
- **debug voip ccapi inout**
- **debug sccp messages**
- **debug voip application supplementary-service**
- **debug voip application state**
- **debug voip application media negotiation**



CHAPTER 49

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

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 59: Feature Information for Cause Code Mapping

Feature Name	Releases	Feature Information
Cause Code Mapping	Cisco IOS 15.5(1)T Cisco IOS XE 3.14S Cisco IOS 15.5(1)T3 Cisco IOS 15.5(1)S3 Cisco IOS 15.5(2)T1 Cisco IOS 15.5(2)S1 Cisco IOS 15.4(3)M4 Cisco IOS 15.4(3)S4	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: • **17—**486 Busy Here • **19—**503 Service Unavailable • **21—**403 Forbidden • **31—**480 Temporarily Unavailable • **102—**504 Server Time-out
Cause Code Mapping (Enhancement)	Cisco IOS 15.6(1)T	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:

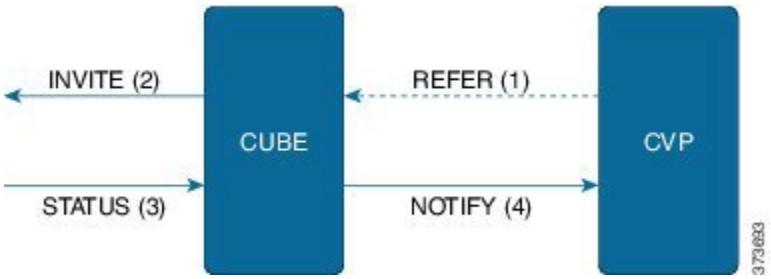
1 CUBE consumes the REFER that it receives from a Customer Voice Portal (CVP).

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 44: 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 60: Cause Code Mappings

Status Message received by CUBE (Step 3)	Cause Code	Notify message sent to CVP (Step 4)
486	17	486 Busy Here
480	31	480 Temporarily Unavailable
403	21	403 Forbidden
480	19	503 Service Unavailable
504	102	504 Server Time-out
404	1	404 Not Found
480	20	480 Temporarily Unavailable
484	28	484 Address Incomplete
502	27	502 Bad Gateway
503	38	503 Service Unavailable

Note Cause code mappings for cause code 19 and 21 require configurations mentioned in Configuring Cause Code Mapping, on page 516.

Note This mapping is only for the REFER consume scenario and not for REFER passthrough.

Configuring Cause Code Mapping

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sip-ua**
4. **reason-header override**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**sip-ua** **Example:** `Device(config)# sip-ua`	Enters the SIP user agent configuration mode.
Step 4	**reason-header override** **Example:** `Device(config-sip-ua)# reason-header override`	Configures the sending of a proper reason for failure of call transfer in the NOTIFY message so that the Customer Voice Portal (CVP) can take an appropriate action.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-sip-ua)# end`	Exits to privileged EXEC mode.

Verifying Cause Code Mapping

SUMMARY STEPS

1. Enter the following:
   - **debug ccsip function**
   - **debug ccsip message**
   - **debug voip application state**
   - **debug voip application core**
   - **debug voip ccapi inout**

DETAILED STEPS

Enter the following:

- **debug ccsip function**
- **debug ccsip message**
- **debug voip application state**
- **debug voip application core**
- **debug voip ccapi inout**

Example:
486 Received by CUBE:

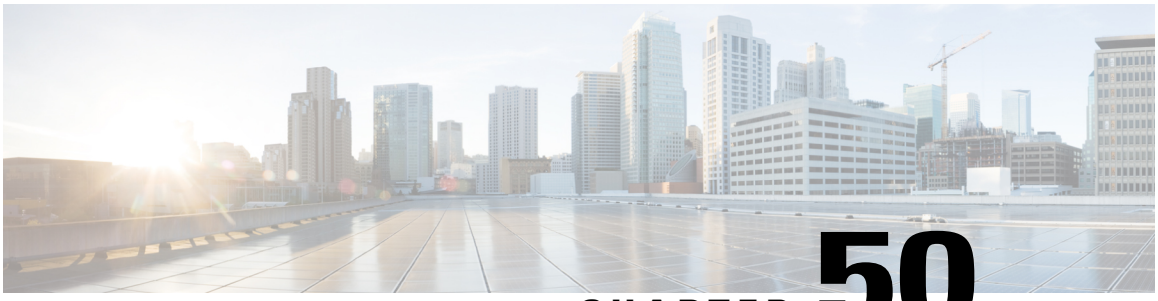
```
Received:
SIP/2.0 486 Busy Here
Via: SIP/2.0/UDP 9.40.3.231:5060;branch=z9hG4bK1C15625F7
From: <sip:2222@9.40.3.231>;tag=49B0964D-213C
To: <sip:3333@9.0.0.174>;tag=1
Call-ID: 7D7073E4-3F3B11E4-917BF9A9-A90B2232@9.40.3.231
CSeq: 101 INVITE
Allow-Events: telephone-event
Content-Length: 0
Reason: Q.850;cause=17
```

486 Busy here response sent in NOTIFY by CUBE

```
Sent:
NOTIFY sip:1111@9.0.0.174:9000 SIP/2.0
Via: SIP/2.0/UDP 9.40.3.231:5060;branch=z9hG4bK1C1571767
From: <sip:2222@9.40.3.231:5060>;tag=49B08E64-1374
To: <sip:1111@9.0.0.174>;tag=1
Call-ID: 1-25970@9.0.0.174
CSeq: 102 NOTIFY
Max-Forwards: 70
Date: Fri, 19 Sep 2014 13:55:46 GMT
User-Agent: Cisco-SIPGateway/IOS-15.5.20140712.124355.
Event: refer
Subscription-State: terminated;reason=noresource
Contact: <sip:2222@9.40.3.231:5060>
Content-Type: message/sipfrag
Content-Length: 25

SIP/2.0 486 Busy here
```

PART XIV

Call Progress Analysis



CHAPTER 50

Call Progress Analysis Over IP-to-IP Media Session

The Call Progress Analysis Over IP-IP Media Session feature enables the detection of automated answering systems and live human voices on outbound calls and communicates the detected information to the external application. Typically, call progress analysis (CPA) is extensively used in contact center deployments in conjunction with the outbound Session Initiation Protocol (SIP) dialer, where CPA is enabled on the Cisco Unified Border Element (Cisco UBE), and digital signal processors (DSP) perform the CPA functionality.

Feature Information for Call Progress Analysis Over IP-IP Media Session

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 Call Progress Analysis Over IP-IP Media Session

Feature Name	Releases	Feature Information
Call Progress Analysis Over IP-to-IP Media Session	15.3(2)T	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: **call-progress-analysis**.
Call Progress Analysis Over IP-to-IP Media Session	Cisco IOS XE Release 3.9S	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: **call-progress-analysis**.
Support for additional call flows	15.5(2)T Cisco IOS XE Release 3.15S	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

Restrictions for Call Progress Analysis Over IP-to-IP Media Session

- Only SIP-to-SIP Early Offer (EO-to-EO) call flows are supported.
- Session Description Protocol (SDP) passthrough and flow-around media calls are not supported.
- Only the G711 flavor of codec is supported.
- High Availability (HA) is not supported.
- Skinny Client Control Protocol (SCCP)-based digital signal processor (DSP) farm is not supported.
- CPA cannot not be detected if Dialer uses Inband as DTMF relay mechanism, that is, Inband to RTP-NTE DTMF inter-working is not supported with CPA.

- CPA call record is not supported for "180 without SDP" and "Direct Call Connect (without 18x)" call flows from Service Provider.

Information About Call Progress Analysis Over IP-IP Media Session

Call Progress Analysis

Call progress analysis (CPA) is a DSP algorithm that analyzes the Real-Time Transport Protocol (RTP) voice stream to look for special information tones (SIT), fax or modem tones, human speech, and answering machine tones. CPA also passes the voice information to Cisco IOS or Cisco Unified Border Element (Cisco UBE).

CPA is initiated on receiving a new SIP INVITE with x-cisco-cpa content. While a call is in progress, the DSP or the Xcoder analyzes the incoming voice or media stream. The DSP identifies the type of voice stream based on statistical voice patterns or specific tone frequencies and provides the information to the Cisco UBE. The Cisco UBE notifies the dialer with a SIP UPDATE with x-cisco-cpa content along with the detected event. Based on the report, the caller (dialer) can decide to either transfer the call or terminate the call.

To use the CPA functionality, you must enable CPA and configure CPA timing and threshold parameters.

Table 62: X-cisco-cpa content meaning

SIP Message	Direction of Message	Meaning
18x or 200	Cisco IOS to dialer	Cisco UBE informs the dialer if CPA is enabled for a call or not.
New INVITE	Dialer to Cisco IOS	Dialer requests Cisco IOS or the Cisco UBE to activate the CPA algorithm for this session.
UPDATE	Cisco IOS to dialer	Cisco IOS or the Cisco UBE notifies the dialer about the detected event.

CPA Events

Table 63: CPA Event Detection List

CPA Event	Definition
Asm	Answer machine
AsmT	Answer machine terminate tone
CpaS	Start of the Call Progress Analysis
FT	Fax/Modem tone

CPA Event	Definition
LS	Live human speech
LV	Low volume or dead air call
SitIC	Special information tone IC -- Intercept -- Vacant number or Automatic Identification System (AIS)
SitNC	SIT tone NC—No Circuit (NC), Emergency, or Trunk Blockage
SitVC	SIT tone VC—Vacant Code
SitRO	SIT tone RO—Reorder Announcement
SitMT	Miscellaneous SIT Tone

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

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**dspfarm profile** *profile-identifier* **transcode** **Example:** `Device(config)# dspfarm profile 15 transcode`	Enters DSP farm profile configuration mode, defines a profile for DSP farm services, and enables the profile for transcoding.
Step 4	**call-progress-analysis** **Example:** `Device(config-dspfarm-profile)# call-progress-analysis`	Enables call progress analysis (CPA) on Cisco UBE. • You must configure this command to activate the CPA feature and set CPA parameters.
Step 5	**exit** **Example:** `Device(config-dspfarm-profile)# exit`	Exits DSP farm profile configuration mode and enters global configuration mode.
Step 6	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters voice service configuration mode.
Step 7	**cpa timing live-person** *max-duration* **Example:** `Device(conf-voi-serv)# cpa timing live-person 2501`	(Optional) Sets the maximum waiting time (in milliseconds) that the CPA algorithm uses to determine if a call is answered by a live human.
Step 8	**cpa timing term-tone** *max-duration* **Example:** `Device(conf-voi-serv)# cpa timing term-tone 15500`	(Optional) Sets the maximum waiting time (in milliseconds) that the CPA algorithm uses to wait for the answering machine termination tone after the answering machine is detected.

	Command or Action	Purpose
Step 9	**cpa threshold active-signal** *signal-threshold* **Example:** `Device(conf-voi-serv)# cpa threshold active-signal 18db`	(Optional) Sets the threshold (in decibels) of an active signal that is related to the measured noise floor level. • 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.
Step 10	**end** **Example:** `Device(conf-voi-serv)# end`	Exits voice service configuration mode and returns to privileged EXEC mode.

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

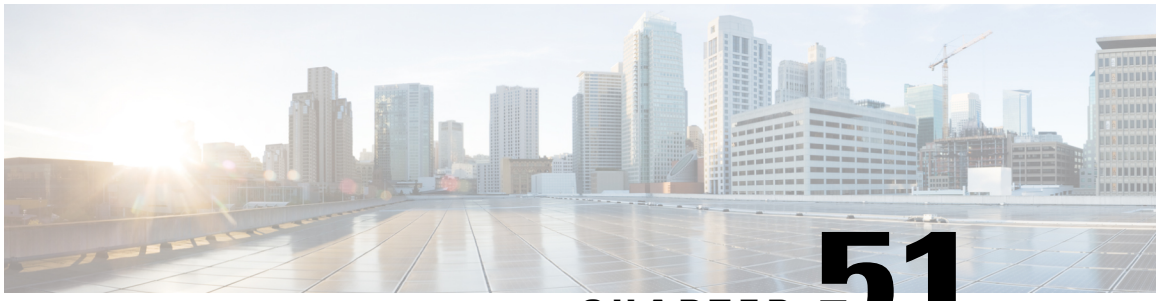
Example: Enabling CPA and Setting the CPA Parameters

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
```

PART XV

Cisco Unified Communications Manager Line-Side Support



CHAPTER 51

Cisco Unified Communications Manager Line-Side Support

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.

Finding Feature Information

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

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

Feature Information for 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 64: Feature Information for Cisco Unified Communications Manager Line-Side Support

Feature Name	Releases	Feature Information
Simplified Line-Side Support of CUCM on CUBE	15.4(2)T Cisco IOS XE Release 3.12S	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**.
Cisco Unified Communications Manager Line-Side Support	15.3(3)M Cisco IOS XE Release 3.10S	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**.

Restrictions for Cisco Unified Communications Manager Line-Side Support

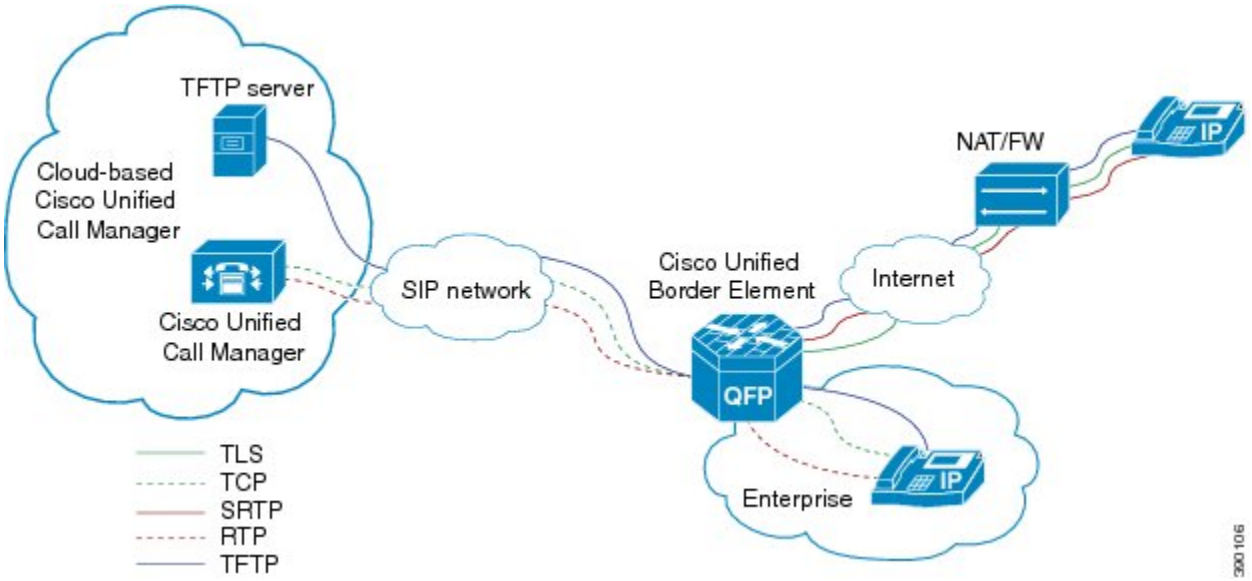
- In Cisco Unified Communications Manager Line-Side Support deployments, Cisco Unified Border Element does not support TFTP encrypted configuration files.

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 45: 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 65: Command for Line-Side Support for CUCM on CUBE

Dial-Peer Configuration Mode (config-dial-peer)	Global VoIP Configuration mode (config-voi-serv)
voice-class sip extension cucm	**sip** **extension cucm**

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 46: Predefined Supported NonMandatory Headers

```
!-- predefined hidden supported non-mandatory header pass-through list
!-- the list number 20001 is out of user configuration range

voice class sip-hdr-passthrulist 20001
 passthru-hdr Call-Info
 passthru-hdr Content-ID
 passthru-hdr Allow-Events
 passthru-hdr Supported
 passthru-hdr Remote-Party-ID
 passthru-hdr Require
 passthru-hdr Referred-By
```

371467

When Line Side Support for CUCM on CUBE is configured, predefined SIP profiles automatically remove the Cisco-Guide header from the outgoing INVITE.

Figure 47: Predefined SIP Profile

```
!-- predefined hidden sip profile
!-- the profile number 20001 is out of user configuration range

voice class sip-profiles 20001
 request INVITE sip-header Cisco-Guid remove
```

371468

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

	Command or Action	Purpose
Step 1	**crypto key generate rsa** [**label** *key-label*] [**modulus** *modulus-size*] **general-keys** **Example:** `Device(config)# crypto key generate rsa label pp_rsa modulus 1024 general-keys`	Generates a RSA key pair. **Note** A self-signed key can only support a *modulus-size* value of 1024 bits.
Step 2	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint callmg23`	Declares the trustpoint that the device should use and enters ca-trustpoint configuration mode.

	Command or Action	Purpose
Step 3	**enrollment selfsigned** **Example:** `Device(config-ca-trustpoint)# enrollment selfsigned`	Specifies self-signed enrollment for a trustpoint.
Step 4	**subject-name** [*x.500-name*] **Example:** `Device(config-ca-trustpoint)# subject-name CN=ASR1006-CCN-4`	Specifies the subject name in the certificate request.
Step 5	**subject-alt-name** *sip-security-profile-name* **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`	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.
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

	Command or Action	Purpose
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.
Step 3	**enrollment terminal** **Example:** `Device(config-ca-trustpoint)# enrollment terminal`	Specifies manual cut-and-paste certificate enrollment.
Step 4	**crypto pki authenticate** *name* **Example:** `Device(config-ca-trustpoint)# crypto pki authenticate cucm_trustpoint`	Authenticates the trustpoint. At the prompt to enter the certificate, copy the contents of the CallManager.pem file that you downloaded above and paste it at the prompt. At the prompt to accept the file, enter “yes”. **Note** When you copy the certificate, ensure that you also copy the BEGIN and END lines.

What to Do Next

Repeat the above steps for the CAPF key (the CAPF.pem file).

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

	Command or Action	Purpose
Step 1	**voice-ctl-file** *ctl-filename* **Example:** `Device(config)#voice-ctl-file ct1`	Creates a CTL file and enters CTL file configuration mode.
Step 2	**record-entry selfsigned trustpoint** *trustpoint-name* **Example:** `Device(config-ctl-file)#record-entry selfsigned trustpoint self-trustpoint6s`	Configures the trustpoints to be used for creating the CTL file.
Step 3	**record-entry capf trustpoint** *trustpoint-name* **Example:** `Device(config-ctl-file)#record-entry capf trustpoint capf-trustpoint6s`	Specifies that the trustpoint is created using the CAPF certificate imported from Cisco Unified Communications Manager to the device.
Step 4	**record-entry cucm-tftp trustpoint** *trustpoint-name* **Example:** `Device(config-ctl-file)#record-entry cucm-tftp trustpoint cucm-trustpoint`	Specifies that the trustpoint is created using the specified TFTP and Cisco Unified Communications Manager certificate imported to the device.
Step 5	**complete** **Example:** `Device(config-ctl-file)# complete`	Completes the CTL-file creation.

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

	Command or Action	Purpose
Step 1	**voice-phone-proxy** *phone-proxy-name* **Example:** `Device(config)# voice-phone-proxy pp`	Configures a phone proxy and enters phone-proxy configuration mode.
Step 2	**voice-phone-proxy file-buffer** *size* **Example:** `Device(config)# voice-phone-proxy file-buffer 30`	Configures the phone-proxy file buffering parameter, in MB.
Step 3	**tftp-server-address** [**ipv4** *server-ip-address* | *domain-name*] **Example:** `Device(config-phone-proxy)# tftp-server-address ipv4 172.110.36.2`	Configures the TFTP server address.
Step 4	**ctl-file** *ctl-filename* **Example:** `Device(config-phone-proxy)# ctl-file ct1`	Configures the CTL filename.
Step 5	**access-secure** **Example:** `Device(config-phone-proxy)# access-secure`	Specifies that the secure (encrypted) mode is to be used for access.
Step 6	**complete** **Example:** `Device(config-phone-proxy)# complete`	Completes the phone-proxy configuration.

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

	Command or Action	Purpose
Step 1	**dial-peer voice** *tag* **voip** **Example:** `Device(config)# dial-peer voice 10 voip`	Defines a particular dial peer, specifies the method of voice encapsulation, and enters dial peer configuration mode.
Step 2	**phone-proxy** *phone-proxy-name* **signal-addr ipv4** *ipv4-address* **cucm ipv4** *ipv4-address* **Example:** `Device(config-dial-peer)# phone-proxy pp1 signal-addr ipv4 10.0.0.8 cucm ipv4 198.51.100.1`	Configures the phone proxy for the related dial peer.
Step 3	**session protocol sipv2** **Example:** `Device(config-dial-peer)# session protocol sipv2`	Specifies a session protocol (SIPv2) for calls between local and remote devices.

	Command or Action	Purpose
Step 4	**session target registrar** **Example:** `Device(config-dial-peer)# session target registrar`	Specifies that a call from a VoIP dial peer is routed to the registrar end point.
Step 5	**session transport** {**udp** | **tcp** [tls]} **Example:** `Device(config-dial-peer)# session transport tcp tls`	Configures the underlying transport layer protocol for SIP messages to transport layer security over TCP (TLS over TCP).
Step 6	**incoming uri** {**from** | **request** | **to** | **via**} *tag* **Example:** `Device(config-dial-peer)# incoming uri request 11`	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.
Step 7	**destination uri** *tag* **Example:** `Device(config-dial-peer)# destination uri 12`	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.
Step 8	**voice-class sip call-route url** **Example:** `Device(config-dial-peer)# voice-class sip call-route url`	Enables call routing based on the URL.
Step 9	**voice-class sip profiles** *number* **Example:** `Device(config-dial-peer)# voice-class sip profiles 10`	Configures a SIP profile for a voice class.
Step 10	**voice-class sip registration passthrough** [**registrar-index** *index*] **Example:** `Device(config-dial-peer)# voice-class sip registration passthrough registrar-index 1`	Configures the SIP registration pass-through options on the dial peer.
Step 11	**voice-class sip pass-thru headers** **Example:** `Device(config-dial-peer)# voice-class sip pass-thru headers 10`	Configures a list of headers for pass through by referring to a globally configured list.

	Command or Action	Purpose
Step 12	**voice-class sip copy-list** {*tag* | **system**} **Example:** `Device(config-dial-peer)# voice-class sip copy-list 10`	Configures the list of entities to be sent to the peer call leg.
Step 13	**codec transparent** **Example:** `Device(config-dial-peer)# codec transparent`	Enables codec capabilities to be passed transparently between endpoints in a Cisco Unified Border Element.

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':
                    Source                                  Destination
 ------------------------------ Sessions of Dial-peer 5              -------------------------------
|Access: 10.74.9.219         :45232          10.74.9.209         :6970
  |
|Core  :   20.21.29.209       :45300          20.21.27.146      :6970
 |
-------------------------------------------------------------------------------------------------------
```

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 55C2FCBFBAC552B7C6CED497D4AD33F8
[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)# 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)# srtp
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
Device(config-sip-ua)# crypto signaling default trustpoint selfsignx
```

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 55C2FCBFBAC552B7C6CED497D4AD33F8
[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

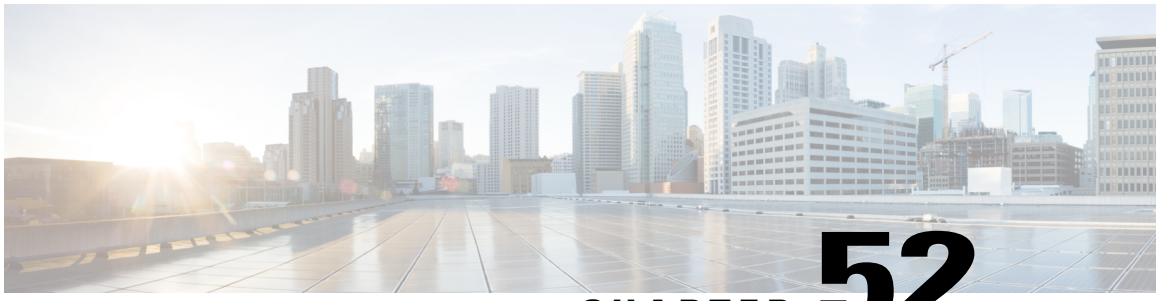
```
Device(config)# dial-peer voice 1 voip
Device(config-dial-peer)# phone-proxy phone_proxy signal-addr ipv4 172.18.110.120 cucm ipv4
 10.50.209.215
 *** Access Dialpeer Facing Outside ***
Device(config-dial-peer)# session protocol sipv2
Device(config-dial-peer)# session target registrar
Device(config-dial-peer)# 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
```

PART XVI

Licensing



CHAPTER 52

CUBE Licensing

Different Cisco IOS® Software images are available for the various platforms that support Cisco Unified Border Element (CUBE) and Cisco Gatekeeper functions. These Cisco IOS Software feature sets contain different levels of features. In addition to the software image, a feature license is required to turn on the desired function. Both the CUBE and Cisco Gatekeeper are licensed features.

Purchasing the appropriate software image is, in many cases, sufficient to deploy Cisco router features. In other cases, a feature license in addition to the software image is required. All features on platforms with a *universal* software image require a license.

Information About CUBE Licensing

The Cisco ® 800, 2900, 3900, and 3900E Series Integrated Services Router platforms have a *universal* Cisco IOS Software image. On these platforms all features are contained in the single software image, and the features that are available for use are determined only by the license(s) purchased. The following are the CUBE deployment use cases:

- Migration from TDM to SIP CUBE.
- CUBE in Contact Center
- CUBE is used as a protocol-interworking device with third party.

Unified Communications Bundles

Cisco® Integrated Services Router Unified Communications Bundles simplify the deployment of a unified communications (UC) solution by packaging a number of needed components in a single part number to make it easy to quote and order. Two types of UC bundles are offered for the Cisco 1861, 2800, and 3800 Series routers: Voice and Voice + Security.

Voice-only UC bundles include:

- The Integrated Services Router
- Cisco IOS® Software based Routing

- Cisco Unified Communications Manager Express for call processing
- Cisco Unity® Express hardware module and licenses for voicemail, integrated messaging, and interactive voice response.
- Digital signal processors (DSPs) for conferencing and connecting to a public switched telephone network (PSTN)
- Phone licenses to deploy Cisco IP phones
- Cisco Unified CallConnector Personal clients for Microsoft Windows desktop integration. This client is offered free of charge as an introductory offer only.

Voice + Security UC bundles also include security features like VPN, Cisco Intrusion Detection System, and Cisco IOS Firewall in addition to all the components in the voice-only UC bundles.

Note CUBE feature license(s) is included in all Cisco ISR UC bundles.

Cisco Unified Border Element License Types

Two types of Cisco Unified Border Element licenses are available:

- Pay-as-You-Grow (or session count) license—This type of license covers the right to use the feature as well as the maximum session count allowed. An example is the FL-CUBE-25 license, which allows up to 25 sessions. This license is designed to allow a specific number of sessions (or calls) on a platform. Purchase only as many sessions as are required in your deployment. You can add more licenses later as your needs expand, thus offering pay-as-you-grow benefits. Session licenses are stackable. These licenses are available on select platforms as shown in Table 1 in the section *CUBE Licenses*, and are available on all software images. Examples include the FL-CUBE-4 and FL-CUBEE-100 licenses.
- Platform (or flat) license—This type of license covers the right to use the feature up to the maximum session count supported on the chosen platform. An example is the FL-INTVVSRV-2811 license, which allows the maximum number of sessions the Cisco 2811 platform supports. These licenses are available on select platforms as given in Table 1 in the section *CUBE Licenses*. Examples include the FL-INTVVSRV-2801 and FL-GK-3945 licenses.

CUBE Licenses

The Cisco Unified Border Element licenses available are listed in the table below.

CUBE redundancy can be deployed in two ways:

- CUBE regular redundancy—Two CUBE platforms (ISR-G2 or ASR1K) are positioned in a **same local network** and are configured in an ACTIVE/STANDBY pairing. The CUBE-RED license allows transferring of the license from the Active to the Standby machine in the event of a failure.
- CUBE dual redundancy—CUBE in High Availability configuration in two independent Data Centers with a CUBE Active/Standby pair (ISR-G2 or ASR1K) in each. The CUBE-RED license allows transfer not only within one redundant pair from Active to Standby, but also from one pair to the other.

For Active/Standby redundant configurations, use the “Redundant-Platform” licenses.

Note Redundancy licenses cost about 30% more than Single-Use licenses.

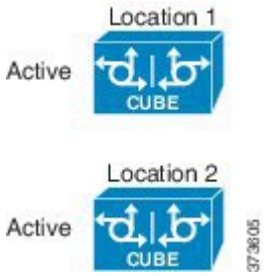
Table 66: CUBE License Choices per Platform

Platform	Single-Use Licenses	Redundancy Licenses (1 SKU for Active/Standby Pair)	License Type
Cisco 881, 886, 887, 888, 892F, SPIAD	FL-NANOCUBE	N/A	Platform
Cisco 2901, 2911, and 2921 Integrated Services Routers	FL-CUBEE-5 FL-CUBEE-25 FL-CUBEE-100	FL-CUBEE-5-RED FL-CUBEE-25-RED FL-CUBEE-100-RED	Pay-as-You-Grow Pay-as-You -Grow Pay-as-You -Grow
Cisco 2951 and 3925 Integrated Services Routers	FL-CUBEE-5 FL-CUBEE-25 FL-CUBEE-100 FL-CUBEE-500	FL-CUBEE-5-RED FL-CUBEE-25-RED FL-CUBEE-100-RED FL-CUBEE-500-RED	Pay-as-You-Grow Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow
Cisco 3945, 3925E, and 3945E Integrated Services Routers	FL-CUBEE-5 FL-CUBEE-25 FL-CUBEE-100 FL-CUBEE-500 FL-CUBEE-1000	FL-CUBEE-5-RED FL-CUBEE-25-RED FL-CUBEE-100-RED FL-CUBEE-500-RED FL-CUBEE-1000-RED	Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow
Cisco ASR 1000 Series Aggregation Services Routers	FLASR1-CUBEE-100P FLASR1-CUBEE-500P FLASR1-CUBEE-1KP FLASR1-CUBEE-4KP FLASR1-CUBEE-16KP	FLASR1-CUBEE-100R FLASR1-CUBEE-500R FLASR1-CUBEE-1KR FLASR1-CUBEE-4KR FLASR1-CUBEE-16KR	Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow
Cisco 2800 Series Integrated Services Routers	FL-CUBE-4 FL-CUBE-25 FL-CUBE-100 FL-INTVVSRV-2801 FL-INTVVSRV-2811 FL-INTVVSRV-2821 FL-INTVVSRV-2851	N/A	Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow Platform Platform Platform Platform

Platform	Single-Use Licenses	Redundancy Licenses (1 SKU for Active/Standby Pair)	License Type
Cisco 3800 Series Integrated Services Routers	FL-CUBE-4 FL-CUBE-25 FL-CUBE-100 FL-INTVVSRV-3825 FL-INTVVSRV-3845	N/A	Pay-as-You -Grow Pay-as-You -Grow Pay-as-You -Grow Platform Platform
Cisco AS5350XM and AS5400XM Universal Gateways	FL-INTVVSRV-5X	N/A	Platform
Cisco 880 Series Integrated Services Routers	FL-CUBE-4	N/A	Pay-as-You -Grow
Cisco IAD880 Integrated Access Devices	FL-CUBE-4		Pay-as-You -Grow
Cisco 1861 Integrated Services Router	FL-CUBE-4		Pay-as-You -Grow
Cisco IAD2430 Integrated Access Device	FL-CUBE-4 FL-CUBE-25		Pay-as-You -Grow Pay-as-You -Grow

Licensing Requirements for Customer Deployment Scenarios

Two Active CUBEs and No Redundancy



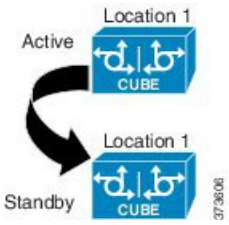
- Scenario—Two active CUBEs, no redundancy (that is no call preservation on failure of box), and no load balancing.
- Expectation—Expecting 500 sessions across each location.
- Licensing Requirement—**Two FL-CUBEE-500**.

Geographic Redundancy

- Scenario—Two active CUBEs, no redundancy, no call preservation on failure of box, but there is load balancing.
- Expectation—Expecting 500 sessions across each location, and in case one location fails, expecting the new 500 calls to failover to the other location.
- Licensing Requirement—**Two FL-CUBEE-500-RED**.
- If a box fails in this scenario, the calls on it are lost. The load balancing algorithm ensures that the next call is sent to the non-failed location.

Note In this scenario, you can also pool the CUBE licenses—one FL-CUBEE-1000-RED can be used.

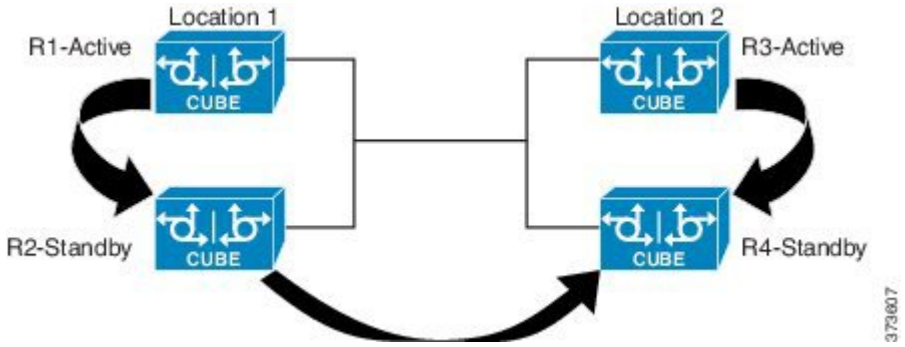
Layer 2 Box-to-Box Redundancy with Call Preservation



- Scenario—L2 box-to-box redundancy with call preservation.
- Expectation—Expecting 500 sessions across an active CUBE.
- Licensing Requirement—**One FL-CUBEE-500-RED**.

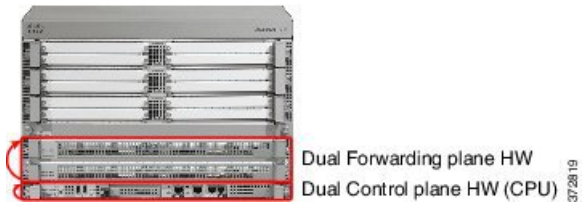
Note If the standby CUBE is at a different location, L2 box-to-box redundancy is not supported across geographical data centers. Typically, it is two boxes in the same rack.

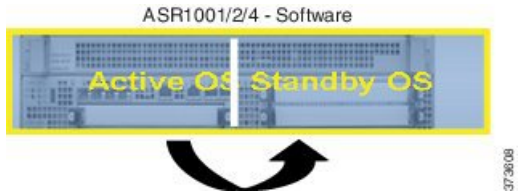
Box-to-Box Redundancy and Load Balancing Across Locations



- Scenarios Covered—Box-to-Box and Redundancy (call preservation on failure within location) and load balancing/redundancy across locations.
  - If R1 or R3 goes down, R2 or R4 respectively will take over.
  - If Location 1 (both R1 and R2) becomes unavailable, RED license allows newer calls to flow to Location 2. RED license allows transfer not only within one redundant pair from Active to Standby, allowing call preservation, but also from one pair to the other, that is from one Data Center to the other for new calls. In that case, Location 2 will handle 1000 sessions. This is called dual redundancy.
- Expectation—Expecting 500 sessions per location.
- Licensing Requirement—**Two FL-CUBEE-500-RED, one per Active/Standby pair. In total, you will have one 1000-RED**.

In-box Hardware and Software Redundancy





- Licensing Requirement—RED license is not required in this scenario; regular Single-Use CUBE license covers all in-box redundancies.

CUBE Licensing FAQs

Q. Is CUBE Licensing enforced?

A. No, CUBE is a paper-based honor license (no file to install) that allows you to run the CUBE RTU (Right-to-Use) feature set after you have the UCK9 license installed.

Q. Can CUBE licenses be transferred?

A.
- **No**, CUBE licensing is not transferable between chassis at this time.
- FL-CUBEE-XX licenses can be bought for any ISR-G2 platform, but cannot be transferred between platforms.
- FL-CUBEE-XX licenses are only for ISR-G2 (that is, you buy FL-CUBEE-5—it applies to a single ISR G2 that you buy it for, which could be a 2901, 2911, 3925, and so on, but only a single platform.)

Q. Are CUBE licenses incremental?

A. Yes, CUBE licenses can be added together to provide an aggregate session count. This way, customers can start with a smaller number of sessions and grow their system over time as call volume increases. For example, a customer may buy a FL-CUBEE-5 license to start with allowing a total of 5 sessions, and later add 2 more FL-CUBE-5 licenses for a total of 15 sessions.

Q. What constitutes a session?

A. A session is a single audio or a video call across the CUBE, regardless of call legs. Some vendors consider one call as two sessions.

Q. Does a call recording solution require additional licensing?

A. No, sessions created between CUBE and the Call Recording server such as MediaSense® do not require additional licenses and are not counted against the CUBE licensing limit. However, consider the platform capacity numbers.

Q. Can a customer migrate from a Single-Use to a RED license?

A. No, currently there are no migration SKUs, that is, if the customer previously purchased a Single-Use license, it cannot be converted into a RED license in future.

Q. Does CUBE Phone Proxy require licensing?

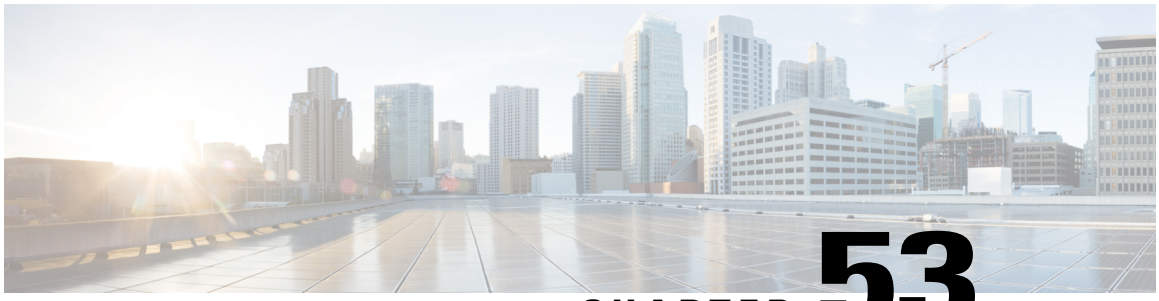
A. Yes, licenses will be required for the CUBE Phone Proxy feature just like it is required for trunk-side connections. Licenses will be required for the number of simultaneous off-hook phones, that is the number of phones trying to make calls concurrently.

Q. For what platforms is NanoCUBE licensing supported?

A. NanoCUBE licensing (SKU : FL-NANOCUBE) is a single license that allows you to go up to the maximum capacity of the platform. NanoCUBE is available on the platforms: 86X, 88X, and 89X; 2901/2911 is only for service providers.

PART XVII

Security



CHAPTER 53

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

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 Name	Releases	Feature Information
SIP TLS Version 1.0 Support on CUBE	Cisco IOS 12.4(6)T	Support is provided for SIP-to-SIP calls with Transport Layer Security (TLS) version 1.0. The following cipher suites are introduced for release Cisco IOS 12.4(6)T : • SSL_RSA_WITH_RC4_128_MD5 • TLS_RSA_WITH_AES_128_CBC_SHA The following commands were introduced or modified: **transport tcp tls** and **crypto signaling default trustpoint cube**.

Feature Name	Releases	Feature Information
SIP TLS Version 1.2 Support on CUBE	Cisco IOS 15.6(1)T Cisco IOS XE 3.17S	Support is provided for SIP-to-SIP calls with Transport Layer Security (TLS) version 1.2. The following cipher suites are introduced for release Cisco IOS 15.6(1)T: • TLS_DHE_RSA_WITH_AES_128_CBC_SHA1 • TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 The following commands were introduced or modified: **transport tcp tls**, **crypto signaling default trustpoint cube**, and **srtp (voice)**.

Restrictions

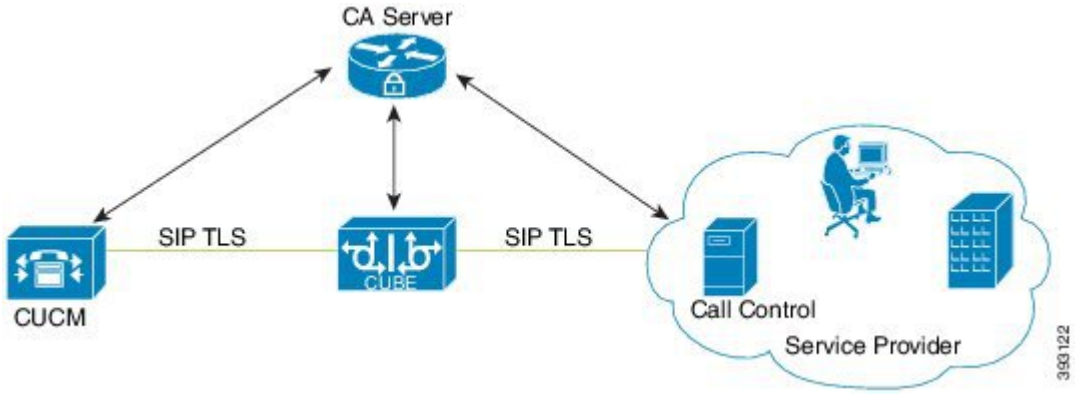
- EDCS ciphers are not supported on TLS version 1.0.

Information About SIP TLS Support on CUBE

Deployment

The following figure illustrates an example of CUBE with SIP TLS connections.

Figure 48: CUBE with SIP TLS connections



In a typical deployment, CUBE is placed between CUCM and the service provider. These devices are authenticated and enrolled with a Certificate Authority (CA) server that issues certificates. It can be Cisco or

a third party entity. When a call is made, a TLS handshake is initiated between CUCM and CUBE, and the IOS PKI infrastructure is used to exchange certificates signed by a common trusted CA during the handshake. During the TLS handshake, a dynamically generated symmetric key and cipher algorithms are negotiated between the devices. After the successful TLS handshake, the devices establish a SIP session between the service provider and CUBE. Keys exchanged during the TLS handshake process are used to encrypt or decrypt all SIP signaling messages.

Note The use of PKI on the Cisco IOS software requires that the clock on the devices be synchronized with the network time to ensure proper validation of certificates.

TLS Cipher Suite Category

Prior to release Cisco IOS15.6(1)T, CUBE supported TLS v1.0 with the following cipher suites:

- SSL_RSA_WITH_RC4_128_MD5
- TLS_RSA_WITH_AES_128_CBC_SHA

CUBE supports only the mandatory cipher suites for TLS implementation. From Cisco IOS15.6(1)T release onwards, CUBE supports TLS v1.2 which is backward compatible. Following are the cipher suites added:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA1
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

Use the **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 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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generatersa{general-keys | usage-keys}label** *key-label*[**exportable** |][**modulus** *modulus-size*][**storage** *device:*]
4. **crypto key generateeckey-size{256 | 384}**[**label***label*][*ec key-label*]
5. **crypto pki trustpoint** *name*
6. **rsakeypair** *key-label* [*key-size [encryption-key-size]*]
7. **eckeypair** *keyname*] *! Applicable only for TLS version 1.2.*
8. **serial-number** [**none**]
9. **ip-address** {*ip-address*|**interface**|**none**]
10. **subject-name** [*x.500-name*]
11. **enrollment** [**mode**][**retry period** *minutes*][**retry count** *number*]**url** *url*[**pem**]
12. **crl optional** or **revocation-check** *method1*[*method2*[*method3*]]
13. **password** *string*
14. **exit**
15. **crypto ca authenticate** *name* or **crypto pki authenticate** *name*
16. **crypto ca enroll** *name* or **crypto pki enroll** *name*
17. **sip-ua**
18. **crypto signaling**[(**remote-addr** {*ip address* |*subnet mask*})**default**]**trustpoint** *trustpoint-name*[**ecdsa-cipher**][**strict-cipher**] *! ECDSA ciphers are not supported on TLS version 1.0.*
19. **voice service** {**pots**| **voatm** |**vofr**|**voip**}
20. **transport tcp tls**
21. **url** {**sip**| **sips** |**tel**}
22. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generatersa{general-keys | usage-keys}label** *key-label*[**exportable** |][**modulus** *modulus-size*][**storage** *device:*] **Example:** `Router(config)# crypto key generate rsa general-keys label kp1 exportable`	Generates RSA key pairs. Arguments and keywords are as follows: • **general-keys—**Specifies that the general-purpose key pair should be generated. • **usage-keys—**Specifies that two RSA special-usage key pairs should be generated (that is, one encryption pair and one signature pair) instead of one general-purpose key pair. • **label key-label—**(Optional) Name that is used for an RSA key pair when they are being exported. If a key label is not specified, the fully qualified domain name (FQDN) of the router is used. • **exportable—**(Optional) Specifies that the RSA key pair can be exported to another Cisco device, such as a router. • **modulus modulus-size—**(Optional) IP size of the key modulus in a range from 350 to 2048. If you do not enter the modulus keyword and specify a size, you will be prompted. • **storage device:—**(Optional) Specifies the key storage location. The name of the storage device is followed by a colon (:). • **kp1—** kp1 is a label name that you select.
Step 4	**crypto key generateeckey-size{256 | 384}**[**label***label*][*ec key-label*] **Example:** `Router(config)# crypto key generate rsa general-keys label kp1 exportable`	Generates EC key pairs.
Step 5	**crypto pki trustpoint** *name* **Example:** `Router(config)# crypto pki trustpoint cube1`	Declares the trustpoint that your router should use. Argument is as follows: • *name***—**Creates a name for the trustpoint that you created. • cube1**—**Represents the trustpoint name that the user specifies.
Step 6	**rsakeypair** *key-label* [*key-size [encryption-key-size]*] **Example:** `Router(config)# rsakeypair kp1`	Specifies which key pair to associate with the certificate. Arguments are as follows: • *key-label***—**Name of the key pair, which is generated during enrollment if it does not already exists or if the **auto-enroll regenerate** command is configured. • *key-size***—**(Optional) Size of the desired RSA key. If not specified, the existing key size is used.

	Command or Action	Purpose
		• *encryption-key-size*—(Optional) Size of the second key, which is used to request separate encryption, signature keys, and certificates.
Step 7	**eckeypair** *keyname*] *! Applicable only for TLS version 1.2.* **Example:** `Router(config)# eckeypair mykey`	Generates EC keys for ECDSA cipher suites.
Step 8	**serial-number** [**none**] **Example:** `Router(ca-trustpoint)# serial-number`	Specifies whether the router serial number should be included in the certificate request. Keyword is as follows: • **none**—(Optional) Specifies that a serial number will not be included in the certificate request.
Step 9	**ip-address** {*ip-address*|**interface**|**none**] **Example:** `Router(ca-trustpoint)# ip-address 172.18.197.154`	Specifies a dotted IP address or an interface that will be included as "unstructuredAddress" in the certificate request. Arguments and keyword are as follows: • ip-address—Specifies a dotted IP address that will be included as "unstructuredAddress" in the certificate request. • interface—Specifies an interface, from which the router can get an IP address, that will be included as "unstructureAddress" in the certificate request. • **none**—Specifies that an IP address is not to be included in the certificate request.
Step 10	**subject-name** [*x.500-name*] **Example:** `Router(ca-trustpoint)# subject-name CN=172.18.197.154`	Specifies the subject name in the certificate request. Argument is as follows: • **x.500-name**—(Optional) Specifies the subject name used in the certificate request.
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 will wait before sending the CA another certificate request. The default is 1 minute between retries. (Specify from 1 through 60 minutes.) • **retry count number**—(Optional) Specifies the number of times a router will resend a certificate request when it does not receive a response from the previous request. The default is 10 retries. (Specify from 1 through 100 retries.)

	Command or Action	Purpose
		• **url url—**URL of the file system where your router should send certificate requests. For enrollment method options, see the enrollment url command. • **pem—**(Optional) Adds privacy-enhanced mail (PEM) boundaries to the certificate request.

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.
		• **ocsp—**Certificate checking is performed by an online certificate status protocol (OCSP).
		Note If the second and the third methods are specified, each method will be used only if the previous method returns an error, such as the server being down.

Step 13	**password** *string* **Example:** `Router(ca-trustpoint)# password password`	Specifies the revocation password for the certificate. Argument is as follows: • *string***—**Name of the password
Step 14	**exit** **Example:** `Router# exit`	Exists the current mode.

Step 15	**crypto ca authenticate** *name* or **crypto pki authenticate** *name*
	Example:
	`Router(config)# crypto ca authenticate cube1`
	`or`
	`Router(config)# crypto pki authenticate cube1`
		Authenticates the CA (by getting the certificate of the CA). Argument is as follows:
		• *name***—**Specifies the name of the CA. This is the same name used when the CA was declared with the crypto ca identity command.
		Note This is where you paste the remote root CA certificate (PEM file format).

	Command or Action	Purpose

Step 16	**crypto ca enroll** *name* or **crypto pki enroll** *name*
	
	Example:
	`Router(config)# crypto ca name cube1`
	`or`
	`Router(config)# crypto pki name cube1`
		Obtains the certificates of your router from the certificate authority. The CA server issues two certificates to the trustpoint (CUBE): one to certify the CA server and the other to certify the trustpoint (CUBE). Argument is as follows:
		• *name*—Specifies the name of the CA. Use the same name when you declared the CA using the **crypto pki trustpoint** command.

Step 17	**sip-ua** **Example:** `Router(config)# sip-ua`	Enters SIP user-agent configuration mode.
Step 18	**crypto signaling**[(**remote-addr** {*ip address* |*subnet mask*})**default**]**trustpoint** *trustpoint-name*[**ecdsa-cipher**][**strict-cipher**] *! ECDSA ciphers are not supported on TLS version 1.0.* **Example:** `Router(config-sip-ua)# crypto signaling default trustpoint cube1`	Configures 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 proces. **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. Keywords and arguments are as follows: • **remote-addr** *address*—Associates an IP address to a trustpoint. • **remote-addr** *subnet mask*—Associates a subnet mask to a trustpoint. • **default**—Configures a default trustpoint. • **trustpoint** *string*—Refers to the SIP gateways certificate generated as part of the enrollment process using Cisco IOS PKI commands • **ecdsa-cipher**—Examples are the following: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 and TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384. **Note** **ecdsa-cipher** is applicable only for the **TLS version 1.2** • **strict-cipher**—Examples are the following: TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA1, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, and TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.

	Command or Action	Purpose
Step 19	**voice service** {**pots**| **voatm** |**vofr**|**voip**} **Example:** `Router(config)# voice service voip`	Specifies a voice encapsulation type and enters voice service VoIP configuration mode.
Step 20	**transport tcp tls** **Example:** `Router(config-voi-sip)# transport tcp tls`	Enters this command in SIP configuration mode to enable the TLS port on TCP 5061 to listen.
Step 21	**url** {**sip**| **sips** |**tel**} **Example:** `Router(config-serv-sip)# url sips`	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 22	**end** **Example:** `Router(conf-serv-sip)# end`	Ends the current mode.

Verifying SIP TLS Support on CUBE

After a call is made, the **show sip-ua connections tcp tls** command is used to verify whether the transport used for the call is TLS.

Sample output for this command when TLS version is 1.0:

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      : 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 TLS-Version
  =========== ======= =========== =========== ============= ===========
         5061       1 Established           0   10.64.86.88     TLSv1.2
=================================================================================
```

Alternatively, the debug ccsip messages command can be used to verify the “Via:” header for TLS is included. This output is a sample INVITE request of a call that uses SIP TLS and the “sips:” URI scheme:

```
INVITE sips:777@172.18.203.181 SIP/2.0
Via: SIP/2.0/TLS 172.18.201.173:5060;branch=z9hG4bK2C419
From: <sips:333@172.18.201.173>;tag=581BB98-1663
To: <sips:5555555@172.18.197.154>
Date: Wed, 28 Dec 2005 18:31:38 GMT
Call-ID: EB5B1948-770611DA-804F9736-BFA4AC35@172.18.201.173
Remote-Party-ID: "Bob" <sips:+14085559999@1.2.3.4>
Contact: <sips:123@host>
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, COMET, 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
```

Configuration Examples for SIP TLS Support on CUBE

Example: SIP TLS Support on CUBE

```
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
!
no logging queue-limit
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
enrollment terminal
revocation-check none
!
!
crypto pki certificate chain ecdsacert1
certificate 07
  30820248 308201CD A0030201 02020107 300A0608 2A8648CE 3D040303 30593112
  30100603 5504030C 09706C75 746F3164 6F64310C 300A0603 55040B0C 03544143
  310E300C 06035504 0A0C0543 6973636F 310B3009 06035504 06130242 45311830
  16060355 04070C0F 74757269 6E672D65 7865632D 6C6E7830 1E170D31 35303831
  38313235 3431345A 170D3136 30383137 31323534 31345A30 36311330 11060355
  0403130A 706C7574 6F646F64 736E311F 301D0609 2A864886 F70D0109 02161063
  656E7472 616C6973 65645F72 74723230 76301006 072A8648 CE3D0201 06052B81
  04002203 62000446 4E28C72B 9A66C344 7D6EB2C7 51CE17F3 D125D12B 7043A98B
  F21825DF 0621A34D 3119E23F DB2A5ACE 1C744F17 789450F5 1071E504 DA7DC56C
  CDA24A8B 5318F11B EBA618A1 4BE2C66A 27857932 48485192 74923495 E762E3B4
  5BDCDBD0 BC6B1FA3 818B3081 88301D06 03551D0E 04160414 BE2A3FDE F3CDA85E
  A0EC7EA1 A47F3AEB 6B16D388 301F0603 551D2304 18301680 1460CAB8 AF1250CF
  BB00C516 ACEEAF72 FDB6198D 6C303606 082B0601 05050701 01042A30 28302606
  082B0601 05050730 01861A68 7474703A 2F2F2031 37332E33 392E3537 2E38333A
  38303830 2F300E06 03551D0F 0101FF04 04030207 80300A06 082A8648 CE3D0403
  03036900 30660231 00977017 6DCE34A4 3B0F78CF 2C69C7AD 2123B5F9 C10999E7
  A3316D34 43E9C928 8FBF42A4 84583017 856D513D C5B66547 1E023100 AEF7EFE8
  48AC2C81 884E8C8D 421A9B11 3177582D DBE9973F D67EA687 0CF08620 375628D0
  F5F7FDFA 53052711 E493A754
              quit
certificate ca 00
  3082023B 308201C1 A0030201 02020100 300A0608 2A8648CE 3D040303 30593112
  30100603 5504030C 09706C75 746F3164 6F64310C 300A0603 55040B0C 03544143
  310E300C 06035504 0A0C0543 6973636F 310B3009 06035504 06130242 45311830
  16060355 04070C0F 74757269 6E672D65 7865632D 6C6E7830 1E170D31 35303830
  36303934 3730345A 170D3136 30383035 30393437 30345A30 59311230 10060355
  04030C09 706C7574 6F31646F 64310C30 0A060355 040B0C03 54414331 0E300C06
  0355040A 0C054369 73636F31 0B300906 03550406 13024245 31183016 06035504
  070C0F74 7572696E 672D6578 65632D6C 6E783076 30100607 2A8648CE 3D020106
  052B8104 00220362 0004D2EE C8BE0015 AE8DF590 3F0A8955 C1B3D80F 99B3CE51
  241719ED 4D733BDC 061F92D0 36899A05 71E515B9 A86306B4 6DC49D66 87843054
  71E3151B 293971A2 94B14074 893BB537 09D4BC9C BF57E3DC AD5FA66B 590DA475
  B303068C 66899963 763CA35D 305B300C 0603551D 13040530 030101FF 300B0603
  551D0F04 04030201 06301D06 03551D0E 04160414 60CAB8AF 1250CFBB 00C516AC
  EEAF72FD B6198D6C 301F0603 551D2304 18301680 1460CAB8 AF1250CF BB00C516
  ACEEAF72 FDB6198D 6C300A06 082A8648 CE3D0403 03036800 30650230 390E60B9
  9AF19940 B0898320 AE96D8CB 52FB3B75 CE599444 EA3DBAC1 F4517F13 B96C26CB
  3B719834 A99AF174 6EF9E35D 0231008E 337B0A8F 864F32D4 C85CC7CC 585FCD8B
  6F5F4BCE 3B0313D1 E3B76598 0D9E43EB B11EFCF5 8C76318C 0F835560 0CD149
              quit
crypto pki certificate chain selfsign
certificate self-signed 01
  30820235 3082019E A0030201 02020101 300D0609 2A864886 F70D0101 05050030
  36311330 11060355 0403130A 706C7574 6F646F64 736E311F 301D0609 2A864886
  F70D0109 02161063 656E7472 616C6973 65645F72 74723230 1E170D31 35303831
  38313434 3234355A 170D3230 30313031 30303030 30305A30 36311330 11060355
  0403130A 706C7574 6F646F64 736E311F 301D0609 2A864886 F70D0109 02161063
  656E7472 616C6973 65645F72 74723230 819F300D 06092A86 4886F70D 01010105
  0003818D 00308189 02818100 A01400F8 9A599812 F5CC7347 1F9E223C E395073B
  9138C777 7EAEA997 5EA3B937 4B858866 2A022ADA 7D29C4C6 DC9B01EB 0E9E77DF
  782B099F 8F701221 A11C8B81 D82AB7F3 DBC1FFCB 809FC745 3FC6BD87 725F6B66
  EBEBBD78 6597DDFB 700D3DA6 73C52342 568670EA 1DEB6619 2ED5EC71 99B2612A
  BEC9B76E 38C442D9 DB9C2293 02030100 01A35330 51300F06 03551D13 0101FF04
  05300301 01FF301F 0603551D 23041830 1680141D 5971FE06 1D126AA3 6767DBA6
  C30E2EF0 2C044430 1D060355 1D0E0416 04141D59 71FE061D 126AA367 67DBA6C3
  0E2EF02C 0444300D 06092A86 4886F70D 01010505 00038181 0033BC90 8AF1DFBD
  B03AE032 ABBD80B7 7418402B 0BFB9E0B 341CB523 7077570C CD495BE3 47A1B35B
  C878C693 A491B433 37BA1170 45F1DF85 9BC22CA8 94E25907 F91C7B75 450B0DE1
  76AC2C6B 5517F42A 46260F76 4A1DF81F 733A14FE 918F43F4 9BABAA49 227B5014
  986044E7 8E98E373 7A361696 F0AD3ACC C9B101DF 2F80CCF7 E3
              quit
```

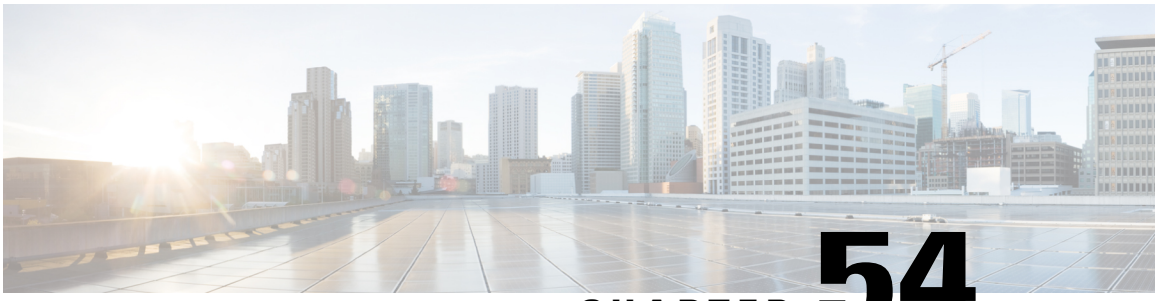
```
crypto pki certificate chain ccm155RSA
certificate ca 4E23E56C7339CC679FD444D77F7A463F
  308203AB 30820293 A0030201 0202104E 23E56C73 39CC679F D444D77F 7A463F30
  0D06092A 864886F7 0D01010B 0500306A 310B3009 06035504 06130249 4E310E30
  0C060355 040A0C05 63697363 6F310D30 0B060355 040B0C04 73727467 31143012
  06035504 030C0B50 4C55544F 2D435543 4D313112 30100603 5504080C 096B6172
  6E617461 6B613112 30100603 5504070C 0962616E 67616C6F 7265301E 170D3135
  30383034 31333431 35315A17 0D323030 38303231 33343135 305A306A 310B3009
  06035504 06130249 4E310E30 0C060355 040A0C05 63697363 6F310D30 0B060355
  040B0C04 73727467 31143012 06035504 030C0B50 4C55544F 2D435543 4D313112
  30100603 5504080C 096B6172 6E617461 6B613112 30100603 5504070C 0962616E
  67616C6F 72653082 0122300D 06092A86 4886F70D 01010105 00038201 0F003082
  010A0282 010100CC 39112782 D93A3501 8913EBEA 42522D27 E2C58D3F 4FC896D2
  8F38F4A5 7CCC2519 9683142A 6B203E9F C7C92673 85D5A940 99B20FBD CC8F97D1
  F42C1580 D34B8831 3BA74AE0 79AC0C74 E7BFAFCE 4D23F106 3D4EA333 16BA4768
  66C5561C 5CE19946 DA731D9E 6E743FA0 5F25E445 8E5B6789 64076291 7E5EB0DA
  C482074E 56DA6841 245EEB96 F44C900D 85C5EDEC 32E89675 BC934EC3 8C0FC7D8
  02BBCC06 93EE3698 A8B44527 93A73391 9C71869D BDEB96BF 06D68AC0 D47D810E
  FCAB3C8F 13BC3D62 02591976 CD49436E 3E2D5B20 079A031E 3FDDEC1C DFBF8261
  3CC5C6AF 7C6FC79C 0234D266 6C508DD7 CC72C8C6 239372F6 7D7CF5CD B56FFB26
  DB4122E2 01E15F02 03010001 A34D304B 300B0603 551D0F04 04030202 B4301D06
  03551D25 04163014 06082B06 01050507 03010608 2B060105 05070302 301D0603
  551D0E04 1604142D DF3CC8F3 57F44974 38D8E8E8 20B15658 9C17F430 0D06092A
  864886F7 0D01010B 05000382 01010038 060F1AC3 C3938667 8A3A0513 B5B2CE16
  0DC6BAE5 5B1D6DD7 CEB68832 F92A4270 5FC7EC97 7AAFB2AA 4FA288DD 66A94AB4
  A466CA7E F974B9B8 630FAC21 AB95C3BB ECB7A082 AB0343BE 2F89399D AD94D4A5
  6B477B44 88FB94BF FEE2E571 4917D0BB 2A5733B5 4F1F58BA CCCE710F 64365B39
  3F1F9E8F 81A1B71B 61BD51EB C45A2FAD CA743432 A61C19AB E6C4B5F1 6E673A38
  53421ECE 992505BD 5BAAF32A 954E37EA FE03B725 283A7F19 374A87E9 891E4E60
  B8399050 0902EA25 99FD2A26 2BD3A2E9 74F01C53 EFB3D4D6 654D064E 56878F6C
  21D8D184 88C24AD9 E655B78E 12EDB7EE 696B9B77 3E73A3F0 10DEBDF2 3CDF2BC9
  606700D1 2D42389C EEE43B56 22977A
              quit
voice-card 0
dspfarm
dsp services dspfarm
!
!
!
voice service voip
no ip address trusted authenticate
allow-connections sip to sip
fax protocol t38 version 0 ls-redundancy 0 hs-redundancy 0 fallback none
sip
  bind control source-interface GigabitEthernet0/1
  bind media source-interface GigabitEthernet0/1
  asymmetric payload full
  srtp negotiate cisco
!
!
!
!
voice iec syslog
!
!
!
!
mta send mail-from username $s$
license udi pid CISCO2921/K9 sn FGL1538116L
hw-module pvdm 0/0
!
!
!
no memory lite
!
redundancy
!
!
!
!
!
interface Embedded-Service-Engine0/0
no ip address
```

```
shutdown
!
interface GigabitEthernet0/0
ip address 9.45.38.192 255.255.0.0
shutdown
duplex auto
speed auto
!
interface GigabitEthernet0/1
ip address 10.64.86.177 255.255.255.0
ip traffic-export apply 1 size 5000000
duplex auto
speed auto
  no clns route-cache
!
interface GigabitEthernet0/2
no ip address
shutdown
duplex auto
speed auto
!
ip forward-protocol nd
!
ip http server
no ip http secure-server
!
ip rtcp report interval 9000
ip route 0.0.0.0 0.0.0.0 10.64.86.1
ip route 10.0.0.0 255.0.0.0 10.64.86.1
!
!
!
access-list 123 permit udp any any
access-list 123 permit tcp any any
!
control-plane
!
call treatment on
!
!
!
!
!
mgcp behavior rsip-range tgcp-only
mgcp behavior comedia-role none
mgcp behavior comedia-check-media-src disable
mgcp behavior comedia-sdp-force disable
!
mgcp profile default
!
sccp local GigabitEthernet0/1
sccp ccm 10.64.86.154 identifier 1 version 7.0
!
!
!
dial-peer voice 1 voip
destination-pattern 6003
session protocol sipv2
session target ipv4:10.64.86.206:5061
session transport tcp tls
incoming called-number 7003
codec g711ulaw
!
dial-peer voice 2 voip
destination-pattern 7003
session protocol sipv2
session target ipv4:10.64.86.206:5061
session transport tcp tls
incoming called-number 6003
codec g711ulaw
!
!
sip-ua
```

```
 transport tcp tls v1.2
connection-reuse
crypto signaling default trustpoint ecdsacert1 ecdsa-cipher
!
!
!
gatekeeper
shutdown
!
!
!
line con 0
exec-timeout 0 0
speed 115200
line aux 0
line 2
no activation-character
no exec
transport preferred none
transport output pad telnet rlogin lapb-ta mop udptn v120 ssh
stopbits 1
line vty 0 4
login
transport input none
!
!
end
```

PART XVIII

Voice Quality in CUBE



CHAPTER 54

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

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 67: Feature Information for Call Quality Statistics Enhancement

Feature Name	Releases	Feature Information
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.

For more information on how to calculate the voice quality metrics related to media(voice) quality, such as conversational mean opinion score (MOS), jitter, and so on, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube/configuration/cube-book/voi-cube-call-monitoring.html.

How to Configure Call Quality Parameters

Configuring Call Quality Criteria Parameters

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice service voip**
4. **call-quality**
5. **max-dropout** *number-of-packets*
6. **max-reorder** *number-of-packets*
7. **clock-rate** *payload-type-number frequency*
8. **clock-rate dynamic-default** *frequency*
9. **exit**
10. **rtcp all-pass-through**
11. **end**

DETAILED STEPS

	Command or Action	Purpose
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	Purpose
Step 3	**voice service voip** **Example:** `Device(config)# voice service voip`	Enters global VoIP configuration mode.
Step 4	**call-quality** **Example:** `Device(conf-voi-serv)# call-quality`	Enters call quality configuration mode; this is the global call quality of service setup.
Step 5	**max-dropout** *number-of-packets* **Example:** `Device(conf-serv-call-quality)# max-dropout 300`	Configures the acceptable out of sequence future packets to drop. The range is from 2 to 2000 packets. The default value is 100.
Step 6	**max-reorder** *number-of-packets* **Example:** `Device(conf-serv-call-quality)# max-reorder 500`	Configures the acceptable out of sequence late packets. The range is from 2 to 2000 packets. The default value is 100.
Step 7	**clock-rate** *payload-type-number frequency* **Example:** `Device(conf-serv-call-quality)# clock-rate 5 1500`	Sets the payload type number and frequency. Clock rate is the RTP timestamp field's sampling frequency.
Step 8	**clock-rate dynamic-default** *frequency* **Example:** `Device(conf-serv-call-quality)# clock-rate dynamic-default 10000`	(Optional) Changes the default clock rate for all the dynamic payload types. The frequency range (in Hz) is from 1000 to 192000. • You have several options to set the clock rate, such as for the different codecs.
Step 9	**exit** **Example:** `Device(conf-serv-call-quality)# exit`	Exits to global VoIP configuration mode.
Step 10	**rtcp all-pass-through** **Example:** `Device(conf-voi-serv)# rtcp all-pass-through`	(Optional) Passes through all RTCP in data path.
Step 11	**end** **Example:** `Device(conf-voi-serv)# end`	Returns to privileged EXEC mode.

Troubleshooting Call Quality Statistics

Use the following **debug** and **show** commands to enable the logs, which helps in debugging:

- **debug ccsip verbose**
- **debug voip fpi all**
- **debug platform hardware qfp active feature sbc dbe datapath all**
- **debug platform hard qfp act feature sbc dbe client all**
- **debug ccsip message**
- **debug ccsip info**
- **show call active voice**
- **show platform hardware qfp active feature sbc data path call** *call-id*

The following are some show command outputs that would be useful in troubleshooting:

- Device# show call active voice | include LostPackets

LostPackets=0

LostPackets=36 ---->//*Lost packets detail present in* ***show call active voice*** *output. View the complete command output based on the filters such as call-id to check the packet loss for a particular call leg.*//

- Device# show call active voice | include PlayDelayJitter

PlayDelayJitter=0

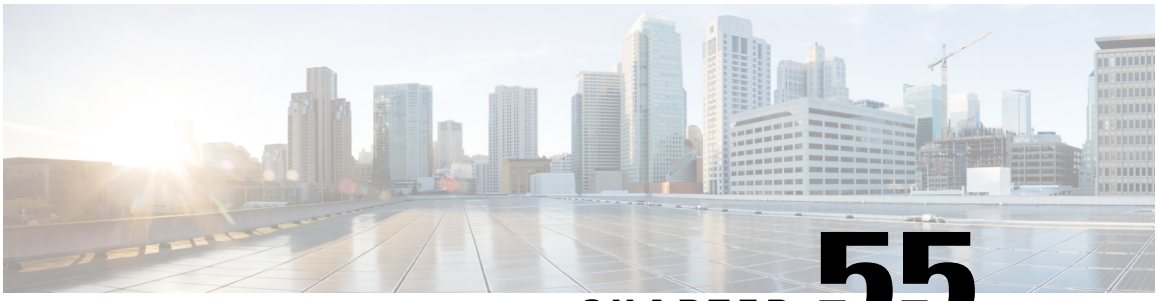
PlayDelayJitter=38 ----->//*Jitter detail present in* ***show call active voice*** *output. View the complete command output based on the filters such as call-id to check the Jitter for a particular call leg.*//

Configuration Example for Call Quality Statistics

```
voice service voip
 no ip address trusted authenticate
 callmonitor
 rtcp all-pass-through
 media statistics
 media bulk-stats
 allow-connections sip to sip
 call-quality
  max-dropout 2
  max-reorder 2
 sip
  g729 annexb-all
  no call service stop
```

PART XIX

Serviceability



CHAPTER 55

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

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 68: Feature Information for Session Identifier Support

Feature Name	Releases	Feature Information
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**

Restrictions

- Session Identifier is not supported for SIP-H.323, H.323-SIP, and H.323-H.323 calls.

Information About Session Identifier

CUBE supports “Session Identifier” that overcomes the limitations with the existing call-identifiers and allows end-to-end tracking of a SIP session. To support session identifier, “Session-ID” header is added in the SIP request and response messages.

Note "Session Identifier" refers to the value of the identifier, whereas "Session-ID" refers to the header field used to convey the identifier.

The Session-ID comprises of Universally Unique Identifier (UUID) for each user agent participating in a call. Each call consists of two UUID known as local UUID and remote UUID. Local UUID is the UUID generated from the originating user agent and remote UUID is generated from the terminating user agent. The UUID values are presented as strings of lower-case hexadecimal characters, with the most significant octet of the

UUID appearing first. Session Identifier comprises of 32 characters and remains same for the entire session. Refer to RFC 4122 for more information on UUID.

Example for Session ID header

```
Session-ID: ab30317f1a784dc48ff824d0d3715d86; remote=47755a9de7794ba387653f2099600ef2
```

In the above example:

Local UUID =

```
ab30317f1a784dc48ff824d0d3715d86
```

Remote UUID =

```
47755a9de7794ba387653f2099600ef2
```

Feature Behavior

- If all the user agents associated with CUBE support session-id, then CUBE allows pass-through of the Session ID header in all SIP request and response messages for the session.
- CUBE looks for the Session ID header present in the SIP messages and validates the SessionID header syntax as defined in draft-ietf-insipid-session-id-15. Session ID format earlier to draft-ietf-insipid-session-id-15 is considered as unsupported.
- If some of the user agents do not support session ID, CUBE generates local UUID on behalf of the user agent and sends the generated UUID in SIP request and response. CUBE generates UUID based on version 5 (SHA-1).
- If a Session ID is received in the format as defined in RFC 7329, CUBE considers it as unsupported. CUBE generates local UUID on behalf of the user agent and sends the generated UUID in SIP request and response.
- In a mid call scenario, where user a session is switched from supporting session identifier to non-supporting session identifier, CUBE saves the previous non-NULL session identifier and sends the saved non-NULL session identifier in re-invite messages as needed.
- For high availability, session ID is check pointed in active and re-created in standby.

Configuring Support for Session Identifier

Session Identifier support is enabled on CUBE by default. No additional configuration required.

Troubleshooting Tips

The following show commands helps you to troubleshoot any issues with session identifier.

- **show call active voice session-id** *WORD*
- **show call active voice brief session-id** *WORD*
- **show call active video session-id** *WORD*
- **show call active video brief session-id** *WORD*

WORD can be complete session identifier (local, remote, or both), or wildcard pattern of local or remote UUID. The valid wildcard patterns for search are *, 0-9, a-f, A-F.

The following session identifier is considered in the below examples:

```
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
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 db248b6cbdc547bbc6c6fdfb6916eeb;
remote=4fd24d9121935531a7f8d750ad16e19

Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
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 db248b6cbdc547bbc6c6fdfb6916eeb
Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 1
```

You can search for the session identifier using complete remote UUID as shown below:

```
Device# show call active voice session-id 4fd24d9121935531a7f8d750ad16e19
Telephony call-legs: 0
SIP call-legs: 1
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 1
```

You can search for session id using wildcard pattern match as shown below:

```
Device# Device# show call active voice session-id 4fd2*
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
```

```
.
.
.
SessionIDLocaluuid=4fd24d9121935531a7f8d750ad16e19
SessionIDRemoteuuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# show call active voice session-id *f*16e*
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=4fd24d9121935531a7f8d750ad16e19
SessionIDRemoteuuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# show call active voice brief session-id *
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=4fd24d9121935531a7f8d750ad16e19
SessionIDRemoteuuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# show call active voice session-id *; remote=*
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=4fd24d9121935531a7f8d750ad16e19
SessionIDRemoteuuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDLocaluuid=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

Device# show call active voice session-id 4fd24d9*;remote=*16eeb
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
.
```

```
.
.
SessionIDLocaluuid=4fd24d9121935531a7f8d750ad16e19
SessionIDRemoteuuid=db248b6cbdc547bbc6c6fdfb6916eeb
.
.
.
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2
```

Invalid Search Patterns

The following wild card search patterns are invalid:

```
Device# show call active voice session-id ;remote=
Invalid Pattern. Pattern can have a string with ^[0-9a-fA-F*]+$ only  OR a string with
^[0-9a-fA-F*];remote=[0-9a-fA-F*]+$.

Device# show call active voice session-id *;remote=
Invalid Pattern. Pattern can have a string with ^[0-9a-fA-F*]+$ only  OR a string with
^[0-9a-fA-F*];remote=[0-9a-fA-F*]+$.

Device# show call active video session-id ;remote=*
Incorrect format for Session-ID Wildcard Pattern regular expression must be of the form
^[0-9A-Fa-f*]+$
Invalid Pattern. Pattern can have a string with ^[0-9a-fA-F*]+$ only  OR a string with
^[0-9a-fA-F*];remote=[0-9a-fA-F*]+$.

Device# show call active voice session-id 4fd24d9*remote=*16eeb
Incorrect format for Session-ID Wildcard Pattern regular expression must be of the form
^[0-9A-Fa-f*]+$
Invalid Pattern. Pattern can have a string with ^[0-9a-fA-F*]+$ only  OR a string with
^[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=00000000000000000000000000000000
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19

Device# show call active voice session-id 0
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
.
.
.
SessionIDLocaluuid=00000000000000000000000000000000
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
.
.
.
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=db248b6cbdc547bbc6c6fdfb6916eeb
SessionIDRemoteuuid=4fd24d9121935531a7f8d750ad16e19
```

You can search for session identifier using the local UUID as shown below:

```
Device# show call active voice session-id d82c680a3eaecd5c29ac6ceeaa225061
Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
.
.
.
.
VOIP:
ConnectionId[0x8CDAC180 0x10000 0x1B7 0x5B56400A]
IncomingConnectionId[0x8CDAC180 0x10000 0x1B7 0x5B56400A]
CallID=1022
GlobalCallId=[0xC3DAB665 0x770C11E5 0x80318550 0x5A000ED7]
SessionIDLocaluuid=d82c680a3eaecd5c29ac6ceeaa225061
SessionIDRemoteuuid=6497636d0b747785241cfbf5aa225064
CallReferenceId=0
CallServiceType=Unknown
RTP Loopback Call=FALSE
RemoteIPAddress=10.64.86.91
RemoteUDPPort=16614
RemoteSignallingIPAddress=10.64.86.91
RemoteSignallingPort=5060
RemoteMediaIPAddress=10.127.17.142
RemoteMediaPort=16614
CoderTypeRate=g711ulaw
.
.
.
.
GlobalCallId=[0xC3DAB665 0x770C11E5 0x80318550 0x5A000ED7]
SessionIDLocaluuid=6497636d0b747785241cfbf5aa225064
SessionIDRemoteuuid=d82c680a3eaecd5c29ac6ceeaa225061
RemoteIPAddress=10.64.86.91
RemoteUDPPort=21978
RemoteSignallingIPAddress=10.64.86.91
RemoteSignallingPort=5060
RemoteMediaIPAddress=10.127.17.188
RemoteMediaPort=21978
```

From the above output, you get to know that 1022 (highlighted) is the call identifier associated with the local session identifier **d82c680a3eaecd5c29ac6ceeaa225061**. 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=6f0a93a3a79451aebeb6d83f79a3359f
SessionIDRemoteuuid=a55b0f45861551b88f57d1fb5bb23f89
```

Note All the search patterns listed above for voice calls are also valid for video calls.

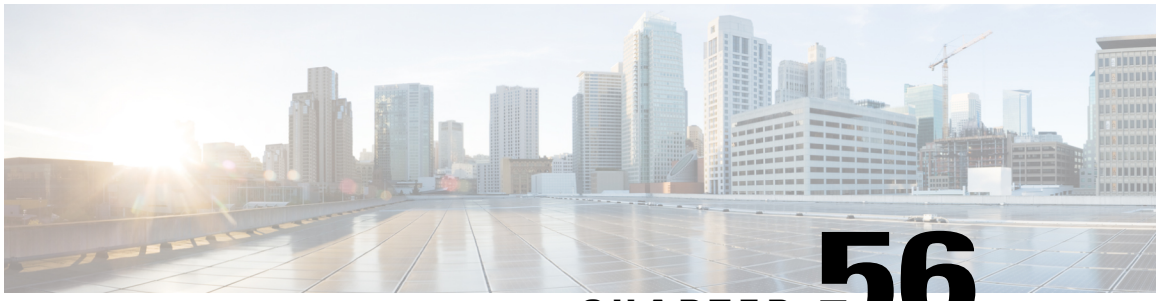
You can search for the session identifier using complete UUID (local, remote, or both) or use a wildcard pattern.

```
Device# show call active video session-id 6f*

Telephony call-legs: 0
SIP call-legs: 2
H323 call-legs: 0
Call agent controlled call-legs: 0
```

```
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 2

GENERIC:
SetupTime=56399650 ms (*16:58:12.964 IST Thu Aug 20 2015)
Index=1
PeerAddress=sipp
PeerSubAddress=
PeerId=1
PeerIfIndex=14
LogicalIfIndex=0
ConnectTime=56400660 ms (*16:58:13.974 IST Thu Aug 20 2015)
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=6f0a93a3a79451aebeb6d83f79a3359f
SessionIDRemoteuuid=a55b0f45861551b88f57d1fb5bb23f89
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 XX

Appendixes



CHAPTER 56

Additional References

The following sections provide references related to the CUBE Configuration Guide.

Related References

Related Topic	Document Title
Feature Navigator	For information about platforms supported, and Cisco IOS software image support., search by Feature Name listed in Feature Information Table in www.cisco.com/go/cfn
Bug Search Tool Kit	For information about latest caveats and feature information, see Bug Search Tool
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Cisco IOS Voice commands	*Cisco IOS Voice Command Reference*
Cisco IOS Voice Configuration Library	For more information about Cisco IOS voice features, including feature documents, and troubleshooting information--at http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/config_library/15-mt/cube-15-mt-library.html

Related Topic	Document Title
Related Application Guides	• *Cisco Unified Communications Manager and Cisco IOS Interoperability Guide* • *Cisco IOS SIP Configuration Guide* • Cisco Unified Communications Manager (CallManager) Programming Guides
Troubleshooting and Debugging guides	• Cisco IOS Debug Command Reference, Release 15.3. • *Troubleshooting and Debugging VoIP Call Basics* at http://www.cisco.com/en/US/tech/tk1077/technologies_tech_note09186a0080094045.shtml • *VoIP Debug Commands* at http://www.cisco.com/en/US/docs/routers/access/1700/1750/software/configuration/guide/debug.html

Standards

Standard	Title
ITU-T G.711	—

MIBs

MIB	MIBs Link
• CISCO-PROCESS MIB • CISCO-MEMORY-POOL-MIB • CISCO-SIP-UA-MIB • DIAL-CONTROL-MIB • CISCO-VOICE-DIAL-CONTROL-MIB • CISCO-DSP-MGMT-MIB • IF-MIB • IP-TAP-MIB • TAP2-MIB • USER-CONNECTION-TAP-MIB	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

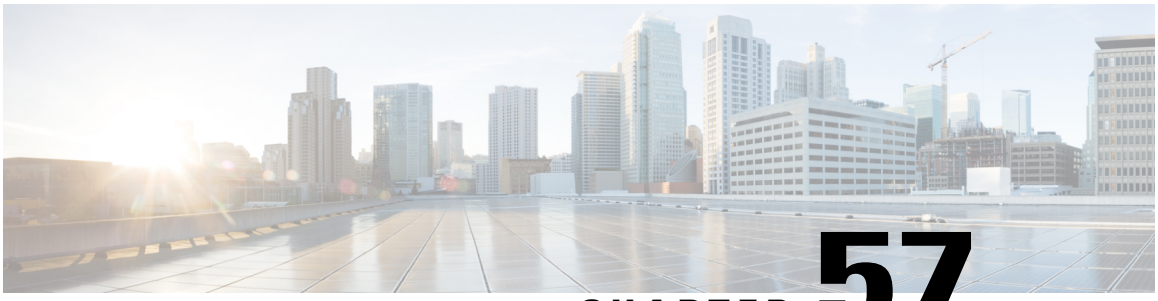
RFCs

RFC	Title
RFC 1889	*RTP: A Transport Protocol for Real-Time Applications*
RFC 2131	*Dynamic Host Configuration Protocol*
RFC 2132	*DHCP Options and BOOTP Vendor Extensions*
RFC 2198	*RTP Payload for Redundant Audio Data*
RFC 2327	*SDP: Session Description Protocol*
RFC 2543	*SIP: Session Initiation Protocol*
RFC 2543-bis-04	*SIP: Session Initiation Protocol, draft-ietf-sip-rfc2543bis-04.txt*
RFC 2782	*A DNS RR for Specifying the Location of Services (DNS SRV)*
RFC 2806	*URLs for Telephone Calls*

RFC	Title
RFC 2833	*RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals*
RFC 3203	*DHCP reconfigure extension*
RFC 3261	*SIP: Session Initiation Protocol*
RFC 3262	*Reliability of Provisional Responses in Session Initiation Protocol (SIP)*
RFC 3323	*A Privacy Mechanism for the Session Initiation Protocol (SIP)*
RFC 3325	*Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks*
RFC 3515	*The Session Initiation Protocol (SIP) Refer Method*
RFC 3361	*Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers*
RFC 3455	*Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)*
RFC 3608	*Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration*
RFC 3711	*The Secure Real-time Transport Protocol (SRTP)*
RFC 3925	Vendor-Identifying Vendor Options for Dynamic Host Configuration Protocol version 4 (DHCPv4)

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 57

Glossary

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 2*xx* INVITE response that it sends. Proxies and UACs determine this value from the Session-Expires header in a 2*xx* 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.