CUBE Media Proxy

CUBE Media Proxy is a solution that provides multiple forking function, and is built on CUBE architecture. Multiple forks are required for recorder redundancy and advanced media processing needs. The CUBE Media Proxy solution supports mandatory and optional recorders.

CUBE Media Proxy supports Unified CM Network-Based Recording (NBR) and SIP-Based Media Recording (SIPREC), to enable forking and recording of Real-Time Transport Protocol (RTP) streams.

Feature Information for CUBE Media Proxy

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

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. You do not require an account on Cisco.com.

Table 1: Feature Information for Recording Proxy

Feature Name	Releases	Feature Information
Secure forking of nonsecure calls	Cisco IOS XE Bengaluru 17.5.1a	CUBE Media Proxy supports both secure and nonsecure forking of nonsecure calls.
SIPREC-Based CUBE Media Proxy	Cisco IOS XE Amsterdam 17.3.1a	The SIPREC-based CUBE Media Proxy solution supports forking to multiple recorders.
CUBE Media Proxy	IOS XE Gibraltar Release 16.10.1a	The CUBE Media Proxy solution provides multiple forking functions for redundancy and advanced media processing.

Supported Platforms

CUBE Media Proxy is supported on the following Cisco router platforms running on Cisco IOS XE Software Releases:

- Cisco 4000 Series Integrated Services Routers (ISR4321, ISR4331, ISR4351, ISR4431, ISR4451, and ISR4461)
- Cisco Aggregated Services Routers (ASR - ASR1001-X, ASR1002-X, ASR1004 with RP2, ASR1006 with RP2, Cisco ASR1006-X Aggregated Services Routers with RP2 and ESP40, ASR 1006-X with RP3 and ESP40/ESP100)
- Cisco Cloud Services Routers (CSR1000V series)
- Cisco Catalyst 8000V Edge Software (Catalyst 8000V) series
- Cisco 8300 Catalyst Edge Series Platforms
- Cisco 8200 Catalyst Edge Series Platform (C8200-1N-4T)
- Cisco 8200L Catalyst Edge Series Platform (C8200L-1N-4T)

Note When upgrading to C8000V software from a CSR1000V release, an existing throughput configuration will be reset to a maximum of 250Mbps. Install an HSEC authorization code, which you can obtain from your Smart License account, before reconfiguring your required throughput level.

Restrictions for CUBE Media Proxy

CUBE Media Proxy using Unified CM NBR, and SIPREC-Based CUBE Media Proxy do not support the following:

- Forking of video sessions
- Recording of calls from endpoints that are registered with the Cloud. For example, Cisco Webex Calling.
- SRTP fallback

- Midcall block
- Concurrent use with CUBE B2BUA SBC features.
- Server Groups in outbound dial-peers toward recorders.
- Midcall updates from the recorders such as pause or resume recording, RE-INVITE with SDP changes, INVITE that replaces header that is sent by recorders when they switch from active to standby CUBE Media Proxy.

Note Midcall update "BYE" from the recorders is supported.

- Unified CM NBR and SIPREC for the same call flow.

The following restriction applies when using CUBE Media Proxy with Unified CM NBR:

- If the primary recorder sends a=inactive in the response SDP, the same is forwarded to Unified CM. Forking is not triggered to any of the recorders.

CUBE Media Proxy Using Unified CM Network-Based Recording

CUBE Media Proxy using Unified CM Network-Based Recording (NBR), is Unified CM dependent and requires you to configure inbound dial-peers from Unified CM. After receiving a media forking request from Unified CM, the CUBE Media Proxy establishes media forks to the configured targets.

SIPREC-Based CUBE Media Proxy

The SIPREC (SIP Media Recording) feature supports media recording for Real-Time Transport Protocol (RTP) streams in compliance with section 3.1.1. of RFC 7245, with CUBE Media Proxy acting as the Session Recording Client (SRC). SIP is used to establish a Recording Session between the CUBE Media Proxy and recorders (or any other media application).

For SIPREC solutions, CUBE Media Proxy accepts an inbound RTP fork from a CUBE SBC and replicates this RTP fork to multiple SIPREC targets based on its inbound configuration.

About Multiple Media Forking Using CUBE Media Proxy

Unified CM Network-Based CUBE Media Proxy and SIPREC-Based CUBE Media Proxy support the following functions:

- Media forking for up to five destinations per call
- Destination redundancy by hunting algorithm
- Media fork policy control
- Load balancing during initial call setup
- High Availability

- TLS, TCP, and UDP transport protocols
- Secure forking of nonsecure calls
- Secure forking of secure calls

Secure Forking of Secure and Nonsecure Calls

From Cisco IOS XE Bengaluru 17.5.1a onwards, you can configure a combination of secure and nonsecure forks for a nonsecure call.

CUBE Media Proxy Using Unified CM Network-Based Recording, on page 3 supports secure forking of secure and nonsecure calls.

Note You cannot use the **mandatory policy** command with secure forking configurations.

For SRTP pass through to work in secure media forking, the Command Line Interface **srtp pass-thru** should be configured at global or dial-peer level.

Deployment Scenarios for CUBE Media Proxy

Note From Cisco IOS XE Bengaluru 17.5.1a onwards, you can deploy a combination of secure and nonsecure destinations.

CUBE Media Proxy Using Unified CM Network-Based Recording

In Network Based Recording (NBR) deployments, Cisco Unified Communications Manager establishes an initial forked media leg with CUBE Media Proxy. This may either be from a phone using its built-in bridge (Deployment Scenario for CUBE Media Proxy Using Unified CM NBR for Internal Call), or from a CUBE SBC using the eXtended Media Forking (XMF) API (Deployment Scenario for CUBE Media Proxy Using Unified CM NBR for External Call).

Figure 1: Deployment Scenario for CUBE Media Proxy Using Unified CM NBR for Internal Call

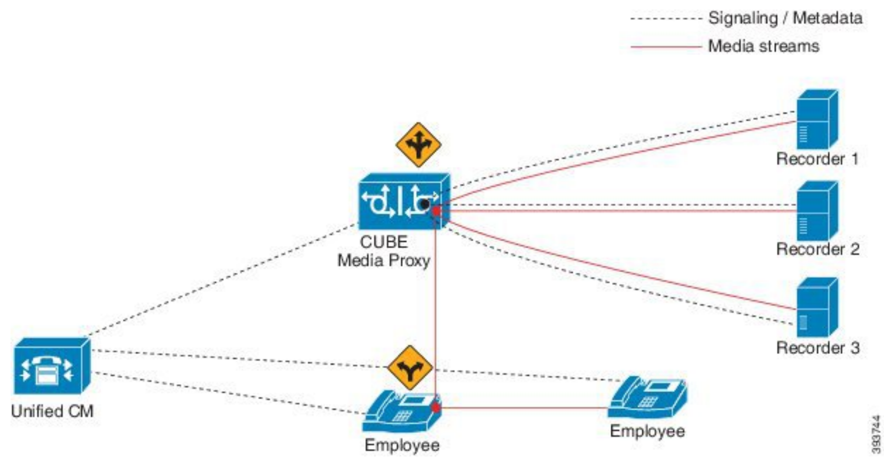
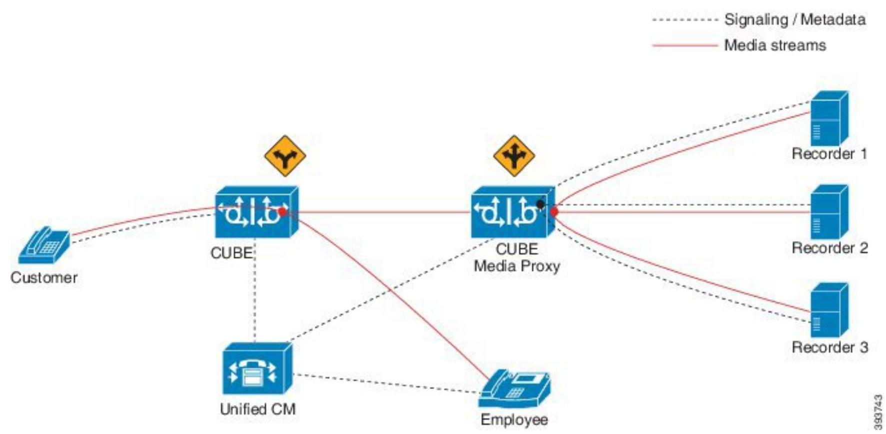


Figure 2: Deployment Scenario for CUBE Media Proxy Using Unified CM NBR for External Call



The information flow is as follows:

1. External or internal call is set up between the endpoints.
2. CUBE Media Proxy receives the media forking request from UCM.
3. CUBE Media Proxy sets up sessions with the recorders based on the proxy policy.
    - Mandatory recorder: Proxy policy is configured to set a recorder as mandatory. CUBE Media Proxy tries to establish connection with the mandatory recorder. Forking to the remaining recorders happen only if the connection with the mandatory recorder is successful.
    - Optional recorders: When the proxy policy is not configured, all the recorders are set as optional. CUBE Media Proxy tries to establish a connection with the remaining recorders even if any of the recorders fail.

Note

- If the CUBE Media Proxy receives a '486' response from the initial recorder, CUBE Media Proxy does not fork the INVITE to other recorders. To perform alternate routing, configure the **voice hunt user-busy** command in global configuration mode.

  Example: **Router(config)# voice hunt user-busy**

- Secure recorders: When secure recorders are configured, mandatory proxy policy configuration does not apply. CUBE Media Proxy tries to establish a connection with the first secure recorder from the list of configured dial-peers. Forking to the remaining recorders happens after establishing a connection with the first secure recorder.

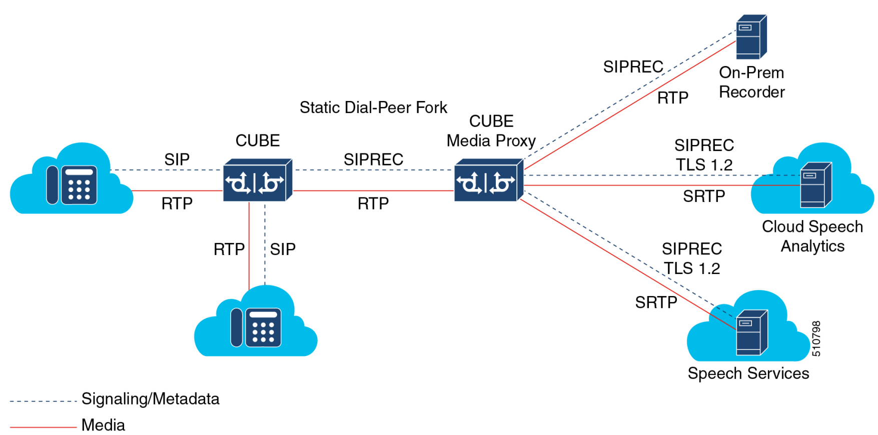
4. If required, Cisco Unified SIP Proxy may be used to route or load balance a media fork for a group of recorders.

Note The CUBE Media Proxy solution supports Unified CM Release 12.5.1 and Cisco Unified SIP Proxy Release 9.1.8.

SIPREC-Based CUBE Media Proxy

CUBE Media Proxy may be configured to fork media autonomously using SIPREC, as shown in the following scenario.

Figure 3: Deployment Scenario for SIPREC-Based CUBE Media Proxy



The information flow in this scenario is as follows:

1. CUBE SBC receives a call from a SIP trunk and routed to the intended destination.
2. CUBE SBC uses SIPREC to establish a media fork of the call with CUBE Media Proxy.
3. CUBE Media Proxy uses SIPREC to establish secure or nonsecure media forks with up to five destinations.

Note On receiving BYE from the primary secure recorder, Media Proxy disconnects all secure and nonsecure recording sessions. BYE received from any other recorder, secure or nonsecure, will not impact other active recording sessions.

Recording Metadata

Metadata is the information that a Recording Server (RS) receives from a Recording Client (RC) in a SIP session. Metadata has the following functions:

- Carries the communication session data that describes the call to the Recording Server.
- Identifies the participants list.
- Identifies the session and media association time.

Recording Metadata in CUBE Media Proxy Using Unified CM NBR

Unified CM passes information about the forked call to CUBE Media Proxy in up to 16 metadata parameters that are included in the **From** header of the SIP Invite. CUBE Media Proxy includes a copy of this metadata in the Invite it sends to the configured destinations. The following is an example of a **From** header with metadata.

Note The **From** header, including all metadata must not exceed 583 bytes.

Following is a sample SIP header of a recording request:

```
From: "abcd" <sip:198101@10.200.25.137;
    x-nearend;x-refci=27298698;x-nearendclusterid=NY-NJ-Labcluster;
    x-nearenddevice=SEP2834A28318CE;
    x-nearendaddr=198101;x-farendrefci=27298699;
    x-farendclusterid=NY-NJ-Labcluster;x-farenddevice=AFIFIM-VI1;x-farendaddr=172001;
    x-sessionid=696dd5d3f7755c6abdc438e93d01febf>;
    tag=14087~b35a5915-3167-4d6a-871d-c121221602bf-27298703
```

Recording Metadata in SIPREC-Based CUBE Media Proxy

The initial SIPREC Invite from CUBE to CUBE Media Proxy, and the SIPREC Invite from CUBE Media Proxy to the recorders, includes recording metadata in a SIPREC XML body.

Following is a sample SIPREC INVITE:

```
INVITE sip:9876@8.43.33.203:5060 SIP/2.0
Via: SIP/2.0/UDP 8.43.33.209:5060;branch=z9hG4bK20959B
From: <sip:8.43.33.209>;tag=678813-6AC
To: <sip:9876@8.43.33.203>
Date: Thu, 13 Feb 2020 03:35:19 GMT
Call-ID: B0FA2851-4D4811EA-82E5D263-E98F8024@8.43.33.209
```

```
Supported: 100rel,timer,resource-priority,replaces,sdp-anat
Require: siprec
Min-SE:  1800
Cisco-Guid: 2967454021-1296568810-2195116643-3918495780
User-Agent: Cisco-SIPGateway/IOS-17.3.20200207.160928
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE, NOTIFY, INFO,
REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Timestamp: 1581564919
Contact: <sip:8.43.33.209:5060>;+sip.src
Expires: 180
Allow-Events: telephone-event
Session-ID: 812eae44f57c50b38e897d75d8e12809;remote=00000000000000000000000000000000
Content-Type: multipart/mixed;boundary=uniqueBoundary
Mime-Version: 1.0
Content-Length: 2250

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

v=0
o=CiscoSystemsSIP-GW-UserAgent 5146 1045 IN IP4 8.43.33.209
s=SIP Call
c=IN IP4 8.43.33.209
t=0 0
m=audio 8278 RTP/AVP 0
c=IN IP4 8.43.33.209
a=rtpmap:0 PCMU/8000
a=ptime:20
a=sendonly
a=label:1
m=audio 8280 RTP/AVP 0
c=IN IP4 8.43.33.209
a=rtpmap:0 PCMU/8000
a=ptime:20
a=sendonly
a=label:2

--uniqueBoundary
Content-Type: application/rs-metadata+xml
Content-Disposition: recording-session

<?xml version="1.0" encoding="UTF-8"?>
<recording xmlns="urn:ietf:params:xml:ns:recording:1">
    <datamode>complete</datamode>
    <session session_id="sPVtz01IEeqC3dJj6Y+AJA==">

<sipSessionID>0e0960d88013509f86e7ad2d78da208a;remote=4d0de1325c205fa08f77d8d31c1b3a6f</sipSessionID>

        <start-time>2020-02-13T03:35:19.008Z</start-time>
    </session>
    <participant participant_id="sPVtz01IEeqC3tJj6Y+AJA==">
        <nameID aor="sip:3478@8.41.17.71">
        </nameID>
    </participant>
    <participantsessionassoc participant_id="sPVtz01IEeqC3tJj6Y+AJA=="
session_id="sPVtz01IEeqC3dJj6Y+AJA==">
        <associate-time>2020-02-13T03:35:19.008Z</associate-time>
    </participantsessionassoc>
    <stream stream_id="sPgFxk1IEeqC49Jj6Y+AJA==" session_id="sPVtz01IEeqC3dJj6Y+AJA==">
        <label>1</label>
    </stream>
```

```
    <participant participant_id="sPVtz01IEeqC39Jj6Y+AJA==">
        <nameID aor="sip:98765@8.41.17.71">
        </nameID>
    </participant>
    <participantsessionassoc participant_id="sPVtz01IEeqC39Jj6Y+AJA=="
session_id="sPVtz01IEeqC3dJj6Y+AJA==">
        <associate-time>2020-02-13T03:35:19.008Z</associate-time>
</participantsessionassoc>
    <stream stream_id="sPgFxk1IEeqC5NJj6Y+AJA==" session_id="sPVtz01IEeqC3dJj6Y+AJA==">
        <label>2</label>
    </stream>
    <participantstreamassoc participant_id="sPVtz01IEeqC3tJj6Y+AJA==">
        <send>sPgFxk1IEeqC49Jj6Y+AJA==</send>
        <recv>sPgFxk1IEeqC5NJj6Y+AJA==</recv>
    </participantstreamassoc>
    <participantstreamassoc participant_id="sPVtz01IEeqC39Jj6Y+AJA==">
        <send>sPgFxk1IEeqC5NJj6Y+AJA==</send>
        <recv>sPgFxk1IEeqC49Jj6Y+AJA==</recv>
    </participantstreamassoc>
</recording>

--uniqueBoundary--
```

For a SIPREC call, the Require header in the SIP Invite (from Cisco UBE to CUBE Media Proxy, and from CUBE Media Proxy to the recorders) must have a "siprec" extension. The Require header must also have metadata in the XML body, else the call is dropped. The Contact header in a SIP invite has a "+sip.src" extension.

Session Identifier

In both NBR and SIPREC modes, CUBE Media Proxy uses the Session-ID header in request and response messages to exchange session identifiers for tracking a recording session between peers.

The Session-ID comprises of the following two Universally Unique Identifiers (UUIDs) corresponding to the initiator and recipient of the recording request respectively:

- Local UUID corresponds to UUID of the User Agent that sends a recording request to the participants of a recording session.
- Remote UUID corresponds to UUID of the User Agent that recieves the recording request in a recording session.

Session-ID Handling

CUBE Media Proxy generates a unique UUID locally, and this UUID is passed as local UUID value in the Session-ID header of the following SIP request and response:

- Request to primary and optional recorders.
- Response to Unified CM (Network-Based Recording) or CUBE (SIPREC-Based).

The following events are involved in the Session-ID handling by CUBE Media Proxy:

1. The initial Invite received by CUBE Media Proxy includes a local UUID generated by the originating platform and a null remote UUID as shown in the following example.

```
Session-ID: db248b6cbdc547bbc6c6fdfb6916eeb;remote=00000000000000000000000000000000
```

2. When sending an Invite to the primary recorder, CUBE Media Proxy generates a new UUID to use for the local Session Identifier. The remote UUID remains null.

```
Session-ID: 8dfb2f2e1d4c518db6122080fb8b1d83;remote=00000000000000000000000000000000
```

3. The subsequent `200 OK` response from the primary recorder includes a local session identifier that it generated and the UUID provided by CUBE Media Proxy in the Invite as the remote session identifier.

```
Session-ID: 4fd24d9121935531a7f8d750ad16e19;remote=8dfb2f2e1d4c518db6122080fb8b1d83
```

4. When sending a `200 OK` to the originating platform, CUBE Media Proxy uses the UUID it generated as the local session identifier and the UUID it received initially as the remote session identifier.

```
Session-ID: 8dfb2f2e1d4c518db6122080fb8b1d83;remote=db248b6cbdc547bbc6c6fdfb6916eeb
```

5. CUBE Media Proxy sends a forking request to the remaining four recorders with Session-ID header containing the same locally generated UUID as the local UUID and a "NULL" value for the remote UUID.

```
Session-ID: 8dfb2f2e1d4c518db6122080fb8b1d83;remote=00000000000000000000000000000000
```

6. CUBE Media Proxy receives `200OK` response from the remaining four recorders. The Session-ID header of the response message from each recorder contains UUID of the recorder as the local UUID and the locally generated UUID by the CUBE Media Proxy as the remote UUID.

```
Session-ID: 4fd24d9121935531a7f8d750ad17f20;remote=8dfb2f2e1d4c518db6122080fb8b1d83
```

7. In NBR mode, CUBE Media Proxy sends a SIP Info Message to Unified CM. For more information on SIP Info Message, see SIP Info Messages from CUBE Media Proxy to Unified CM, on page 11. The Session-ID header of the SIP Info Message contains locally generated UUID by CUBE Media Proxy as local UUID and the UUID of Unified CM as the remote UUID.

```
Session-ID: 8dfb2f2e1d4c518db6122080fb8b1d83;remote=db248b6cbdc547bbc6c6fdfb6916eeb
```

Recording State Notification

SIP Info Messages from CUBE Media Proxy to Unified CM

After trying or establishing an NBR session with the recorders, the CUBE Media Proxy sends SIP Info message to Unified CM to provide the consolidated status of all the recorders.

A SIP Info message is sent during the following stages of a recording session:

1. Initial Call: After receiving a response from all the configured recorders during the initial call, a SIP Info message with status of each recorder is sent to the initiator of the recording session.
2. Mid-Call: When the status of any of the recorders changes during a call, another SIP Info message with status of each recorder is sent to the initiator of the recording session. A change in status may result from any of the recorders sending a "BYE" or rejecting a midcall RE-INIVITE.

Note The examples in the following sections illustrate CUBE Media Proxy forking to two of the maximum five destinations.

XML Format of a SIP Info Message

The Content-Type header present in the SIP Info message is:

```
Content-Type:application/x-cisco-proxy-recording-status+xml
```

The following is the XML format of a SIP info message.

```
<recorderList>
    <recorder>
         <uri>recorder1</uri>
         <recordertype>Mandatory</recordertype>
         <status>Success</status>
         <errormessage>null</errormessage>
    </recoder>
    <recorder>
         <uri>recorder2</uri>
         <recordertype>Mandatory</recordertype>
         <status>Failed</status>
        <errormessage>SIP error code received from Recorder</errormessage>       </recoder>
</recorderList>
```

Table 2: Details of XML Tag and Data Type

XML Tag	Data Type
`uri` (Mandatory)	String
`recordertype` (Mandatory)	Enum (Mandatory, Optional)
`status` (Mandatory)	Enum (Success, Failed)
`errormessage` (Optional)	String

Note The primary recorder in a secure forking scenario functions the same way as a mandatory recorder functions in a nonsecure forking scenario except that the `recorderType` tag is shown as optional. The following is the XML format of a SIP INFO message in a combination of secure and nonsecure forking scenario:

```
<recorderList>
    <recorder>
        <recorderType>Optional</recorderType>
        <status>Success</status>
    </recorder>
    <recorder>
        <recorderType>Optional</recorderType>
        <status>Success</status>
    </recorder>
    <recorder>
        <recorderType>Optional</recorderType>
        <status>Success</status>
    </recorder>
    <recorder>
        <recorderType>Optional</recorderType>
        <status>Success</status>
    </recorder>
    <recorder>
        <recorderType>Optional</recorderType>
        <status>Success</status>
    </recorder>
</recorderList>
```

SIP Info Message Sent During the Initial Call

SIP Info Message Sent During the Initial Call (All the Recorders as Optional)

For information on how to configure the recorders as Optional, see Step 3 and Step 4 of Configure CUBE Media Proxy, on page 16.

The SIP Info Message that is sent during a recording session depends on the scenarios that are given in the following table:

Table 3: Call Scenarios and Recorder Status During the Initial Call with All Recorders as Optional

Scenario	<status> of *recorder-1* in a SIP Info Message	<status> of *recorder-2* in a SIP Info Message
Call to the primary recorder *recorder-1* is established and forking to *recorder-2* is triggered successfully.	`<success>`	`<success>`
Call to the primary recorder *recorder-1* is established and forking to *recorder-2* is rejected with `503 Service Unavailable`.	`<success>`	`<failure>`

Scenario	<status> of *recorder-1* in a SIP Info Message	<status> of *recorder-2* in a SIP Info Message
Call to the primary recorder *recorder-1* is established and there is no response from *recorder-2* to the forking request.	`<success>`	`<failure>`
Call to the recorder *recorder-1* and *recorder-2* is rejected with `503 Service Unavailable`.	`<failure>`	`<failure>`
There is no response from *recorder-1* or *recorder-2* are down.	`<failure>`	`<failure>`
recorder-1 and *recorder-2* responds to the call with a `488 Not Acceptable Here` response.	`<failure>`	`<failure>`
recorder-1 and *recorder-2* reponds to the call with a `600 Busy Everywhere` response.	`<failure>`	`<failure>`

Note

- After a SIP Info Message is sent, a `200 OK` response is received from the initiator of the recording session.
- In all failure scenarios, an error code is sent in the **<errormessage>**.

SIP Info Message Sent During the Initial Call (One Recorder as Mandatory and Remaining as Optional)

For information on how to configure the recorders as Mandatory, see Step 3, Step 4 and, Step 5 of Configure CUBE Media Proxy, on page 16.

The SIP Info Message that is sent during a recording session depends on the scenarios that are given in the following table.

Table 4: Call Scenarios and Recorder Status During the Initial Call with a Mandatory Recorder

Scenario	<status> of *recorder-1* in a SIP Info Message	<status> Of *recorder-2* in a SIP Info Message
Call to the mandatory recorder *recorder-1* is established and forking to the optional recorder *recorder-2* is triggered successfully.	`<success>`	`<success>`

Scenario	<status> of *recorder-1* in a SIP Info Message	<status> Of *recorder-2* in a SIP Info Message
Call to the mandatory recorder *recorder-1* is rejected with a failure message and hence the optional recorder *recorder-2* is not tried.	`<failure>`	`<cancelled>`
Call to the mandatory recorder *recorder-1* is established and when the optional recorder *recorder-2* is tried, the mandatory recorder disconnects with a `BYE`.	`<failure>` **Note** `BYE` is sent in the `<errormessage>`.	`<cancelled>` **Note** The connection to the optional recorder is cancelled as the primary recorder disconnects.
After the call is established with a mandatory recorder *recorder-1* and the optional recorder *recorder-2*, the mandatory recorder disconnects with a `BYE`.	`<failure>` **Note** `BYE` is sent in the `<errormessage>`.	`<disconnected>` **Note** The optional recorder is disconnected.

Note

- After a SIP Info Message is sent, a `200 OK` response is received from the initiator of the recording session. Unified CM sends a `415 Unsupported Media Type` message if the INFO sent from CUBE Media Proxy has a malformed XML body.
- For all failure scenarios, an error code is sent in the `<errormessage>`.

How to Configure CUBE Media Proxy

How to Configure CUBE Media Proxy for Network-Based Recording Solutions

Following are the steps to configure CUBE Media Proxy for Network-Based Recording solutions:

Configure Outbound Dial-Peers to the Recorders

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *recorder-dial-peer-tag* **voip**
4. **destination-pattern** [+] *string*
5. **session protocol sipv2**
6. **session target ipv4:**[*recording-server-destination-address* | *recording-server-dns*]
7. **session transport** [**udp**| **tcp** | **tls**]

8. (Optional) **voice-class sip srtp crypto <crypto-tag>** OR **srtp pass-thru**
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	**dial-peer voice** *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 4	**destination-pattern** [+] *string* **Example:** `Device(config-dial-peer)# destination-pattern 595959`	Specifies either the prefix or full E.164 number required to reach the recorder. A destination pattern must not include regular expressions in this case. **Note** Alternatively, "destination uri" may be used.
Step 5	**session protocol sipv2** **Example:** `Device(config-dial-peer)# session protocol sipv2`	Configures the VoIP dial peer to use Session Initiation Protocol (SIP).
Step 6	**session target ipv4:**[*recording-server-destination-address* | *recording-server-dns*] **Example:** `Device(config-dial-peer)# session target ipv4:198.51.100.1`	Specifies the target network address for the recorder. Keyword and argument are as follows: • **ipv4:** *destination address* --IP address of the media target. **Note** Cisco Unified SIP Proxy may be used to route or load balance forked sessions between a group of recorders. In this case, the Unified SIP Proxy IPv4 address should be configured as the session target.
Step 7	**session transport** [**udp**| **tcp** | **tls**] **Example:** `Device(config-dial-peer)# session transport tcp`	Configures a VoIP dial peer to use TCP. Using the **session transport** command, you can also configure UDP and TLS protocols.

	Command or Action	Purpose

Step 8	(Optional) **voice-class sip srtp crypto <crypto-tag>** OR **srtp pass-thru**
	Example:
	`Device(config-dial-peer)#voice-class sip srtp crypto 20`
	OR
	`Device(config-dial-peer)#srtp pass-thru`
		Configures SRTP crypto profile on the dial-peer.
		OR
		Configure the SRTP pass through on the outbound dial-peer for incoming INVITE.
		Note • This step is optional and is required only for secure media forking.
		• The **voice-class sip srtp crypto <crypto-tag>** is configured for RTP-SRTP Interworking.
		• The **srtp pass-thru** is configured for SRTP-SRTP pass through.

Step 9	**end** **Example:** `Device(config-dial-peer)# end`	Returns to privileged EXEC mode.

Configure CUBE Media Proxy

Before you begin

For secure forking, outbound dial peers must be configured for TLS or SRTP. For further information, refer to Configuring CUBE for SIP TLS.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **media profile recorder** *profile-tag*
4. **media-recording proxy** [*dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5*]
5. **media-recording proxy secure** [*dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5*]
6. **proxy policy mandatory** *dial-peer-tag*
7. **exit**
8. **media class** *tag*
9. **recorder profile** *tag*
10. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
	`Device> enable`	
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-recording proxy** [*dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5*] **Example:** `Device(cfg-mediaprofile)# media-recording proxy 8000 8001 8002`	Configures the dial-peers for forking. The **proxy** configures the first dial-peer of the sequence for establishing a back-to-back (B2B) call, and the remaining dial-peers for media forking. **Note** You can specify maximum of five dial-peer tags.
Step 5	**media-recording proxy secure** [*dial-peer-tag1 dial-peer-tag2 dial-peer-tag3 dial-peer-tag4 dial-peer-tag5*] **Example:** `Device(cfg-mediaprofile)# media-recording proxy secure 9000 9001 9002`	From Cisco IOS XE Bengaluru 17.5.1a onwards, CUBE Media Proxy supports both secure and nonsecure forking. You can configure the dial-peers for both secure and nonsecure forking. The permitted number of configured secure and nonsecure dial peers for forking is five. The behaviour in Cisco IOS XE Bengaluru 17.4.1a and earlier releases is unchanged if there are no secure dial peers configured. **Note** • All secure dial peers must use the same **voice class srtp-crypto** profile.
Step 6	**proxy policy mandatory** *dial-peer-tag* **Example:** `Device(cfg-mediaprofile)# proxy policy mandatory 8001`	(Optional) Specifies the dial peer that must be connected before other forks are attempted. **Note** • The **proxy policy mandatory** command cannot be used when dial peers are configured using **media recording proxy** secure command. • Only one mandatory dial peer may be configured for each profile. • The mandatory dial peer must be one of those configured with the **media-recording proxy** command.

	Command or Action	Purpose
Step 7	**exit** **Example:** `Device(cfg-mediaprofile)# exit`	Exits media profile configuration mode.
Step 8	**media class** *tag* **Example:** `Device(config)# media class 100`	Configures a media class and enters media class configuration mode.
Step 9	**recorder profile** *tag* **Example:** `Device(cfg-mediaclass)# recorder profile 100`	Configures the media profile recorder.
Step 10	**exit** **Example:** `Device(cfg-mediaclass)# exit`	Exits media class configuration mode.

Configure Inbound Dial-Peer from Unified CM

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *call-manager-dial-peer-tag* **voip**
4. **incoming uri** {**from** | **request** |**to** | **via** } *tag*
5. **media-class** *tag*
6. (Optional) **srtp pass-thru**
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.

	Command or Action	Purpose
Step 3	**dial-peer voice** *call-manager-dial-peer-tag* **voip** **Example:** `Device(config)# dial-peer voice 1000 voip`	Configures an inbound dial peer and enters the dial peer voice configuration mode.
Step 4	**incoming uri** {**from** | **request** |**to** | **via** } *tag* **Example:** `Device(config-dial-peer)# incoming uri via 101`	Configures the voice class to match the VoIP dial-peer to the URI of an incoming call from Unified CM using the header in an incoming SIP INVITE message. **Note** For more information on **incoming uri** command, see incoming uri.
Step 5	**media-class** *tag* **Example:** `Device(config-dial-peer)# media-class 100`	Configures media class on the inbound dial peer from Unified CM.
Step 6	(Optional) **srtp pass-thru** **Example:** `Device(config-dial-peer)#srtp pass-thru`	Configure the SRTP pass through on the inbound dial peer for incoming INVITE. **Note** This step is optional and is required only for secure media forking. The **srtp pass-thru** is configured for SRTP-SRTP pass through.
Step 7	**exit** **Example:** `Device(cfg-mediaclass)# exit`	Exits media class configuration mode.

How to Configure CUBE Media Proxy for SIPREC Solutions

Following are the steps to configure SIPREC-based CUBE Media Proxy:

1. Configure Outbound Dial-Peers to the Recorders, on page 14.
2. Configure CUBE Media Proxy, on page 16.
3. Configure SIPREC on CUBE. For more information, see SIPREC (SIP Recording).

Verification of CUBE Media Proxy Configuration

You can verify the configuration of CUBE Media Proxy using Unified CM NBR and SIPREC-Based CUBE Media Proxy using the following **show** and **debug** commands.

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

- **debug voip recmsp all**
- **debug ccsip all**
- **debug ccsip messages**(for audio calls)

  The CUBE Media Proxy sends INVITEs to the recorders with a single stream, which successfully forks the primary call to the recorders. INVITEs to recorders have a single m-line with a send-only attribute.

- **show voip rtp connections**

  Displays Real-Time Transport Protocol (RTP) connections.

  Example:

  For CUBE Media Proxy with Unified CM NBR, recording sessions consist of two sets of RTP streams that are set up independently for near-end and far-end streams. The following example shows RTP connections from 198.51.100.1 is forked to three recorders 8.41.17.71 to 73.

  This example shows NBR with 3 recorders. Two inbound INVITEs (one each for near-end or far-end).

```
Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use:8
Port range not configured
Min   Max   Ports     Ports     Ports

Media-Address Range                    Port  Port   Available  Reserved  In-use
Global Media Pool                      8000  48198  19999      101        8
VoIP RTP active connections :
No. CallId     dstCallId  LocalRTP  RmtRTP     LocalIP       RemoteIP      MPSS
VRF
1   100        101        8218      8372     198.51.100.1    192.0.2.1      NO
NA
2   101        100        8220      9000     8.43.21.69      8.41.17.71     NO
NA
3   104        103        8222      9238     8.43.21.69      8.41.17.72     NO
NA
4   107        106        8224      9250     8.43.21.69      8.41.17.73     NO
NA
5   108        109        8226      8374     198.51.100.1    192.0.2.1      NO
NA
6   109        108        8228      9002     8.43.21.69      8.41.17.71     NO
NA
7   112        111        8230      9240     8.43.21.69      8.41.17.72     NO
NA
8   115        114        8232      9252     8.43.21.69      8.41.17.73     NO
NA
Found 8 active RTP connections
```

  For CUBE Media Proxy using SIPREC, both near-end and far-end streams are established with the same inbound INVITE, which includes the detail in 2 m-lines. The following example shows how the inbound RTP connections are established before creating the RTP connections for five forks.

  This example shows SIPREC with 5 recorders. One inbound INVITE (both near-end or far-end streams).

```
Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 19999, Ports Reserved: 101, Ports in Use: 12
Port range not configured
Min   Max   Ports     Ports     Ports

Media-Address Range                    Port  Port  Available Reserved  In-use
Global Media Pool                      8000  48198 19999     101       12
```

```
VoIP RTP active connections :
No. CallId     dstCallId  LocalRTP RmtRTP   LocalIP        RemoteIP        MPSS    VRF

1     200        202        8108     6012     198.51.100.1     192.0.2.1     NO     NA
2     201        203        8110     6014     198.51.100.1     192.0.2.1     NO     NA
3     202        200        8112     6004       8.43.21.69     8.41.17.71    NO     NA
4     203        201        8114     8882       8.43.21.69     8.41.17.71    NO     NA
5     208        204        8116     6000       8.43.21.69     8.41.17.72    NO     NA
6     209        204        8118     8886       8.43.21.69     8.41.17.72    NO     NA
7     212        205        8120     6008       8.43.21.69     8.41.17.73    NO     NA
8     213        205        8122     9990       8.43.21.69     8.41.17.73    NO     NA
9     216        206        8124     6024       8.43.21.69     8.41.17.74    NO     NA
10    217        206        8126     9978       8.43.21.69     8.41.17.74    NO     NA
11    220        207        8128     6016       8.43.21.69     8.41.17.75    NO     NA
12    221        207        8130     9968       8.43.21.69     8.41.17.75    NO     NA
Found 12 active RTP connections
```

- **show voip recmsp session**

  Displays active recording Media Service Provider (MSP) session information internal to CUBE Media Proxy.

  Following is the sample output for CUBE Media Proxy using Unified CM NBR or SIPREC-Based CUBE Media Proxy:

```
Device# show voip recmsp session

RECMSP active sessions:
MSP Call-ID              AnchorLeg Call-ID        ForkedLeg Call-ID
103                        99                             107
104                        99                             111
105                        99                             115
106                        99                             119
Found 4 active sessions
```

- **show voip recmsp session detail call-id** *call-id*

  Displays detailed information about the recording MSP Call ID.

  Example:

  Following is the sample output for CUBE Media Proxy using Unified CM NBR:

```
Device# show voip recmsp session detail call-id 104
RECMSP active sessions:
Detailed Information
=========================
Recording MSP Leg Details:
Call ID: 103
GUID : 7C5946D38ECD

AnchorLeg Details:
Call ID: 100
Forking Stream type: voice-nearend
Participant: 10000

Non-anchor Leg Details:
Call ID: 101
Forking Stream type: voice-farend
Participant: 708090

Forked Leg Details:
Call ID: 104
```

```
Voice Near End Stream CallID 104
Stream State ACTIVE
Found 1 active sessions
```

In SIPREC-based CUBE Media Proxy, there are two voice near-end streams for the forked call leg. Following is the sample output:

```
Device# show voip recmsp session detail call-id 208
RECMSP active sessions:
Detailed Information
=========================
Recording MSP Leg Details:
Call ID: 204
GUID : C710812A808A

AnchorLeg Details:
Call ID:  200
Forking Stream type: voice-nearend
Participant: sipp

Non-anchor Leg Details:
Call ID: 202
Forking Stream type: voice-farend
Participant: 9876

Forked Leg Details:
Call ID: 208
Voice Near End Stream CallID 208
Stream State ACTIVE
Voice Near End Stream CallID 209
Stream State ACTIVE
Found 1 active sessions
```

- **show voip rtp forking**

  Displays RTP media-forking connections.

  Example:

  Following is the sample output for CUBE Media Proxy using Unified CM NBR:

```
Device# show voip rtp forking
VoIP RTP active forks :
 Fork 1
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 1
     remote ip 8.41.17.72,  remote port 9238,  local port 8222
       codec g711ulaw,  logical ssrc 0x53
       packets sent 29687,  packets received 0
   stream type voice+dtmf-nearend (4): count 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count 0
   stream type video-nearend (8): count 0
   stream type video-farend (9): count 0
   stream type application (10): count 0
 Fork 2
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 1
     remote ip 8.41.17.73,  remote port 9250,  local port 8224
       codec g711ulaw,  logical ssrc 0x53
       packets sent 29687,  packets received 0
```

```
   stream type voice+dtmf-nearend (4): count 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count 0
   stream type video-nearend (8): count 0
   stream type video-farend (9): count 0
   stream type application (10): count 0
Fork 3
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 1
     remote ip 8.41.17.72,  remote port 9240,  local port 8230
       codec g711ulaw,  logical ssrc 0x58
       packets sent 2980,  packets received 0
   stream type voice+dtmf-nearend (4): count 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count 0
   stream type video-nearend (8): count 0
   stream type video-farend (9): count 0
   stream type application (10): count 0
 Fork 4
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 1
     remote ip 8.41.17.73,  remote port 9252,  local port 8232
       codec g711ulaw,  logical ssrc 0x58
       packets sent 2980,  packets received 0
   stream type voice+dtmf-nearend (4): count 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count 0
   stream type video-nearend (8): count 0
   stream type video-farend (9): count 0
   stream type application (10): count 0
```

Following is the sample output for SIPREC-Based CUBE Media Proxy:

```
Device# show voip rtp forking
VoIP RTP active forks :
 Fork 1
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 2
     remote ip 8.41.17.72,  remote port 6000,  local port 8116
       codec g711ulaw,  logical ssrc 0x53
       packets sent 29687,  packets received 0
     remote ip 8.41.17.72,  remote port 8886,  local port 8118
       codec g711ulaw,  logical ssrc 0x53
       packets sent 1296,  packets received 0
   stream type voice+dtmf-nearend (4): count 0
   stream type voice+dtmf-farend (6): count 0
   stream type video (7): count 0
   stream type video-nearend (8): count 0
   stream type video-farend (9): count 0
   stream type application (10): count 0
Fork 2
   stream type voice-only (0): count 0
   stream type voice+dtmf (1): count 0
   stream type dtmf-only (2): count 0
   stream type voice-nearend (3): count 2
     remote ip 8.41.17.73,  remote port 6008,  local port 8120
       codec g711ulaw,  logical ssrc 0x53
       packets sent 29687,  packets received 0
```

```
      remote ip 8.41.17.73,  remote port 9990,  local port 8122
        codec g711ulaw,  logical ssrc 0x53
        packets sent 1296,  packets received 0
    stream type voice+dtmf-nearend (4): count 0
    stream type voice+dtmf-farend (6): count 0
    stream type video (7): count 0
    stream type video-nearend (8): count 0
    stream type video-farend (9): count 0
    stream type application (10): count 0
 Fork 3
    stream type voice-only (0): count 0
    stream type voice+dtmf (1): count 0
    stream type dtmf-only (2): count 0
    stream type voice-nearend (3): count 2
      remote ip 8.41.17.74,  remote port 6024,  local port 8124
        codec g711ulaw,  logical ssrc 0x53
        packets sent 29687,  packets received 0
      remote ip 8.41.17.74,  remote port 9978,  local port 8126
        codec g711ulaw,  logical ssrc 0x53
        packets sent 1296,  packets received 0
    stream type voice+dtmf-nearend (4): count 0
    stream type voice+dtmf-farend (6): count 0
    stream type video (7): count 0
    stream type video-nearend (8): count 0
    stream type video-farend (9): count 0
    stream type application (10): count 0
 Fork 4
    stream type voice-only (0): count 0
    stream type voice+dtmf (1): count 0
    stream type dtmf-only (2): count 0
    stream type voice-nearend (3): count 2
      remote ip 8.41.17.75,  remote port 6016,  local port 8128
        codec g711ulaw,  logical ssrc 0x53
        packets sent 29687,  packets received 0
      remote ip 8.41.17.75,  remote port 9968,  local port 8130
        codec g711ulaw,  logical ssrc 0x53
        packets sent 1296,  packets received 0
    stream type voice+dtmf-nearend (4): count 0
    stream type voice+dtmf-farend (6): count 0
    stream type video (7): count 0
    stream type video-nearend (8): count 0
    stream type video-farend (9): count 0
    stream type application (10): count 0
```

- **show call active voice compact**

  Displays a compact version of voice CallsInProgress. An extra call leg is displayed for media forking.

  Example:

  Following is a sample using NBR:

```
Device# show call active voice compact
<callID>  A/OFAX  T<sec>  Codec       type        Peer Address     IP R<ip>:<udp>
Total call-legs: 8
 100     ANS      T644    g711ulaw    VOIP        P10000        192.0.2.1:8372
 101     ORG      T644    g711ulaw    VOIP        P708090       8.41.17.71:9000
 104     ORG      T643    g711ulaw    VOIP        P708090       8.41.17.72:9238
 107     ORG      T643    g711ulaw    VOIP        P708090       8.41.17.73:9250
 108     ANS      T642    g711ulaw    VOIP        P10000        192.0.2.1:8374
 109     ORG      T642    g711ulaw    VOIP        P708090       8.41.17.71:9002
 112     ORG      T641    g711ulaw    VOIP        P708090       8.41.17.72:5240
 115     ORG      T641    g711ulaw    VOIP        P708090       8.41.17.72:9252
```

Following is a sample output using SIPREC:

```
Device# show call active voice compact
<callID>  A/O FAX T<sec> Codec       type        Peer Address       IP R<ip>:<udp>
Total call-legs: 6
       200 ANS     T644   g711ulaw    VOIP        P10000        192.0.2.1:8108
       202 ORG     T644   g711ulaw    VOIP        P708090      8.41.17.71:8112
       208 ORG     T643   g711ulaw    VOIP        P708090      8.41.17.72:8116
       212 ORG     T643   g711ulaw    VOIP        P708090      8.41.17.73:8120
       216 ORG     T643   g711ulaw    VOIP        P708090      8.41.17.74:8124
       220 ORG     T643   g711ulaw    VOIP        P708090      8.41.17.75:8128
```

- **show sip-ua calls**

  Displays active user agent client (UAC) and user agent server (UAS) information on SIP calls.

  Example:

  Following is the sample output for CUBE Media Proxy using Unified CM NBR:

```
Device# show sip-ua calls
Total SIP call legs:3, User Agent Client:2, User Agent Server:1
SIP UAC CALL INFO
Call 1
  SIP Call ID              : 4091A49B-308911E8-8008EC4C-8D01D66C@192.0.2.1
  State of the call        : STATE_ACTIVE (7)
  Substate of the call     : SUBSTATE_NONE (0)
  Calling Number           : 808808
  Called Number            : 8453
  Called URI               :
  Bit Flags                : 0xC04018 0x80000100 0x80
  CC Call ID               : 2
  Local UUID               : c7351800dd135daba19758eac6b1dd70
  Remote UUID              : ab9f4823802156aaaa8d62e04aaa2b96
  Source IP Address (Sig ): 192.0.2.1
  Destn SIP Req Addr:Port : [192.0.2.2]:9312
  Destn SIP Resp Addr:Port: [192.0.2.2]:9312
  Destination Name         :
  Number of Media Streams : 1
  Number of Active Streams: 1
  RTP Fork Object          : 0x0
  Media Mode               : flow-through
  Media Stream 1
  State of the stream       : STREAM_ACTIVE
  Stream Call ID            : 2
  Stream Type               : voice-only (0)
  Stream Media Addr Type    : 1
  Negotiated Codec          : g711ulaw (160 bytes)
  Codec Payload Type        : 0
  Negotiated Dtmf-relay     : inband-voice
  Dtmf-relay Payload Type   : 0
  QoS ID                    : -1
  Local QoS Strength        : BestEffort
  Negotiated QoS Strength   : BestEffort
  Negotiated QoS Direction : None
  Local QoS Status          : None
  Media Source IP Addr:Port: [192.0.2.1]:8002
  Media Dest IP Addr:Port   : [192.0.2.2]:9000
  Mid-Call Re-Assocation Count: 0
  SRTP-RTP Re-Assocation DSP Query Count: 0

Options-Ping    ENABLED:NO    ACTIVE:NO
```

Following is the sample output for SIPREC-based CUBE Media Proxy:

```
Device# show sip-ua calls
Total SIP call legs:6, User Agent Client:5, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID                : C711BA13-7E9B11EA-8090D6ED-255EEFA0@8.43.21.69
   State of the call       : STATE_ACTIVE (7)
   Substate of the call    : SUBSTATE_NONE (0)
   Calling Number          : sipp
   Called Number           : 9876
   Called URI              : sip:9876@8.41.17.71:8881
   Bit Flags               : 0xC04018 0x90000100 0x80
   CC Call ID              : 101
   Local UUID              : eeabf35db3d25ca4b8276616cdcf5d15
   Remote UUID             : 8afa5ed7b8a052e29235bade4affcf9e
   Source IP Address (Sig ): 8.43.21.69
   Destn SIP Req Addr:Port : [8.41.17.71]:8881
   Destn SIP Resp Addr:Port: [8.41.17.71]:8881
   Destination Name        : 8.41.17.71
   Number of Media Streams : 2
   Number of Active Streams: 2
   RTP Fork Object         : 0x0
   Media Mode              : flow-through
   Media Stream 1
     State of the stream      : STREAM_ACTIVE
     Stream Call ID           : 101
     Stream Type              : voice+dtmf (1)
     Stream Media Addr Type   : 1
     Negotiated Codec         : g711ulaw (160 bytes)
     Codec Payload Type       : 0
     Negotiated Dtmf-relay    : rtp-nte
     Dtmf-relay Payload Type  : 101
     QoS ID                   : -1
     Local QoS Strength       : BestEffort
     Negotiated QoS Strength  : BestEffort
     Negotiated QoS Direction : None
     Local QoS Status         : None
     Media Source IP Addr:Port: [8.43.21.69]:8112
     Media Dest IP Addr:Port  : [8.41.17.71]:6005
   Media Stream 2
     State of the stream      : STREAM_ACTIVE
     Stream Call ID           : 102
     Stream Type              : voice+dtmf (1)
     Stream Media Addr Type   : 1
     Negotiated Codec         : g711ulaw (160 bytes)
     Codec Payload Type       : 0
     Negotiated Dtmf-relay    : rtp-nte
     Dtmf-relay Payload Type  : 101
     QoS ID                   : -1
     Local QoS Strength       : BestEffort
     Negotiated QoS Strength  : BestEffort
     Negotiated QoS Direction : None
     Local QoS Status         : None
     Media Source IP Addr:Port: [8.43.21.69]:8114
     Media Dest IP Addr:Port  : [8.41.17.71]:8883
   Mid-Call Re-Assocation Count: 0
   SRTP-RTP Re-Assocation DSP Query Count: 0


Options-Ping    ENABLED:NO    ACTIVE:NO
```

- **show voip fpi calls**

Displays the call (both inbound and outbound leg) information at the application level.

Example:

Following is the sample output for CUBE Media Proxy using Unified CM NBR:

```
Device#show voip fpi calls
Number of Calls : 1
---------- ---------- ---------- ----------- --------------- ------------
 confID    correlator   AcallID   BcallID     state           event
---------- ---------- ---------- ----------- --------------- ------------
1005          1       1019       1020       ALLOCATED     DETAIL_STAT_RSP
```

As there are 2-m lines in the incoming invite to SIPREC-based CUBE Media Proxy, two FPI sessions are created. Following is the sample output:

```
Device#show voip fpi calls
Number of Calls : 2
---------- ---------- ---------- ----------- --------------- ------------
 confID    correlator   AcallID   BcallID     state           event
---------- ---------- ---------- ----------- --------------- ------------
  42         13         102         100       ALLOCATED     DETAIL_STAT_RSP
  41         14          99         101       ALLOCATED     DETAIL_STAT_RSP
```

- **show media-proxy sessions**

Displays the inbound and forked Call-ID, Session-ID, and dial peer tag details of the active recording sessions. The "Secure" field in the command output is tagged `Y` if the recording session is secure and `N` if the recording session is nonsecure. The "SIPREC" field in the command output is tagged `Y` for SIPREC-based recording session and `N` for Unified CM-based recording session.

Example:

```
Device# show media-proxy sessions

No.        Call-ID             Session-ID                      Dialpeer    Secure
SIPREC
           Inbound/Forked      LocalUuid;RemoteUuid              Tag        (Y/N)
(Y/N)
=====================================================================================
1           36770/-         a234a20672ce596d969c59ee9767f127;    3          N
Y
                           aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

- **show media-proxy sessions summary**

Displays the active recording session details such as the dial peer tag, IP address, port number, number of failed recording sessions, and total number of recording sessions.

Example:

NBR:

```
Device# show media-proxy sessions summary

 No       Inbound/Forked      Dialpeer-Tag           IP:Port           Total/Failed
Sessions
--------------------------------------------------------------------------------------

 1         Forked                100          ipv4:8.41.17.71:5060             2/0
 2         Forked                200          ipv4:8.41.17.72:5060             2/0
 3         Forked                300          ipv4:8.41.17.73:5060             2/0
 4         Inbound              5678                                           2/0
```

SIPREC:

```
Device# show media-proxy sessions summary

 No       Inbound/Forked      Dialpeer-Tag           IP:Port           Total/Failed
Sessions
--------------------------------------------------------------------------------------

 1         Forked                100          ipv4:8.41.17.71:5060             1/0
 2         Forked                200          ipv4:8.41.17.72:5060             1/0
 3         Forked                300          ipv4:8.41.17.73:5060             1/0
 4         Forked                400          ipv4:8.41.17.74:5060             1/0
 5         Forked                500          ipv4:8.41.17.75:5060             1/0
 6         Inbound              5678                                           1/0
```

- **show media-proxy sessions call-id** *call-id*

  Displays the details of the inbound leg and all the forked legs that are associated with the specified SIP leg call-ID. MSP call-ID is not a valid call-ID for this command. Specify the CCAPI call identifier of the SIP leg.

  Example:

```
Device# show media-proxy sessions call-id 101
CC Call-ID: 100 Inbound-leg
Dur: 00:00:15 tx: 0/0 rx: 1484/296800 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 192.0.2.1:8372 Local-Addr: 192.0.2.1:8218 rtt:0ms pl:0/0ms
Dialpeer-Tag: 5678 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: ab9f4823802156aaaa8d62e04aaa2b96

CC Call-ID: 101 Forked-leg (Primary)
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.71:9000 Local-Addr: 8.43.21.69:8220 rtt:0ms pl:0/0ms
Dialpeer-Tag: 100 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 104 Forked-leg
Dur: 00:00:15 tx: 1480/296000 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.72:9238 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 200 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: dcdf882f0876890b930f3427be7fa5f6

CC Call-ID: 107 Forked-leg
Dur: 00:00:15 tx: 1479/295800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.73:9250 Local-Addr: 8.43.21.69:8224 rtt:0ms pl:0/0ms
Dialpeer-Tag: 300 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: 8df0863a6434263f60e50124dae649e6
```

- **show media-proxy sessions session-id** *WORD*

  Displays the details of the Media Proxy recording sessions that are associated with the specified session-ID. To display the details of a specific call-leg, specify the complete session ID string as, *local-uuid;remote=remote-uuid*. Tokens that are allowed for *WORD* are '*', [0-9], [a-f], and [A-F].

  Example:

```
Device# show media-proxy sessions session-id 6bde661e9767590b930f3427ad6e94e9
CC Call-ID: 100 Inbound-leg
Dur: 00:00:15 tx: 0/0 rx: 1484/296800 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 192.0.2.1:8372 Local-Addr: 192.0.2.1:8218 rtt:0ms pl:0/0ms
```

```
Dialpeer-Tag: 5678 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: ab9f4823802156aaaa8d62e04aaa2b96

CC Call-ID: 101 Forked-leg (Primary)
Dur: 00:00:15 tx: 1484/296800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.71:9000 Local-Addr: 8.43.21.69:8220 rtt:0ms pl:0/0ms
Dialpeer-Tag: 100 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: ab9f4823802156aaaa8d62e04aaa2b96 RemoteUUID: 6bde661e9767590b930f3427ad6e94e9

CC Call-ID: 104 Forked-leg
Dur: 00:00:15 tx: 1480/296000 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.72:9238 Local-Addr: 8.43.21.69:8222 rtt:0ms pl:0/0ms
Dialpeer-Tag: 200 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: dcdf882f0876890b930f3427be7fa5f6

CC Call-ID: 107 Forked-leg
Dur: 00:00:15 tx: 1479/295800 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.73:9250 Local-Addr: 8.43.21.69:8224 rtt:0ms pl:0/0ms
Dialpeer-Tag: 300 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 6bde661e9767590b930f3427ad6e94e9 RemoteUUID: 8df0863a6434263f60e50124dae649e6
```

- **show media-proxy sessions metadata-session-id** *x-session-id*

  Displays the details of the Media Proxy recording sessions based on the x-session-id present in the "From" header of the INVITE from Cisco Unified Communications Manager.

  Example:

```
Device# show media-proxy sessions metadata-session-id 696dd5d3f7755c6abdc438e93d01febf

CC Call-ID: 108 Inbound-leg
Dur: 00:00:46 tx: 0/0 rx: 3105/578880 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 192.0.2.1:8374 Local-Addr: 198.51.100.1:8226 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 1 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 528b282b804c5fd098eaba3696c00de2 RemoteUUID: 4fd8036613424366fe00521d46ea16e3

CC Call-ID: 108 Forked-leg (Primary)
Dur: 00:00:46 tx: 3105/578880 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.71:9002 Local-Addr: 8.43.21.69:8228 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 2 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 4fd8036613424366fe00521d46ea16e3 RemoteUUID: 528b282b804c5fd098eaba3696c00de2

CC Call-ID: 112 Forked-leg
Dur: 00:00:46 tx: 3100/577880 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.72:9240 Local-Addr: 8.43.21.69:8230 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 3 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 528b282b804c5fd098eaba3696c00de2 RemoteUUID: 74ad4a4da25e71f2ba0cdc58b8e22f04

CC Call-ID: 115 Forked-leg
Dur: 00:00:46 tx: 3101/578080 rx: 0/0 lost: 0/0/0 delay: 0/0/0ms
Remote-Addr: 8.41.17.73:9252 Local-Addr: 8.43.21.69:8232 rtt: 0ms pl: 0/0ms
Dialpeer-Tag: 4 Negotiated-Codec: g711ulaw
SRTP-Status: off SRTP-Cipher: NA
LocalUUID: 528b282b804c5fd098eaba3696c00de2 RemoteUUID: 96c06c6fc4809314dc2efe7ada030ed6
```

Supported Features

Mid-Call Message Handling

CUBE Media Proxy using Unified CM NBR or SIPREC support midcall signaling events that involve RE-INVITEs from the initiator of the recording session (Unified CM or Cisco UBE) to the recorders. CUBE Media Proxy handles the RE-INVITEs that request a session refresh, change in SDP for media address, direction or codec, or change SRTP crypto suite/key.

For NBR solutions, CUBE Media Proxy sends status updates of a midcall event to Unified CM using SIP Info messages.

When CUBE Media Proxy establishes a new set of forked sessions, the first is referred to as the primary. Where a destination is configured as mandatory, the destination is always the primary. Where all destinations are optional, the first successfully created session is the primary.

Perform the following steps to handle midcall messages:

1. On receipt of a RE-INVITE, CUBE Media Proxy sends the RE-INVITE to the primary recorder.
2. If the primary destination responds to the RE-INVITE with a BYE, then:
    - If the primary is mandatory, the call and all forks are stopped by sending BYE to the destinations and originator.
    - If the primary is optional, the BYE is acknowledged, but not passed back to the originator. The primary session is maintained in a dormant state and further midcall updates are blocked for the remainder of the call.
3. For other responses, the message from the primary is sent to the originator (Unified CM or CUBE).
4. Where the RE-INVITE requests a change in SDP or SRTP and only if this is successfully acknowledged (200 OK) by the primary, the RE-INVITE is sent to the other destinations.
5. If any of the other destinations respond to the RE-INVITE with a failure, CUBE Media Proxy clears that fork by sending a BYE to that destination. The status of this failed session is provided to Unified CM in an INFO message in NBR configurations.

Secure Recording of Secure Calls and Nonsecure Calls

Secure Recording of Secure Calls

With CUBE Media Proxy using Unified CM NBR, it is possible to extend encrypted calls to forked destinations. In this scenario, call signaling is secured using TLS for each connection between CUBE Media Proxy and Unified CM and recorders. As SRTP passthrough is used for media flows, the cipher suite and encryption key negotiated between Unified CM and the primary destination is used for all forks.

Refer to Configuring SIP TLS to secure signaling on Unified CM and forked legs. SRTP configuration is only required for the Unified CM.

Secure Recording of Nonsecure Calls

From Cisco IOS XE Bengaluru 17.5.1a, CUBE Media Proxy used in NBR or SIPREC mode may be configured to secure specific forked sessions when the original call is not encrypted. In this case, the primary destination must be secured and is treated in the same way as a mandatory destination as described in the message handling section above. Refer to SIP TLS and SRTP-RTP internetworking

Support for High Availability

CUBE Media Proxy may be run on a high availability pair of platforms to ensure that calls and media forks are maintained if hardware failure. Call and forked session state is continuously synchronized between the platforms, ensuring that the standby can seemlessly take over media forwarding and call control if necessary.

High availability is available for Cisco Media Proxy configured for Unified CM NBR or SIPREC using either box-to-box or inbox redundancy options.

The following conditions apply when using CUBE Media Proxy high availability:

- Both Active and Standby platforms must have a common hardware and software configuration.
- Calls are synchronized by establishing a checkpoint with the standby on completion of each INVITE, REINVITE, UPDATE, or BYE message transaction.
- Connections that are not successfully established at the point of switchover are not maintained (as there is no checkpoint for the incomplete message transaction).
- In Unified CM NBR mode, checkpoint information includes call metadata, SRTP context and common session ID for all forked sessions. Checkpoints are created after message flows between a recorder and Unified CM are complete. For example, when an optional recorder sends a BYE, the checkpoint is created after CUBE Media Proxy receives the 200 OK response from Unified CM for the INFO message it sends.
- In SIPREC mode, checkpoint information includes common session ID, but not metadata.

You can use the following **show** commands to monitor the recording sessions on the Active and the Standby instances of CUBE Media Proxy:

- **show call active voice compact**
- **show voip rtp connections**
- **show voip recmsp session**
- **show media-proxy sessions**
- **show media-proxy sessions summary**
- **show sip-ua calls**

Media Latch

By default, CUBE Media Proxy using Unified CM NBR uses source address validation to check if the IP address and port details that are received in the UDP header of the RTP or SRTP packets match with the details in the SDP sent by the SIP User Agent. Packets without matching IP address and port are dropped.

In a typical SCCP-based BiB recording using Unified CM NBR CUBE Media Proxy, Unified CM first sends an SDP with the IP address and a dummy port to the CUBE Media Proxy to get the capabilities of CUBE Media Proxy. Unified CM then sends this SDP to the SCCP phone. The CUBE Media Proxy does not know the BiB IP address and port details of the SCCP phone. In these call flows, the IP address and port details in the media packets that are sent from BiB of the SCCP phone to SCCP phone, are different from the IP address and port details in the packets that are sent from Unified CM to the CUBE Media Proxy.

Media Latching is enabled on Unified CM NBR CUBE Media Proxy by default so that the CUBE Media Proxy learns the remote IP address and port details from the UDP transport header of the first RTP or SRTP packet. Media latching is turned on for every call that flows through the CUBE Media Proxy, and works for initial and midcall scenarios. Media Latching is enabled on the inbound leg (Unified CM leg), such that the media packets are accepted even if they are sent from a source IP address and port that is different from the IP address that is advertised in the SDP.