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 1: Feature Information for Network-Based Recording

Feature Name	Releases	Feature Information
Security Readiness Criteria (SRC)—Modified the command **show sip-ua calls**.	Cisco IOS XE Gibraltar Release 16.11.1a	Command **show sip-ua calls** is modified to display local crypto key and remote cryto key.
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**.

Feature Name	Releases	Feature Information
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)

Note Mid-call gateway recording session stops when the call is on hold. For the use case demonstrating the Hold function on the IP phone, see Call Recording Examples for Network-Based and Phone-Based Recording.

- Any media service parameter change via Re-INVITE or UPDATE from Recording server is not supported Midcall renegotiation and supplementary services can be done through the primary call only.
- Media service parameter change via Re-INVITE or UPDATE message from the recording server is not supported
- Recording is not supported if CUBE is running a TCL IVR application with the exception of survivability.tcl, which is supported with network based recording.

- Media mixing on forked streams is not supported
- Digital Signal Processing (DSP) resources are not supported on forked legs
- RecordTone insertion is not supported with SRTP calls.
- Forking does not stop when RTP stream changes mid call to RTP stream. This is for backward compatibility.
- MediaForkingReason tag is to notify midcall stream events. Notification for codec change is not supported.
- Server Groups in outbound dial-peers towards recorders is not supported.
- Forking a single call on a CUBE using both dial-peer based recording and SIPREC is not supported.

Restrictions for Video Recording

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

Information About Network-Based Recording Using CUBE

Deployment Scenarios for CUBE-based Recording

CUBE as a recording client has the following functions:

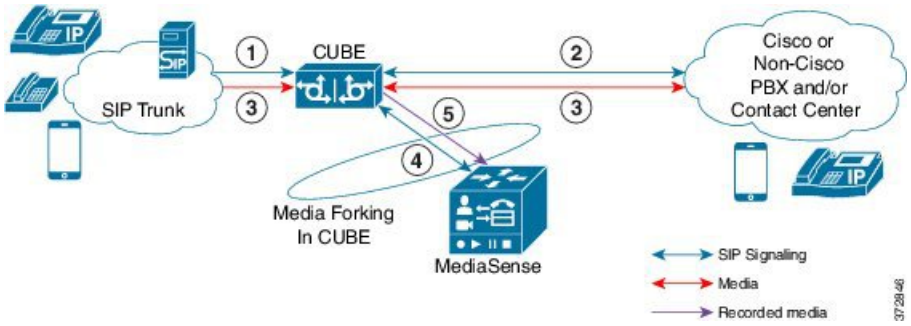
- Acts as a SIP user agent and sets up a recording session (SIP dialog) with the recording server.
- Acts as the source of the recorded media and forwards the recorded media to the recording server.
- Sends information to a server that helps the recording server associate the call with media streams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

Note CUBE simply forwards the RTP streams it receives to the SIP recorder. It does not support omitting any pre-agent VRU activity from the recording.

If you want to omit the VRU segment from a recording, you must use the Unified CVP to route the agent segment of the call back through CUBE. To do this, you need to separate ingress and media forking function from one another, which means you must either route the call through the ingress router a second time, or route it through a second router.

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

Figure 1: 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

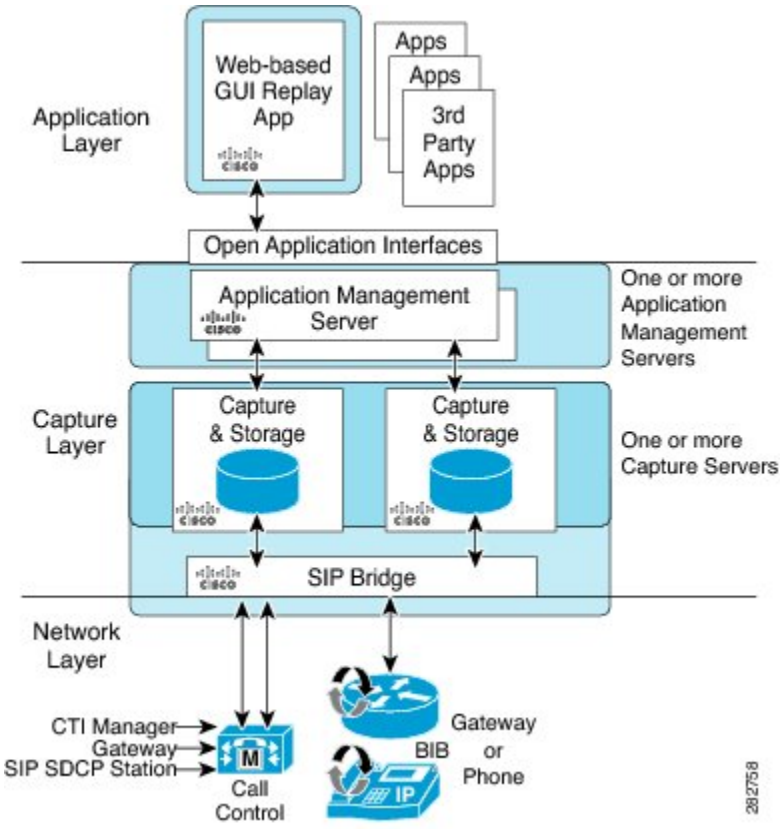
The metadata carried in the SIP session between the recording client and the recording server is to:

- 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.

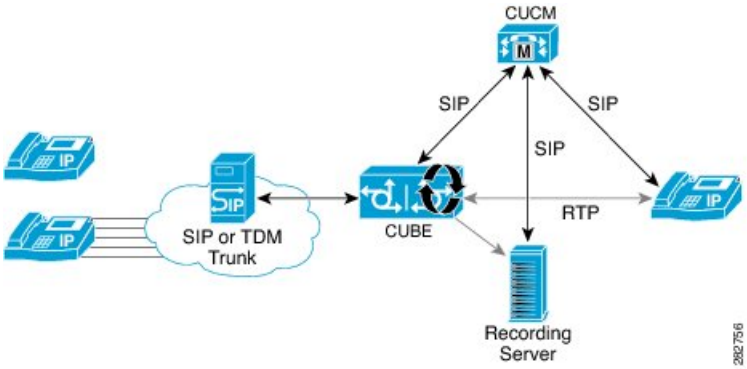
- Enterprise-wide policy management.

Media Forking Topologies

The following topologies support media forking:

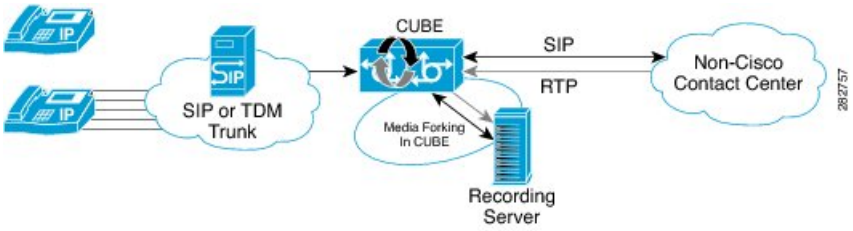
Media Forking with Cisco UCM

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

Procedure

	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	**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	(Optional) **media-type audio** **Example:** `Device(cfg-mediaprofile)# media-type audio`	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.
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.

	Command or Action	Purpose
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. **Note** The predefined valid entries for *string* are the digits 0 to 9, the letters A to F and, the following special characters: • The asterisk (*****) and pound sign (#) that appear on standard touch-tone dial pads. • Plus sign (+), which indicates that the preceding digit occurred one or more times. • Backslash symbol (\), which is followed by a single character, and matches that character. Media Forking functionality does not work with the wildcard entries other than the predefined set.
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

Procedure

	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	(Optional) **media-type audio** **Example:** `Device(cfg-mediaprofile)# media-type audio`	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.

	Command or Action	Purpose
		Note The predefined valid entries for *string* are the digits 0 to 9, the letters A to F and, the following special characters: • The asterisk (*) and pound sign (#) that appear on standard touch-tone dial pads. • Plus sign (+), which indicates that the preceding digit occurred one or more times. • Backslash symbol (\), which is followed by a single character, and matches that character. Media Forking functionality does not work with the wildcard entries other than the predefined set.
Step 12	**session protocol sipv2** **Example:** `Device(config-dial-peer)# session protocol sipv2`	Configures the VoIP dial peer to use Session Initiation Protocol (SIP).
Step 13	**session target ipv4:**[*recording-server-destination-address* | *recording-server-dns*] **Example:** `Device(config-dial-peer)# session target ipv4:10.42.29.7`	Specifies a network-specific address for a dial peer. Keyword and argument are as follows: • **ipv4:** *destination address* --IP address of the dial peer, in this format: *xxx.xxx.xxx.xxx*
Step 14	**session transport tcp** **Example:** `Device(config-dial-peer)# session transport tcp`	Configures a VoIP dial peer to use Transmission Control Protocol (TCP).
Step 15	**end** **Example:** `Device(config-dial-peer)# end`	Returns to privileged EXEC mode.

Verifying the Network-Based Recording Using CUBE

Perform this task to verify the configuration of the Network-Based Recording Using CUBE. The **show** and **debug** commands can be entered in any order.

SUMMARY STEPS

1. **enable**
2. **show voip rtp connections**
3. **show voip recmsp session**

4. **show voip recmsp session detail call-id** *call-id*
5. **show voip rtp forking**
6. **show call active voice compact**
7. **show call active video compact**
8. **show sip-ua calls**
9. **show call active video brief**
10. **debug 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

Procedure

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.
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
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:2, User Agent Client:1, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID                : C9A3AA00-B49A11E8-8018A74B-CD0B0450@10.0.0.1
   State of the call       : STATE_ACTIVE (7)
   Substate of the call    : SUBSTATE_NONE (0)
   Calling Number          : 1234
```

```
Called Number           : 9876
Called URI              : sip:9876@10.0.0.2:9800
Bit Flags               : 0xC04018 0x90000100 0x80
CC Call ID              : 13
Local UUID              : 7d14e2d622ec504f9aaa4ba029ddd136
Remote UUID             : 2522eaa82f505c868037da95438fc49b
Source IP Address (Sig ): 10.0.0.1
Destn SIP Req Addr:Port : [10.0.0.2]:9800
Destn SIP Resp Addr:Port: [10.0.0.2]:9800
Destination Name        : 10.0.0.2
Number of Media Streams : 2
Number of Active Streams: 2
RTP Fork Object         : 0x0
Media Mode              : flow-through
Media Stream 1
  State of the stream      : STREAM_ACTIVE
  Stream Call ID           : 13
  Stream Type              : voice-only (0)
  Stream Media Addr Type   : 1
  Negotiated Codec         : g711ulaw (160 bytes)
  Codec Payload Type       : 0
  Negotiated Dtmf-relay    : inband-voice
  Dtmf-relay Payload Type  : 0
  QoS ID                   : -1
  Local QoS Strength       : BestEffort
  Negotiated QoS Strength  : BestEffort
  Negotiated QoS Direction : None
  Local QoS Status         : None
  Media Source IP Addr:Port: [10.0.0.1]:8022
  Media Dest IP Addr:Port  : [10.0.0.2]:6008
  Local Crypto Suite       :  AES_CM_128_HMAC_SHA1_80 (
                              AEAD_AES_256_GCM
                              AEAD_AES_128_GCM
                              AES_CM_128_HMAC_SHA1_80
                              AES_CM_128_HMAC_SHA1_32 )
  Remote Crypto Suite      :  AES_CM_128_HMAC_SHA1_80
  Local Crypto Key         : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z1234tVb2
  Remote Crypto Key        : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z9876tVb2
Media Stream 2
  State of the stream      : STREAM_ACTIVE
  Stream Call ID           : 14
  Stream Type              : video (7)
  Stream Media Addr Type   : 1
  Negotiated Codec         : h264 (0 bytes)
  Codec Payload Type       : 97
  Negotiated Dtmf-relay    : inband-voice
  Dtmf-relay Payload Type  : 0
  QoS ID                   : -1
  Local QoS Strength       : BestEffort
  Negotiated QoS Strength  : BestEffort
  Negotiated QoS Direction : None
  Local QoS Status         : None
  Media Source IP Addr:Port: [10.0.0.1]:8020
  Media Dest IP Addr:Port  : [10.0.0.2]:9802
  Local Crypto Suite       :  AES_CM_128_HMAC_SHA1_80 (
                              AEAD_AES_256_GCM
                              AEAD_AES_128_GCM
                              AES_CM_128_HMAC_SHA1_80
                              AES_CM_128_HMAC_SHA1_32 )
  Remote Crypto Suite      :  AES_CM_128_HMAC_SHA1_80
  Local Crypto Key         : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z2345tVb2
  Remote Crypto Key        : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z8765tVb2
Mid-Call Re-Assocation Count: 0
SRTP-RTP Re-Assocation DSP Query Count: 0
```

```
Options-Ping    ENABLED:NO    ACTIVE:NO
   Number of SIP User Agent Client(UAC) calls: 1

SIP UAS CALL INFO
Call 1
SIP Call ID                : 1-12049@10.0.0.2
   State of the call       : STATE_ACTIVE (7)
   Substate of the call    : SUBSTATE_NONE (0)
   Calling Number          : 1234
   Called Number           : 9876
   Called URI              : sip:9876@10.0.0.1:5060
   Bit Flags               : 0xC0401C 0x10000100 0x4
   CC Call ID              : 11
   Local UUID              : 2522eaa82f505c868037da95438fc49b
   Remote UUID             : 7d14e2d622ec504f9aaa4ba029ddd136
   Source IP Address (Sig ): 10.0.0.1
   Destn SIP Req Addr:Port : [10.0.0.2]:5060
   Destn SIP Resp Addr:Port: [10.0.0.2]:5060
   Destination Name        : 10.0.0.2
   Number of Media Streams : 2
   Number of Active Streams: 2
   RTP Fork Object         : 0x0
   Media Mode              : flow-through
   Media Stream 1
     State of the stream      : STREAM_ACTIVE
     Stream Call ID           : 11
     Stream Type              : voice-only (0)
     Stream Media Addr Type   : 1
     Negotiated Codec         : g711ulaw (160 bytes)
     Codec Payload Type       : 0
     Negotiated Dtmf-relay    : inband-voice
     Dtmf-relay Payload Type  : 0
     QoS ID                   : -1
     Local QoS Strength       : BestEffort
     Negotiated QoS Strength  : BestEffort
     Negotiated QoS Direction : None
     Local QoS Status         : None
     Media Source IP Addr:Port: [10.0.0.1]:8016
     Media Dest IP Addr:Port  : [10.0.0.2]:6009
     Local Crypto Suite       :  AES_CM_128_HMAC_SHA1_80
     Remote Crypto Suite      :  AES_CM_128_HMAC_SHA1_80
     Local Crypto Key         : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z9876tVb2
     Remote Crypto Key        : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z1234tVb2
   Media Stream 2
     State of the stream      : STREAM_ACTIVE
     Stream Call ID           : 12
     Stream Type              : video (7)
     Stream Media Addr Type   : 1
     Negotiated Codec         : h264 (0 bytes)
     Codec Payload Type       : 97
     Negotiated Dtmf-relay    : inband-voice
     Dtmf-relay Payload Type  : 0
     QoS ID                   : -1
     Local QoS Strength       : BestEffort
     Negotiated QoS Strength  : BestEffort
     Negotiated QoS Direction : None
     Local QoS Status         : None
     Media Source IP Addr:Port: [10.0.0.1]:8018
     Media Dest IP Addr:Port  : [10.0.0.2]:5062
     Local Crypto Suite       :  AES_CM_128_HMAC_SHA1_80
     Remote Crypto Suite      :  AES_CM_128_HMAC_SHA1_80
     Local Crypto Key         : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z8765tVb2
```

```
     Remote Crypto Key        : bTQqZXbgFJddA1hE9wJGV3aKxo5vPV+Z2345tVb2
   Mid-Call Re-Assocation Count: 0
   SRTP-RTP Re-Assocation DSP Query Count: 0


Options-Ping    ENABLED:NO    ACTIVE:NO
   Number of SIP User Agent Server(UAS) calls: 1
```

Step 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:111100010@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

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*